University of California

Los Angeles

Turbo code design for high spectral efficiency

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Electrical Engineering

by

Christina Fragouli

The dissertation of Christina Fragouli is approved.

Greg Pottie

John Villasenor

Panagiotis Christofides

Richard D. Wesel, Committee Chair

University of California, Los Angeles

To my father

Table of Contents

List of Figures

List of Tables

Acknowledgments

Prof. Rick Wesel had the kindness to accept me in his research group, and since then has invariably been the best advisor a student could wish for. His enthusiasm for research, high standard of excellence and integrity, and deep knowledge of his subject have been a great source of inspiration for me. He offered substantial help to every aspect of a graduate student life at UCLA, from highlighting innovative research perspectives to buying new chairs and computers, from funding Brazil conferences to English language suggestions (he did not edit this text). What I most appreciated though was his active involvement in research, his immediate understanding of a problem and fast response with possible solutions and literature references, his ability to communicate subtle notions with clarity and conciseness, and his unlimited availability for questions and comments. This work could not have been made without his constant help and support.

I was also fortunate to have the opportunity to work with Prof. Ali Sayed. His professional expertise and knowledge set an example of excellence for me. Prof. M. Srivastava and Prof. R. Jain have promoted my admission at UCLA, and involved me in research during my first year of studies. Prof. Srivastava' s dedication and expertise were greatly appreciated.

My fellow group members Christos Komninakis, Xueting Liu, Wei Shi and later Mark Shane have greatly contributed to my research with their useful comments and interaction, and have created a friendly and inspiring working environment.

Prof. Greg Pottie, Prof. John Villasenor and Prof. Panagiotis Christofides have served on my Ph.D. committee despite their heavy schedule.

Dr Divsalar and Dr McEliece have very kindly replied to my questions regarding their work, and generously supplied me with relevant material. Dr K. Narayanan and Dr J. Gass have offered very helpful suggestions and comments. Dr N. Seshadri and Dr W. Turin have created an inspiring and motivating environment during my summer internship at AT&T.

Irene Andreadis has warmly welcomed me in her house during my first days in the US, and all the years thereafter. I would like to thank her for her care, her hospitality, and her kindness.

My fellow students and good friends Christos Komninakis, Vaggelis Petsalis, Javier Garcia-Frias, George Kondylis, Peyman Meshat, and Vijay Sivaraman have also helped me with acquiring a technical background through fruitful discussions. I would like to thank them for the many good times.

My friends Athina Markopoulou, Dina Danaka, Ellh Paxou, and Persa Kyritsi, although not in LA, were still very close to me. My grandmother Marika, my grandfather Stelios, my mother and my sister Nataly have offered me their love and support.

Vita

1991–1996	B.S. in Electrical Engineering, National Technical University of Athens.
1996–1998	M.S. in Electrical Engineering, University of California, at Los Angeles.
1998–2000	Ph.D. in Electrical Engineering University of California, at Los Angeles.

Publications

- C. Fragouli, C. Komninakis, and R. D. Wesel, "Minimality under periodic puncturing", submitted in ICC 2001, June 11-15, Helsinki, Finland.

- C. Fragouli and R. D. Wesel, "Turbo encoder design for symbol interleaved parallel concatenated trellis coded modulation," accepted (summer 2000) to IEEE Transactions on Communications.

- C. Fragouli, N. Seshadri and W. Turin, "On the reduced trellis equalization Using the M-BCJR Algorithm," CISS 2000, March 15-17 2000, Boston.

- C. Komninakis, C. Fragouli, A. Sayed and R. D. Wesel, "Adaptive multi-input multi-output fading channel equalization using kalman estimation," ICC 2000, N ew Orleans, Louisiana, June 18-22 2000.

- C. Fragouli and R. D. Wesel, “Semi-random interleaver design criteria,” Communication Theory Symposium at Globecom 99, Rio de Janeiro, Brazil, December 5-9, 1999.

- C. Komninakis, C. Fragouli, R.D. Wesel, A. Sayed, “Channel estimation and equalization in fading,” 33rd Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA. October 24-27, 1999.

- C. Fragouli and R. D. Wesel, “Convolutional codes and matrix control theory,”in proceedings of the 7th International Conference on Advances in Communications and Control, June 28-July 2, 1999, Ath ens, Greece.

- C. Fragouli and R. D. Wesel, “Symbol interleaved parallel concatenated trellis coded modulation ,” in the Communication Theory Miniconference in conjunction with ICC 99, June 6-10, 1999.

- C. Fragouli, V. Sivaraman,and M. B. Srivastava, “Controlled multimedia wireless link sharing vi a enhanced class-based queuing with channel-state-dependent packet scheduling”, in the proceedings of Infocom 98.

- P. Lettieri, C. Fragouli, and M. B.Srivastava, “Low power error control for wireless links”,in the proceedings of Mobicom 97.

Abstract
of the Dissertation

Turbo code design for high spectral efficiency

by

Christina Fragouli
Doctor of Philosophy in Electrical Engineering
University of California, Los Angeles, 2000
Professor Richard D. Wesel, Chair

This thesis addresses turbo-encoder design for coding with high spectral efficiency using parallel concatenated trellis coded modulation (PCTCM) and symbol interleaving. The turbo-encoder design involves the constituent encoder design and the interleaver design. The state space description of convolutional codes serves as the foundation of the constituent encoder optimization. As an application we examine minimality under periodic puncturing. The constituent encoders are optimized for symbol-wise effective free distance, and each has an infinite symbol-wise impulse response. We identify the canonical structures for the constituent encoder search space. In many cases of practical interest, the optimal structure for these constituent encoders connects the memory elements in a single row. To lower the error floor, new semi-random interleaver design criteria and a construction method extends the spread-interleaver concept introduced by Divsalar and Pollara. Simulation results between 0.5 dB and 0.6 dB from constrained capacity for rates of 2 and 4 bits/sec/Hz, show that the proposed system can converge at a lower SNR than previously reported systems.

CHAPTER 1

Introduction

The channel coding theorem establishes the existence of codes that allow transmission through a channel at any rate less than the channel capacity with an arbitrarily small probability of error [CT91, Gal68, Ash65, Sha48, Sha57]. However, there is no systematic procedure for constructing practical codes that meet the specifications of the coding theorem. Before the emergence of turbo codes in 1993 [BCT93], the error correcting codes achieved a reliable performance several dB from capacity [Wes98, J 95]. Turbo codes managed, with finite complexity, to offer performance within fractions of a dB from capacity. Since then, they have increasingly attracted the interest of the coding community, and are becoming part of the next generation of standards.

The main objective of this thesis is to develop design techniques for turbo codes with high spectral efficiency, i.e. multiple bits/sec/Hz. The proposed turbo encoder structure uses parallel concatenated trellis coded modulation (PCTCM) and symbol interleaving. The turbo encoder design involves the constituent encoder design and the interleaver design.

This thesis is organized as follows. Chapter 2 reviews a general description of convolutional encoders that is provided by linear matrix equations, called discrete-time state-space equations in control theory. We introduce the notion of algebraically equivalent encoders and use the rational form theorem to establish the structural properties of an exhaustive set of encoders that are algebraically distinct.

To acquire a better understanding of the state-space equations framework, we discuss how the notions of controllability, observability and minimality in control, are applied to convolutional code design. The notion of output-observability can be used to ensure minimality of a convolutional code.

As an application, we investigate minimality of convolutional encoders after periodic puncturing. A minimal encoder, when periodically punctured, produces a higher rate encoder that may or may not be minimal. We assess the performance of codes that are non-minimal under puncturing, and propose a fast algorithm to determine minimality under puncturing. As an example we optimize rate 1/4 unpuctured codes for Hamming weight under both bit-wise and symbol-wise periodic puncturing. Code tables and simulation results are included.

The turbo encoder design consists of two components, the constituent encoder design and the interleaver design, which are examined in Chapters 3 and 4 respectively.

Chapter 3 proposes a method for parallel concatenated trellis coded modulation using symbol interleaving. We describe the proposed turbo encoder structure, derive the constituent encoder optimization criteria and extend the effective distance bounds to symbol-wise inputs. Based on the results of Chapter 2, the appropriate encoder structure for turbo-code constituent encoders is identified and applied to the special case of PCTCM. The identified search space applies to all turbo-code constituent encoders and is not restricted to symbol interleaving, or trellis coded modulation. As an example, we optimize constituent encoders for bit-interleaved turbo codes coupled with $BPSK$.

Chapter 4 proposes new interleaver design criteria that extend the spread-interleaver concept introduced in [DBP97] to multiple error events. This extension helps to explain why the spread interleaver is specifically designed to be semi-random. We also propose a semi-random interleaver construction method, and discuss the existence of interleavers that meet specified constraint parameters.

Chapter 5 uses the proposed turbo structure and design techniques, to present simulation results for 2 bits/sec/Hz employing 16-QAM and 8-PSK, and 4 - bits/sec/Hz employing 64-QAM$= 2 \times$ 8-PAM. The performance at BER 10^{-5} is between 0.5 dB and 0.6 dB from constrained capacity. The proposed system can converge at a lower SNR than previously reported systems.

The main contributions of the work presented in this thesis, are:

- An algebraic criterion for minimality among range equivalent encoders (Chapter 2).
- Search space for algebraically equivalent encoders (Chapter 2).
- An algorithm to examine whether periodic puncturing causes a convolutional encoder to have zero output loops other than the self-loop around the zero state (Chapter 2).
- Design criteria and distance bounds for turbo code constituent encoders to symbol interleaving (Chapter 3).
- Search space for turbo code constituent encoders(Chapter 3).
- Interleaver design criteria for turbo codes (Chapter 4).
- A semi-random interleaver construction method (Chapter 4).

CHAPTER 2

Convolutional Codes

Codes optimized for a particular distance property can be identified by means of exhaustive search within an appropriate set of potential codes. The main contribution of this chapter is to develop the framework that Chapter 3 uses to determine an appropriate search space for turbo code constituent encoders.

Without concatenation, searching for good trellis codes that maximize the free distance, requires examining only one code within each group of range equivalent encoders, i.e. equivalent in Forney's sense ([For70], [Wes96] definition 16). So it is sufficient to restrict attention within a set of canonical encoders, which are identified by Forney [For70].

Definition 1 Range Equivalent *Two encoders are called range equivalent if they have the same set of output sequences.*

Turbo codes involve optimizing the constituent convolutional encoders for output distance under specific input to output mappings. Range-equivalent codes can have quite different performance. For example, feedback encoders always have a range-equivalent feedforward encoder which would perform poorly with parallel concatenation. In this case an exhaustive search should use a different search space. This chapter examines the structural properties of encoders that should be included in an exhaustive search for turbo code constituent encoders.

The largest set of encoders an exhaustive search might need to examine is the set of encoders that are not strictly equivalent ([Wes96], definition 15) to each other.

Definition 2 Strictly equivalent *Two encoders are called strictly equivalent if they have the same mapping of input to output sequences.*

If two encoders are not strictly equivalent (or range equivalent), we say they are strictly distinct (or range distinct respectively).

A general description of convolutional encoders is provided by linear matrix equations called discrete-time state-space equations in control theory. To acquire a better understanding of the state-space equations framework, we discuss how the notions of controllability, observability and minimality from control theory are applied to convolutional code design. The notion of output-observability can be used to ensure the minimality of a convolutional code. As an application, we investigate minimality of convolutional encoders after periodic puncturing.

2.1 State Space Equations

A general description of a convolutional encoder with k inputs, n outputs, and m memory elements is given by the state-space equations over $GF(2)$:

$$\begin{aligned} s_{j+1} &= s_j\,\mathbf{A} + u_j\,\mathbf{B} \\ x_j &= s_j\,\mathbf{C} + u_j\,\mathbf{D} \end{aligned} \tag{2.1}$$

where s_j is the state vector of dimension $1 \times m$, u_j is the input vector of dimension $1 \times k$, and x_j is the output vector of dimension $1 \times n$.

Matrix $\mathbf{A} \in GF(2)$ of dimension $m \times m$ determines the way the m memory elements are connected. For a feedforward encoder with memory elements con-

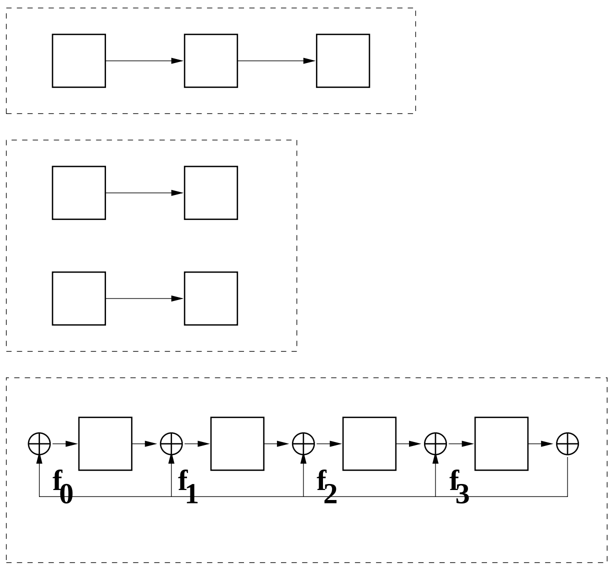


FIGURE 2.1: CANONICAL FORM FOR FEEDFORWARD AND FEEDBACK ENCODERS

nected in a single row, matrix $\mathbf{A}$ has $m-1$ terms "1" in the superdiagonal and all other entries equal to zero. This Toeplitz form matrix is called "backward-shift" [HJ85]. For example, the feedforward encoder with three memory elements connected in one row, depicted in Fig. 2.1, can be described by the backward-shift matrix $\mathbf{A}_1$:

$$\mathbf{A}_1 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

For a feedforward encoder with memory elements connected in R rows, $\mathbf{A}$ is equal to the direct sum of R backward-shift matrices, one corresponding to each of the memory elements.

Definition 3 Direct sum *The direct sum of the matrices* $\mathbf{A}_{11}$, $\mathbf{A}_{22}$, $\ldots$ *is the block diagonal matrix:*

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & 0 & 0 \\ 0 & \mathbf{A}_{22} & 0 \\ 0 & 0 & \ddots \end{bmatrix} \tag{2.2}$$

For example, the feedforward encoder with two rows of memory elements in Fig. 2.1, can be described by $\mathbf{A}_2$:

$$\mathbf{A}_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

For feedforward encoders, there exists an integer $L \leq m$, such that $\mathbf{A}^L = 0$, where m is the number of memory elements.

For a feedback encoder with the memory elements connected in a single row, $\mathbf{A}$ is the companion matrix of the encoder's feedback polynomial. The companion matrix of a polynomial is defined [HJ85] as:

Definition 4 Companion Matrix *Consider the monic polynomial*

$$f(D) = D^4 + f_3D^3 + f_2D^2 + f_1D + f_0$$

The matrix:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ f_0 & f_1 & f_2 & f_3 \end{bmatrix} \tag{2.3}$$

is called companion matrix of the polynomial $f(D)$.

For feedback encoders with R memory rows, $\mathbf{A}$ is the direct sum of R companion matrices, one corresponding to each memory row. For feedback encoders, it holds that $\mathbf{A}^L \neq 0$ for any positive integer L. Moreover, there exists a positive integer $P \leq 2^m$ such that $\mathbf{A}^P = \mathbf{A}$.

Matrix $\mathbf{B} \in GF(2)$ of dimension $k \times m$ determines how the inputs drive the memory elements. For example, the states that can be reached in one trellis step starting from the zero state are the 2^k linear combinations of the rows of $\mathbf{B}$. If the rows of $\mathbf{B}$ are not linearly independent, then the corresponding trellis diagram has parallel transitions.

The encoders with a feedback polynomial that has a nonzero constant term $f_0 = 1$, do not have input-weight-one error events. An input weight one sequence drives the encoder through the sequence of states:

$$0 \to b_i \to b_i\mathbf{A} \to \ldots \to b_i\mathbf{A}^L \to \ldots$$

where L is integer and b_i is a row of $\mathbf{B}$. The nonzero input causes the encoder to go from state zero to state b_i. The encoder can return to the zero state (with zero input), if there exists row b_i and integer L such that $b_i\mathbf{A}^L = 0$. But this would imply that $\mathbf{A}^L$ has left eigenvalue 0, and as a result $\mathbf{A}$ has eigenvalue 0. The characteristic polynomial though, which for companion matrices is equal to the feedback polynomial, since $f_0 = 1$, does not have 0 as a root.

Matrix $\mathbf{C} \in GF(2)$ of dimension $m \times n$ determines how the output depends upon the current state. If an output is systematic, then the corresponding column has zero elements. Finally, matrix $\mathbf{D} \in GF(2)$ of dimension $k \times n$ determines how the current input contributes to the output.

The generator matrix $G(D)$ for an encoder with state space representation $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ can be calculated as:

$$G(D) = \mathbf{D} + \mathbf{B}(D^{-1}\mathbf{I} - \mathbf{A})^{-1})\mathbf{C} \tag{2.4}$$

where D is the indeterminate delay operator and $\mathbf{I}$ the identity matrix. Different $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ encoders may have the same generator matrix.

2.2 Algebraically Equivalent Encoders

Definition 5 Algebraically equivalent *Two encoders are called algebraically equivalent if they map the same input error events to the same output error events.*

If two encoders are not algebraically equivalent, we say they are algebraically distinct.

Definition 6 Error event *([BDM91] pg. 61) An error event is a trellis path of finite length that starts and ends in the zero state, and does not visit the zero state in between.*

The difference between strict equivalence and algebraic equivalence, is that algebraic equivalence refers to error events. Strict equivalence (definition 2), as well as range equivalence (definition 1), refer to sequences that might be infinite or semi-infinite, and are not concerned with the encoder memory state.

Two algebraically equivalent encoders are strictly equivalent. Two encoders are strictly equivalent, if and only if they have the same generator matrix. Two encoders that have the same generator matrix though are not necessarily algebraically equivalent. For example, the encoder $\{\mathbf{A_1}, \mathbf{B_1}, \mathbf{C_1}, \mathbf{D_1}\}$:

$$\begin{aligned}
[s_1\ s_2\ s_3\ s_4]_{j+1} &= [s_1\ s_2\ s_3\ s_4]_j \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{A_1}} + \quad [u_1\ u_2\ u_3]_j \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}}_{\mathbf{B_1}} \\
[y_1\ y_2\ y_3\ y_4]_j &= [s_1\ s_2\ s_3\ s_4]_j \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{C_1}} + \quad [u_1\ u_2\ u_3]_j \underbrace{\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}}_{\mathbf{D_1}}
\end{aligned} \tag{2.5}$$

is strictly equivalent to the encoder $\{\mathbf{A_2}, \mathbf{B_2}, \mathbf{C_2}, \mathbf{D_2}\}$:

$$\begin{aligned}
[s_1\ s_2]_{j+1} &= [s_1\ s_2]_j \underbrace{\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}}_{\mathbf{A_2}} + [u_1\ u_2\ u_3]_j \underbrace{\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}}_{\mathbf{B_2}} \\
[y_1\ y_2\ y_3\ y_4]_j &= [s_1\ s_2]_j \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{C_2}} + [u_1\ u_2\ u_3]_j \underbrace{\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}}_{\mathbf{D_2}}
\end{aligned} \tag{2.6}$$

Fig. 2.2 and 2.3 provide an example to illustrate how encoders $\{\mathbf{A_1}, \mathbf{B_1}, \mathbf{C_1}, \mathbf{D_1}\}$ and $\{\mathbf{A_2}, \mathbf{B_2}, \mathbf{C_2}, \mathbf{D_2}\}$ are strictly equivalent but not algebraic equivalent. Assume that the input to both encoders is the semi-infinite sequence: $\{3\ \ 0\ \ 0\ \ 0\ldots\}$, then both encoders produce the same output sequence: $\{3\ \ 0\ \ 0\ \ 0\ldots\}$. Encoder

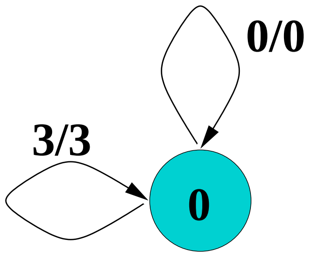


FIGURE 2.2: ENCODER $\{\mathbf{A_2}, \mathbf{B_2}, \mathbf{C_2}, \mathbf{D_2}\}$ HAS AN ERROR EVENT OF LENGTH ONE

$\{\mathbf{A_2}, \mathbf{B_2}, \mathbf{C_2}, \mathbf{D_2}\}$ (Fig. 2.2) goes through an error event of length one, with input and output equal to 3, and then stays in the zero loop around the zero state. Encoder $\{\mathbf{A_1}, \mathbf{B_1}, \mathbf{C_1}, \mathbf{D_1}\}$ with input 3 and output 3, exits the zero state to en-

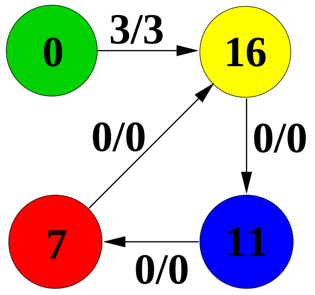


FIGURE 2.3: ENCODER $\{\mathbf{A_1}, \mathbf{B_1}, \mathbf{C_1}, \mathbf{D_1}\}$DOES NOT HAVE AN ERROR EVENT OF LENGTH ONE

ter the zero-input zero-output loop around the states {16, 11, 7}, where it stays forever, i.e. does not return to zero state. This encoder does not have an error event of length one with input 3 and output 3. Actually, it does not have any error event of length one. Thus it is not algebraically equivalent to the encoder $\{\mathbf{A_2}, \mathbf{B_2}, \mathbf{C_2}, \mathbf{D_2}\}$.

Strict equivalence in turn implies range equivalence. Two encoders with generator matrices $G_1(D)$ and $G_2(D)$ are range equivalent if and only if there exists a nonsingular matrix $T(D)$ such that $G_1(D) = T(D)G_1(D)$. For strictly equivalent encoders $T(D) = \mathbf{I}$ where $\mathbf{I}$ is the identity matrix.

Figure 2.4 shows that for any encoder $\mathbf{C}$, progressively exist, a set of encoders algebraically equivalent to $\mathbf{C}$, which is a subset of the set of encoders that are strictly equivalent to $\mathbf{C}$, which in turn is a subset of the set of encoders that are range equivalent to $\mathbf{C}$.

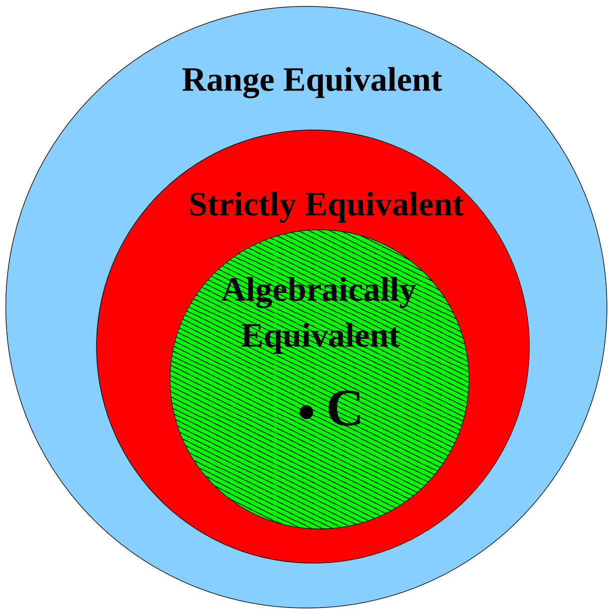


FIGURE 2.4: DIFFERENT TYPES OF ENCODER EQUIVALENCE

2.2.1 Search space for Algebraically Equivalent Encoders

Consider the state vector similarity transformation $\hat{s}_j = s_j\mathbf{S}$, where $\mathbf{S}$ is a nonsingular matrix. Under this transformation, the encoders described by the linear systems:

$$\begin{aligned} s_{j+1} &= s_j\mathbf{A} + u_j\mathbf{B} \qquad & s_{j+1} &= s_j\mathbf{S}\mathbf{A}\mathbf{S}^{-1} + u_j\mathbf{B}\mathbf{S}^{-1} \\ x_j &= s_j\mathbf{C} + u_j\mathbf{D} \qquad & x_j &= s_j\mathbf{S}\mathbf{C} + u_j\mathbf{D} \end{aligned} \tag{2.7}$$

where $\mathbf{S}$ is a nonsingular matrix, are strictly equivalent, since they have the same generator matrix:

$$G(D) = \mathbf{D} + \mathbf{B}\mathbf{S}^{-1}(D^{-1}\mathbf{I} - \mathbf{S}\mathbf{A}\mathbf{S}^{-1})^{-1}\mathbf{S}\mathbf{C} = \mathbf{D} + \mathbf{B}(D^{-1}I - \mathbf{A})^{-1}\mathbf{C} \tag{2.8}$$

where $\mathbf{I}$ is the identity matrix.

Moreover, an invertible transformation maps the zero state to the zero state, thus the encoders have the same mapping from input error events to output error events, and are also algebraically equivalent. The matrix $\mathbf{S}^{-1}\mathbf{A}\mathbf{S}$, with $\mathbf{S}$ nonsingular, is called *similar* to the matrix $\mathbf{A}$. For example, matrix $\mathbf{A_3}$ is similar to matrix $\mathbf{A_1}$:

$$\underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}}_{\mathbf{A_3}} = \underbrace{\begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{S}} \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{A_1}} \underbrace{\begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{S^{-1}}} \tag{2.9}$$

Encoder $\{\mathbf{A_1}, \mathbf{B_1}, \mathbf{C_1}, \mathbf{D_1}\}$ is algebraically equivalent to encoder $\{\mathbf{A_3}, \mathbf{B_3}, \mathbf{C_3}, \mathbf{D_3}\}$:

$$[s_1\ s_2\ s_3\ s_4]_{j+1} = [s_1\ s_2\ s_3\ s_4]_j \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}}_{\mathbf{A_3=SA_1S^{-1}}} + \quad [u_1\ u_2\ u_3]_j \underbrace{\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}}_{\mathbf{B_3=B_1S^{-1}}}$$

$$[y_1\ y_2\ y_3\ y_4]_j = [s_1\ s_2\ s_3\ s_4]_j \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{C_3=SC_1}} + \quad [u_1\ u_2\ u_3]_j \underbrace{\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}}_{\mathbf{D_3=D_1}} \tag{2.10}$$

In an exhaustive search for algebraically distinct codes, it is redundant to examine similar matrices $\mathbf{A}$. The range of matrices $\mathbf{A}$ to consider, that are not similar to each other, can be found from the rational form theorem presented in Horn and Johnson ([HJ85], pg. 154) and the references therein. The following definitions [HJ85] help to review the terminology employed by this theorem.

Definition 7 Annihilate *A polynomial whose value is the zero matrix at matrix* $\mathbf{A}$*, is said to annihilate* $\mathbf{A}$.

Definition 8 Minimal polynomial *The minimal polynomial of a matrix* $\mathbf{A}$ *is the unique monic polynomial of minimum degree that annihilates* $\mathbf{A}$*. The minimal polynomial has exactly the same roots as the characteristic polynomial but possibly with smaller algebraic multiplicity. Every monic polynomial is both the minimal and the characteristic polynomial of its companion matrix.*

Definition 9 Invariant factors *The invariant factors of a matrix* $\mathbf{A}$ *are the unique monic polynomials* $f_1(\lambda), f_2(\lambda) \ldots f_k(\lambda)$ *with the following properties:*

1. $f_1(\lambda)$ *is the minimal polynomial of* $\mathbf{A}$.

2. *The product of all the invariant factors is the characteristic polynomial of* $\mathbf{A}$*.*

3. *Their degrees are monotone non-increasing*

4. $f_{k+1}(\lambda)$ *divides* $f_k(\lambda)\dots f_1(\lambda)$

The invariant factors can be determined in a definite way for a matrix $\mathbf{A}$. Two matrices are similar if and only if their invariant factors are identical. To each invariant factor can be associated a companion matrix.

Theorem 1 Rational Form Theorem *Any matrix over a field* $\mathcal{F}$ *is similar over* $\mathcal{F}$ *to the direct sum of the companion matrices of its invariant factors. These invariant factors are uniquely determined polynomials with coefficients from* $\mathcal{F}$*.*

This theorem tells us that in order to consider all the possible different matrices $\mathbf{A}$ of dimension $m \times m$, consider all the polynomials of degree m, their subsequent valid factorizations into invariant factors, and finally the direct sum of the associated companion matrices.

The direct sum of R companion matrices has exactly the same form as the matrix $\mathbf{A}$ of a canonical form encoder with R rows of memory elements. The feedback in each row is determined by the corresponding invariant factor. If the corresponding invariant factor is of the form D^{m_i} there is no feedback.

In other words the rational form theorem states that for any convolutional encoder with m memory elements, no matter how these memory elements are connected, there exists an algebraically equivalent encoder with the memory elements connected in R rows for some R. Thus in an exhaustive search over algebraically distinct encoders, it is sufficient to consider the canonical memory structures with R rows, provided all possible ways of input and output connection, which are not restricted to be of any specific form, are taken into account.

2.3 Minimality among range-equivalent encoders

Definition 10 Range minimality *Among all range-equivalent encoders (definition 1), the encoder that uses the smallest number of memory elements is called minimal.*

Theorem 2 *A feedforward encoder is minimal if and only if:*

- *It is delay preserving.*
- *It is degree preserving.*
- *It is non catastrophic.*

This particular discussion of minimality is from [Wes00]. A similar analysis was initialy presented in [For70].

Definition 11 Delay Preserving *A feedforward encoder is called delay preserving if and only if $delay(u(D)G(D)) = delay(u(D))$ where $delay(u(D))$ of the $1 \times k$ vector $u(D)$ is the minimum power of D in the vector's polynomial components.*

Definition 12 Degree Preserving *A feedforward encoder is called degree preserving if and only if*

$$degree(u(D)G(D)) = \max_{1 \le i \le k}(degree(u_i(D)) + m_i)$$

where $degree(u(D)$ of a vector $u(D)$ is the maximum power of D in the polynomial vector's components, $m_i = max_j(degreeg_{ij}(D))$, and $g_{ij}(D)$ are the polynomial elements of the generator matrix $G[D]$.

Definition 13 Catastrophic *An encoder, either feedforward or feedback, is called catastrophic, if and only if an infinite input weight sequence is mapped to a finite output weight sequence.*

Although Theorem 2 applies only to feedforward encoders, it has been extended to include feedback encoders in [JW93, FJW96, LFM94].

We are going to present an algebraic criterion to determine whether a given encoder $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ is minimal or not. In this scope we apply well known control-theory results, and examine what the minimality conditions imply for the state diagram of a convolutional code.

2.4 Control theory applied to convolutional encoders

In control theory [Che99], the discrete-time state-space equations are examined through the notions of controllability, observability and minimality. Controllability is a property that applies to both the state vector, and the output vector.

2.4.1 Controllability of state vector

State vector controllability implies that the encoder should have an irreducible state space diagram, have all states accessible.

Definition 14 *The state vector s_j is said to be controllable, if there exists an input sequence, that can drive s_j from any initial state s_0, to any final state s_1, in a finite number of steps.*

Algebraic test

This criterion involves only the $\mathbf{A}$ and $\mathbf{B}$ matrices in the equation for s_j:

$$s_{j+1} = s_j\mathbf{A} + u_j\mathbf{B} \tag{2.11}$$

Because of linearity, it is sufficient to be able to go from the zero state, to any state, in a finite number of steps. Thus:

$$\left.\begin{array}{l} s_0 = 0 \\ s_1 = u_1\mathbf{B} \\ s_2 = u_1\mathbf{BA} + u_2\mathbf{B} \\ \ldots \\ s_m = u_1\mathbf{BA}^{m-1} + u_2\mathbf{BA}^{m-2} + \ldots + u_m\mathbf{B} \end{array}\right\} \Rightarrow s_m = [u_1u_2\ldots u_m]\begin{bmatrix} \mathbf{BA}^{m-1} \\ \mathbf{BA}^{m-2} \\ \vdots \\ \mathbf{B} \end{bmatrix}$$

It is enough to examine up to m steps, since from the Cayley-Hamilton theorem, $\mathbf{A}^m$ can be expressed as a function of lower powers of $\mathbf{A}$. The matrix:

$$\begin{bmatrix} \mathbf{BA}^{m-1} \\ \mathbf{BA}^{m-2} \\ \vdots \\ \mathbf{B} \end{bmatrix}$$

of dimension $km \times m$ is called controllability matrix. The state vector s_j is controllable if and only if the controllability matrix is full rank.

2.4.2 Controllability of output vector

Output vector controllability simply states that all the outputs should be used in the state space diagram.

Definition 15 *The output vector y_j is said to be controllable, if given the current output vector y_0 there exists an input sequence that after a finite number of steps can produce any output vector y_1.*

Algebraic test

Using the same steps as before, the output after $m+1$ steps is given by:

$$y_m = [u_1u_2\ldots u_mu_{m+1}]\begin{bmatrix} \mathbf{BA}^{m-1}\mathbf{C} \\ \mathbf{BA}^{m-2}\mathbf{C} \\ \vdots \\ \mathbf{BC} \\ \mathbf{D} \end{bmatrix} \tag{2.12}$$

The output vector y_j is controllable if and only if the above matrix of dimensions $k(m+1) \times n$ is full rank.

2.4.3 Observability

Definition 16 *A system is said to be observable, if the knowledge of both the input and output sequences for a finite number of steps, is sufficient to determine the initial state s_0 and thus the corresponding state sequence.*

If the state diagram of a system $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ has two loops with the same input and the same output, the system is unobservable. But because of linearity, this implies that the state diagram also has a loop with zero input and zero output. Thus the question of observability can be resolved by looking at the system output when the input is zero.

Algebraic Test

The system is observable if and only if the matrix:

$$[\ \mathbf{CAC} \ldots \mathbf{A}^{m-1}\mathbf{C}\ ] \tag{2.13}$$

of dimensions $m \times mn$ is full rank.

2.4.4 Minimality in Control

Theorem 3 *A realization $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ is called minimal:*

- *if and only if it has the smallest number of state variables among all realizations with the same transfer function.*
- *if and only if the state state vector s is controllable and observable.*

The first condition implies that minimality in control theory corresponds to minimality among strictly equivalent encoders, and not among range equivalent encoders. The second condition leads to the following observation: minimality among strictly equivalent encoders implies that it is not necessary to have two loops with the same input and the same output in the state diagram. One such loop is sufficient.

By the same token minimality among range equivalent encoders implies that it is not necessary to have two loops with the same output in the state diagram, one such loop is enough. To describe this condition, we use the term of output observability.

Definition 17 Output Observability *The linear system* $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ *is said to be output observable, if the state sequence* $\{s_0, s_1, \ldots, s_N\}$ *is uniquely determined by the knowledge of the output sequence* $\{y_0, y_1, \ldots, y_N\}$ *for a finite number of steps* N*.*

Algebraic test

A system is output observable if and only if the following conditions hold:

1. The matrix $\mathbf{D}$ has full rank (condition for delay preserving),
2. The matrix
$$\begin{bmatrix} \mathbf{C} & \mathbf{AC} & \mathbf{A}^2\mathbf{C} & \ldots & \mathbf{A}^{m-1}\mathbf{C} \\ \mathbf{D} & \mathbf{BC} & \mathbf{BAC} & \ldots & \mathbf{A}^{m-2}\mathbf{C} \\ 0 & \mathbf{D} & \mathbf{BC} & \ldots & \mathbf{A}^{m-3}\mathbf{C} \\ & & & \ddots & \\ 0 & 0 & 0 & 0 & \mathbf{D} \end{bmatrix}$$
has full rank.

An encoder is minimal if and only if it is output observable, as is stated in the following theorem, presented in [LFM94]:

Theorem 4 *An encoder is minimal if and only if any of the following equivalent conditions hold:*

1. *There exists an integer N, such that the mapping of output sequences $\{y_0, y_1, \ldots, y_N\}$ to state sequences $\{s_0, s_1, \ldots, s_N\}$ is one to one, where $N \leq m$.*
2. *No state other than the zero state is the starting or ending state of a semi-infinite zero-label path.*

Theorem 4 is equivalent to saying that an output observable encoder is delay preserving, degree preserving, and non catastrophic (Theorem 2). This is easy to see by interpreting these properties for the state diagram of a feedforward convolutional encoder.

1. *Not Delay Preserving*

   The state diagram contains a branch exiting the zero state with output zero

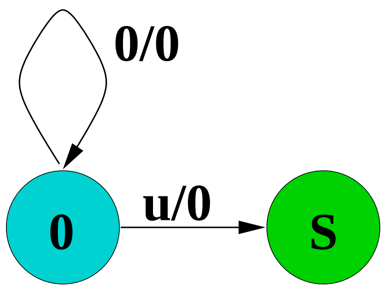


FIGURE 2.5: STATE DIAGRAM OF A NOT DELAY PRESERVING ENCODER

   towards a state S, as depicted in Fig. 2.5. There are two ways to produce a semi-infinite all zero output sequence, by either ending it at state S, or at the zero state. Thus such an encoder is not output observable. Also this encoder does not satisfy the minimality principle, because the existence of this branch is redundant.

2. *Not Degree Preserving*

   The state diagram contains a branch entering the zero state with output zero emanating from a state S, as depicted in Fig. 2.6, different than the

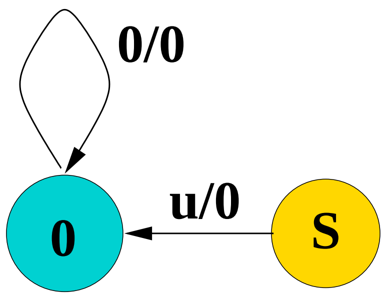


Figure 2.6: State diagram of a not degree preserving encoder

self loop. There are two ways to produce a semi-infinite all zero output sequence, by either starting at state S or at the zero state. Thus such an encoder is not output observable either, and this branch is not necessary.

3. *Catastrophic*

The state diagram contains a loop with output all zero, different than the

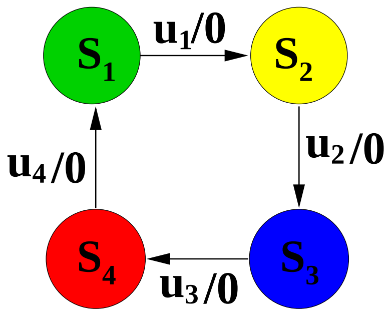


Figure 2.7: State diagram of a catastrophic encoder

self loop, as depicted in Fig. 2.7. An all zero semi-infinite sequence can start from any state of this loop. Thus such an encoder is not output observable. The zero-output loop does not contribute anything new to the possible output sequences, it only reproduces the outputs of the self loop. Thus this encoder cannot be minimal.

Note that for feedforward codes, no loop other than the self-loop around the zero state can have all-zero input. Thus a zero-output loop that does not include the zero state always implies a catastrophic feedforward encoder.

2.5 Minimality under periodic puncturing

Trellis codes can be designed to offer reliable performance over periodic erasure channels. Such channels arise for example from partial-band interference in frequency-hopped or multicarrier transmission, that is dispersed by a block interleaver.

Lapidoth [Lap94] analysed convolutional codes under periodic erasures. Wesel, Liu and Shi [WLS00] describe techniques for the analysis and design of trellis codes, optimized for Euclidean weight under periodic erasures. Founded on their work, this section further investigates codes optimized for puncturing, and elaborates on the minimality of the punctured codes.

When periodically punctured, a minimal encoder produces a higher rate encoder that may or may not be minimal. If it is not minimal, it is not output observable [LFM94, FJW96, FW99] and possibly catastrophic. A code search can use a fast algorithm to determine minimality under puncturing, and a proposed method to assess the performance of non-minimal under puncturing codes. As an example, the paper optimizes rate 1/4 unpunctured codes for Hamming weight under both bit-wise and symbol-wise periodic puncturing. Code tables and simulation results are included.

2.5.1 Periodic symbol puncturing

Periodic symbol puncturing [WLS97] with period p can be described by a p-element vector $\tilde{\mathbf{a}} = [a_1 \ldots a_p]$, $a_i \in \{0, 1\}$, applied to the output of a convolutional encoder:

$$\begin{aligned} s_{j+1} &= s_j\mathbf{A} + u_j\mathbf{B} \\ x_j &= s_j\mathbf{C} + u_j\mathbf{D} \\ y_j &= a_{(j\%p)}x_j \end{aligned} \tag{2.14}$$

where "%" denotes the modulo operation, s_j is the state vector of dimension $1\times m$, u_j is the input vector of dimension $1 \times k$, x_j is the output of the unpunctured convolutional code of dimension $1 \times n$, and y_j is the output after puncturing. A code under periodic symbol puncturing is periodically time-variant, in the sense that the output corresponding to a specific encoder state and input depends on the phase i of the puncturing pattern (on whether $a_i = 0$ or $a_i = 1$).

A $p - q$ erasure pattern has q zero values $a_i = 0$ (erased), and $p - q$ nonzero values $a_i = 1$ (unerased). Trellis codes $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ can offer consistent performance (according to the criteria in [WLS00]) over the set of all $p - q$ erasure patterns of interest. This set is constrained by the number of memory elements m, number of inputs k and number of outputs n of the code, as is determined in [WLS00]. For example every a code fails when punctured to rate greater than one (negative redundancy).

Section 2.3 defined as minimal the encoder that uses the smallest number of memory elements among all its range equivalent encoders. Theorem 4 established that a minimal encoder is output observable (definition 17), that is, the knowledge of the output sequence for a finite number of steps, is sufficient to determine the initial state s_0 and thus the corresponding state sequence.

Assume that the encoder $\{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}\}$ is minimal. Puncturing creates a higher rate encoder, that may or may not be output observable, depending on whether puncturing has left enough structure to the output sequences to determine the corresponding state sequences. If the punctured encoder is not output observable, then two different semi-infinite state sequences are mapped to the same output sequence. From linearity this implies that the code may have a zero-output loop different than the self-loop around the zero state. If the input to such a zero output loop is non-zero, the encoder is catastrophic. A catastrophic code maps an infinite input weight sequence to a finite output weight sequence, which causes the code to fail.

During a search for codes under periodic puncturing, care must be taken to identify codes that become non-minimal under puncturing. In Section 2.5.2 we discuss the appropriate search space for the code search. Section 2.5.3 introduces a fast algorithm to check whether an encoder has a zero output loop under a specific puncturing pattern. Section 2.5.4 proposes a method to assess the performance of the encoders containing such a loop. Section 2.5.5 investigates the performance of our example 1/4 codes under both bit-wise and symbol-wise puncturing, and provides code tables as well as simulation results.

2.5.2 Search space

An exhaustive search maximizing free distance over all minimal encoders of a given rate and number of memory elements yields codes with better distance properties than any non-minimal encoder with the same complexity can achieve. To optimize the performance of a code under a single specific periodic puncturing pattern, an exhaustive search can safely exclude any non-minimal codes under this pattern.

However, optimizing the performance of a single code under a family of periodic puncturing patterns is a multi-criterion optimization problem [WLS00], that does not necessarily have a unique solution. Often, no single code gives the best possible performance for all puncturing patterns. The search seeks to identify the subset of codes that offer reasonable performance over the whole family of puncturing patterns. This subset may include codes that become non-minimal under one or more of the puncturing patterns Such a code may even be the best choice when considering its performance for all puncturing patterns. As a result, an exhaustive search considering several puncturing patterns, cannot safely exclude codes that are do not retain minimality under all puncturing patterns. For example, when punctured to uncoded (rate 1, no redundancy) an encoder is always non-minimal, since the minimal encoder in this case contains no memory elements. Still such a pattern might be of interest in a partial-band jamming application.

The search space for trellis codes is the set of encoders that are not range equivalent to each other. Two encoders are called range equivalent [Wes96] if they have the same set of output sequences. This is Forney's notion of equivalence [For70]. For non-minimal encoders that have a zero output loop, the input sequence mapped to the zero output loop determines whether the encoder is catastrophic or not, which has a dramatic effect on performance. It is quite often the case that there are two range-equivalent encoders, and one is catastrophic while the other is not.

Thus we propose to use an exhaustive set of encoders that are strictly distinct as the search space. Two encoders are called *strictly equivalent* [Wes96] if they map the same input sequence to the same output sequence. If two encoders are not strictly equivalent we say they are strictly distinct. The rational form theorem in [FW00] provides a method for identifying an exhaustive set of strictly distinct encoders. A group theoretic approach in [BGM98] provides a different way of identifying such a set.

2.5.3 Algorithm to determine zero-output loops

To determine whether puncturing causes a code state diagram to have a zero-output loop, one may start from all states, and all the the distinct phases of the puncturing pattern (for example the $p - q = 4 - 2$ puncturing pattern [0 1 0 1] has two distinct phases: [0 1 0 1] and [1 0 1 0]) and check if this state belongs to a zero output loop.

The loop which after puncturing has only zero outputs, before puncturing, must have at least one zero output. Otherwise, since at least one a_i is nonzero, the nonzero output eventually adds distance to the loop. Moreover, the zero output is aligned with a nonzero a_i.

This observation leads to the proposed algorithm: start only from the states with a zero output, and align this zero output with a phase of the puncturing pattern such that $a_1 = 1$, i.e. the first symbol of the period is not punctured. For example, for a code with six memory elements under the $p - q = 4 - 2$ puncturing pattern [0 1 0 1], instead of starting from the 63 nonzero states under both phases, it is sufficient to start from the three states that have a zero output coupled with phase [1 0 1 0], which amounts to a reduction by 96.83% of the search space.

This observation can be applied to the adaptation of Lapidoth's algorithm [Lap94] presented in [WLS00], as follows. Start only from states j that have a zero output ($S_0 = S_0^j$) and use a phase of the puncturing pattern that starts with an unpunctured erasure coefficient.

2.5.4 Performance of encoders with a zero-output loop

The performance of a code over an AWGN channel is determined by its free distance. For codes optimized for periodic puncturing, the corresponding metric is residual distance. Residual distance [WLS00] indicates how much output distance is provided by the code after periodic attenuation. In other words, residual distance is the the free distance of the punctured code.

Consider a non-minimal code with a zero-input zero-output loop, decoded with the standard Viterbi algorithm. The free distance is not necessarily equal to the minimum output distance associated with the encoder's error events. As error events we refer to the trellis paths of finite length that leave the zero state once and return to it only once ([BDM91] pg. 61).

However, the free distance is equal to that of a strictly equivalent minimal encoder, and for minimal encoders free distance is also the minimum distance associated with the encoder's error events. This minimal strictly equivalent encoder may also be used to compute transfer function bounds. A reduced complexity transfer function bound, and a method for calculating the transfer function bound for codes under periodic erasures, are described in [WLS00, Wes99, WL98].

As an example, consider the non-minimal encoder $\{\mathbf{A_1}, \mathbf{B_1}, \mathbf{C_1}, \mathbf{D_1}\}$ (2.2) that contains a zero-input zero-output loop. This encoder is strictly equivalent to the minimal encoder $\{\mathbf{A_2}, \mathbf{B_2}, \mathbf{C_2}, \mathbf{D_2}\}$ in (2.2.1). The encoder $\{\mathbf{A_2}, \mathbf{B_2}, \mathbf{C_2}, \mathbf{D_2}\}$ may be used to calculate the free distance and the transfer function bounds for the encoder $\{\mathbf{A_1}, \mathbf{B_1}, \mathbf{C_1}, \mathbf{D_1}\}$.

The following theorem allows the determination the free distance of a code containing a zero-output loop without converting it to a minimal equivalent encoder:

Theorem 5 *The free distance of a (non-minimal) code containing a zero output loop, is equal to the minimum output distance of all paths that start and end in the zero state, and all paths that start in the zero state and end in the zero-output loop.*

Proof

The non-minimal encoder $\{\mathbf{A},\ \mathbf{B},\ \mathbf{C},\ \mathbf{D}\}$ is not output observable. Thus there exists a state vector equivalence transformation $\hat{\mathbf{s}}_j = \mathbf{P}\mathbf{s}_j$, where $\mathbf{P}$ is a nonsingular matrix determined by the observability matrix $O = [\mathbf{C}\ \mathbf{CA}\ \ldots\ \mathbf{CA}^{m-1}]$ [Che99], that leads to an algebraically equivalent encoder of the form:

$$
\begin{aligned}
[\mathbf{s_o}\ \mathbf{s_{no}}]_{j+1} &= [\mathbf{s_o}\ \mathbf{s_{no}}]_j \underbrace{\begin{bmatrix} \mathbf{A}_o & 0 \\ \mathbf{A_{no1}} & \mathbf{A_{no2}} \end{bmatrix}}_{\mathbf{A}} + u_j \underbrace{\begin{bmatrix} \mathbf{B}_o \\ \mathbf{B}_{no} \end{bmatrix}}_{\mathbf{B}} \\
y_j &= [\mathbf{s_o}\ \mathbf{s_{no}}]_j \underbrace{\begin{bmatrix} \mathbf{C}_o \\ \mathbf{C}_{no} \end{bmatrix}}_{\mathbf{C}} + u_j\mathbf{D}
\end{aligned} \tag{2.15}
$$

where the m-dimensional state vector $\mathbf{s}$ is divided into an an observable part $\mathbf{s_o}$ and and a not observable part $\mathbf{s_{no}}$. By dropping the unobservable state vector $\mathbf{s_{no}}$ we obtain an observable state equation of lesser dimension, that corresponds to a strictly equivalent observable (minimal) encoder.

Assuming $\mathbf{C}$ is full rank, the zero input - zero output loop is described by the state equations:

$$
\mathbf{s_o} = \mathbf{0},\ \ \mathbf{s_{no}} = \mathbf{A_{no2}}\mathbf{s_{no}} \tag{2.16}
$$

where $\mathbf{0}$ is the zero vector. That is, for the strictly equivalent observable encoder, the zero-input zero-output loop is mapped to the self-loop around the zero state. The error events that start and end in the zero state $\mathbf{s_o} = \mathbf{0}$ for the minimal encoder, may start or end in the zero loop for the non-minimal encoder. $\square$

2.5.5 Code search and Simulation results

In this section we present code tables and simulation results for rate 1/4 codes employing BPSK with $m = 6$ memory elements, optimized for periodic puncturing with period $p = 4$. To identify good codes we examined both symbol-wise and bit-wise puncturing.

Symbol-wise puncturing is described by (2.1). For bit-wise puncturing with period equal to the number of outputs ($p = n$) (2.5.1) is replaced by (2.17):

$$\mathbf{y}_j = \mathbf{x}_j \odot \tilde{\mathbf{a}} \tag{2.17}$$

where $\odot$ stands for the element by element multiplication of vectors $\mathbf{x}_j$ and $\tilde{\mathbf{a}}$. Bit puncturing with a $p - q$ puncturing pattern amounts to to ignoring q out of p outputs of the initial code, and thus leads to a time-invariant code of a higher rate.

To describe an encoder we give in octal notation the feedback polynomial $f(D)$, the $k = 1$ row $\mathbf{b}_1$ of matrix $\mathbf{B}$, the $n = 4$ columns $\{\mathbf{c}_1 \ldots \mathbf{c}_4\}$ of matrix $\mathbf{C}$, and the $k = 1$ row $\mathbf{d}_1$ of matrix $\mathbf{D}$. For example, code C_9 in Table 2.4, described by the polynomials {0140, 040, 054, 062, 052, 013, 07} has the $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ state-space description:

$$\mathbf{s}_{j+1} = \mathbf{s}_j \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{A}} + \mathbf{u}_j \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{B}}$$

$$\mathbf{y}_j = \mathbf{s}_j \underbrace{\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{C}} + \mathbf{u}_j \underbrace{\begin{bmatrix} 0 & 1 & 1 & 1 \end{bmatrix}}_{\mathbf{D}}$$

2.5.6 Bit-wise Puncturing

For bit-wise puncturing an exhaustive search examined all strictly distinct rate 1/4 feedback encoders such that, when punctured to uncoded, all resulting 1/1 encoders are non-catastrophic. The search was restricted to feedback encoders, since all rate 1/1 feedforward encoders are catastrophic, (with the exception of the 1/1 codes with generator polynomials $[D^k]$, $k = 1 \dots m$).

Let d^q_{free} denote the *minimum* residual Hamming distance for each set of $4-q$ punctured codes, $q = 0 \dots 3$, i.e., for each set of codes with the same rate. Each 1/4 code, leads to a set of four 1/3 codes under $4-1$ puncturing, six 1/2 codes under $4-2$ puncturing and four 1/1 codes under $4-3$ puncturing. The free distance of the unpunctured code is equal to d^0_{free}. For $4-3$ puncturing the output Hamming distance of non-catastrophic codes is always equal to one ($d^3_{free} = 1$).

Table 2.1 presents three codes, each achieving the highest residual Hamming d^q_{free} for one value of q, $q = 0, 1, 2$, as Table 2.2 shows.

Code	$\{f, \mathbf{b}_1, \mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4, \mathbf{d}_1$
B_1	$\{0101, 040, 010, 052, 066, 077, 01\}$
B_2	$\{0123, 040, 044, 064, 011, 027, 02\}$
B_3	$\{0123, 040, 050, 066, 076, 077, 10\}$

Table 2.1: Codes with $m = 4$ memory elements, $k = 1$ input, and $n = 4$ outputs, optimized for Hamming distance under bit-wise periodic puncturing, with period p=4

Code	d^2_{free}	d^1_{free}	d^0_{free}
B_1	3	7	14
B_2	2	8	11
B_3	4	8	12

Table 2.2: Distance characteristics of bit-wise punctured codes

Table 2.3 provides in detail the distance characteristics of the codes presented in Tables 2.1 and 2.2 . In the second row, the d_* subscript denotes the unpunctured output bits. For example, d_{12} stands for the residual distance when the two MSB bit outputs are not punctured and the two LSB outputs are punctured, d_{234} stands for the residual distance when the output MSB bit is punctured, and d_{1234} stands for the free distance when no output bit is punctured.

The number given in parentheses is N_b. For no puncturing N_b would be the total number of input bits of all error events that have output distance equal to the free distance. For codes under periodic puncturing, N_b is the natural extension of this idea, as is described in [WLS00]. The notation "(zl)" indicates that the encoder has a zero-input zero-output loop, and the distance from the zero state to this loop determines the residual distance.

Code	d^2_{free}					
	d_{12}	d_{13}	d_{14}	d_{23}	d_{24}	d_{34}
B_1	4(7)	5(9)	4(4)	5(zl)	3(zl)	6(zl)
B_2	6(30)	2(zl)	5 (4)	6(30)	3(zl)	5(4)
B_3	5(6)	4(6)	4(zl)	4(zl)	4(zl)	5(4)

Code	d^1_{free}				d^0_{free}
	d_{123}	d_{124}	d_{134}	d_{234}	d_{1234}
B_1	8(2)	8(4)	8(4)	7(zl)	14(12)
B_2	8(4)	8(4)	8(4)	8(4)	11(4)
B_3	8(6)	10(11)	8(11)	9(4)	12(11))

Table 2.3: Analytic distance properties of codes in Table 2.3

2.5.7 Symbol-wise puncturing

For symbol-wise puncturing, we performed a partial search over a set of strictly distinct feedback encoders. Feedback encoders were chosen as more resilient to catastrophicity. As noted in [WLS00] a catastrophic feedforward code may lead to a non-catastrophic feedback code, while a catastrophic feedback code always leads to a catastrophic range equivalent feedforward code. The search was partial in that it did not exhaust the set of strictly distinct encoders due to computational limitations, but it included a full set of range distinct encoders.

Ignoring cyclic shifts, puncturing with period four leads to one $4-3$ pattern [0 0 0 1] with associated free distance denoted by d^{01}_{free}, two $4-2$ puncturing patterns [0 0 1 1] and [0 1 0 1] with associated free distance denoted by d^{03}_{free} and d^{05}_{free} respectively, and one $4-1$ puncturing pattern [0 1 1 1] with associated free distance denoted by d^{07}_{free}. The free distance of the unpunctured code, can be viewed as corresponding to the puncturing pattern [1 1 1 1], and is denoted by d^{17}_{free}. Puncturing to rate one (uncoded), for non-catastrophic codes, always implies distance $d^{01}_{free} = 1$.

Table 2.4 presents codes that are non-catastrophic when punctured to uncoded, and offer consistent performance under the different puncturing patterns. The metrics presented in the third and fourth column are calculated [WLS00] as:

$$J_{dB} = \sum_j 10 \log_{10}(4(d^j_{dfree})^2) \tag{2.18}$$

where $4(d^j_{dfree})^2$ is the squared Euclidean distance corresponding to Hamming distance d_j and BPSK constellation normalized to unit energy ($E_s = 1$). The summation is over all puncturing patterns $\mathbf{a}_j$.

$$J_{MI} = \sum_j \frac{p - q_j}{p} \log_2(4(d^j_{free})^2) \tag{2.19}$$

where q_j is the number of punctured symbols for the puncturing patterns $\mathbf{a}_j$. Table 2.5 gives the residual distance characteristics of these codes. L_D is the natural extension of traceback depth for codes under periodic erasures. Both N_b and L_D are described in detail in [WLS00].

The codes in Table 2.5 achieve much higher residual Hamming distance than the codes in Table 2.3: for $4-0$ puncturing (unpunctured) they achieve distance

C	$\{f,\mathbf{b}_1,\mathbf{c}_1,\mathbf{c}_2,\mathbf{c}_3,\mathbf{c}_4, \mathbf{d}_1\}$	J_{MI}	J_{dB}
C_1	$\{170, 40, 66, 56, 13, 33, 17\}$	70.3	15.21
C_2	$\{140, 40, 42, 16, 11, 25, 17\}$	70.3	15.21
C_3	$\{140, 40, 52, 06, 65, 15, 74\}$	70.3	15.21
C_4	$\{113, 40, 24, 74, 22, 65, 64\}$	70.8	15.25
C_5	$\{151, 40, 40, 60, 76, 07, 13\}$	70.8	15.25
C_6	$\{124, 60, 64, 22, 06, 43, 16\}$	70.8	15.25
C_7	$\{136, 60, 40, 62, 01, 27, 7\}$	70.8	15.25
C_8	$\{160, 40, 74, 16, 41, 71, 15\}$	70.8	15.25
C_9	$\{140, 40, 54, 62, 52, 13, 7\}$	70.9	15.26
C_{10}	$\{140, 40, 12, 56, 35, 63, 17\}$	70.6	15.24
C_{11}	$\{120, 40, 64, 06, 43, 67, 16\}$	70.5	15.23
C_{12}	$\{140, 40, 26, 45, 65, 13, 17\}$	70.5	15.23
C_{13}	$\{174, 40, 15, 55, 07, 47, 16\}$	70.5	15.23
C_{14}	$\{110, 40, 45, 65, 63, 07, 15\}$	69.5	15.09
C_{15}	$\{146, 40, 24, 42, 62, 11, 17\}$	69.5	15.09
C_{16}	$\{150, 60, 25, 65, 37, 77, 15\}$	69.5	15.09

TABLE 2.4: CODES WITH $m = 4$ MEMORY ELEMENTS, $k = 1$ INPUT, AND $n = 4$ OUTPUTS, OPTIMIZED FOR HAMMING DISTANCE UNDER SYMBOL-WISE PERIODIC PUNCTURING, WITH PERIOD P=4

up to 20 as opposed to 14, for $4 - 1$ puncturing they achieve distance up to 14 as opposed to 8, and for $4 - 2$ puncturing they achieve distance up to 8 as opposed to 4. The reason is that the set of non-catastrophic codes under bit-wise puncturing is much smaller than the set of non-catastrophic codes under symbol-wise puncturing. The limited number of non-catastrophic codes incurs much smaller achievable distance for the rest of the puncturing patterns. An intuitive explanation may be the following. The rate 1/4 codes of our example transmit at each trellis step four BPSK symbols. Symbol and bit-puncturing of period four can be viewed as special cases of puncturing BPSK symbols (output bits) with period sixteen. The $4-2$ puncturing pattern [1 1 0 0], applied bit-wise is equivalent to the bit erasure pattern: $\mathbf{a}_{16}^1 = [1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0]$. When applied symbol-wise, it corresponds to the bit puncturing pattern: $\mathbf{a}_{16}^2 = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$. Bit-wise puncturing with period four is more dispersed than symbol-wise puncturing with period four, which may lead to codes more prone to catastrophicity.

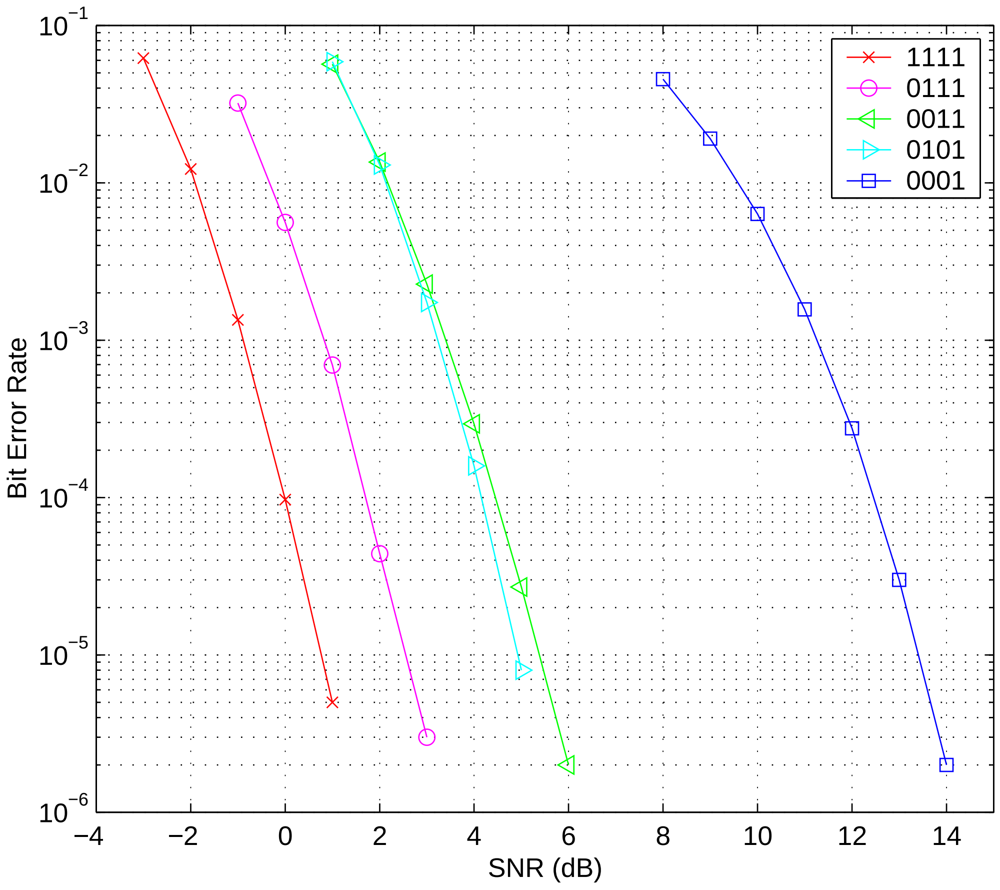


FIGURE 2.8: PERFORMANCE OF CODE C_2 UNDER PERIODIC SYMBOL PUNCTURING WITH PERIOD $p = 4$

Table 2.6 shows the Bit Error Rate performance at two SNR points for each puncturing pattern, for the codes in Table 2.4. The codes are distinguished by small trade-offs.

Figures 2.8, 2.9 and 2.10 plot the performance of codes C_2, C_3 and C_9 respectively, in BER vs. SNR (in dB), for all possible periodic puncturing patterns. Code C_3 offers better performance when unpunctured (pattern 1111), and code

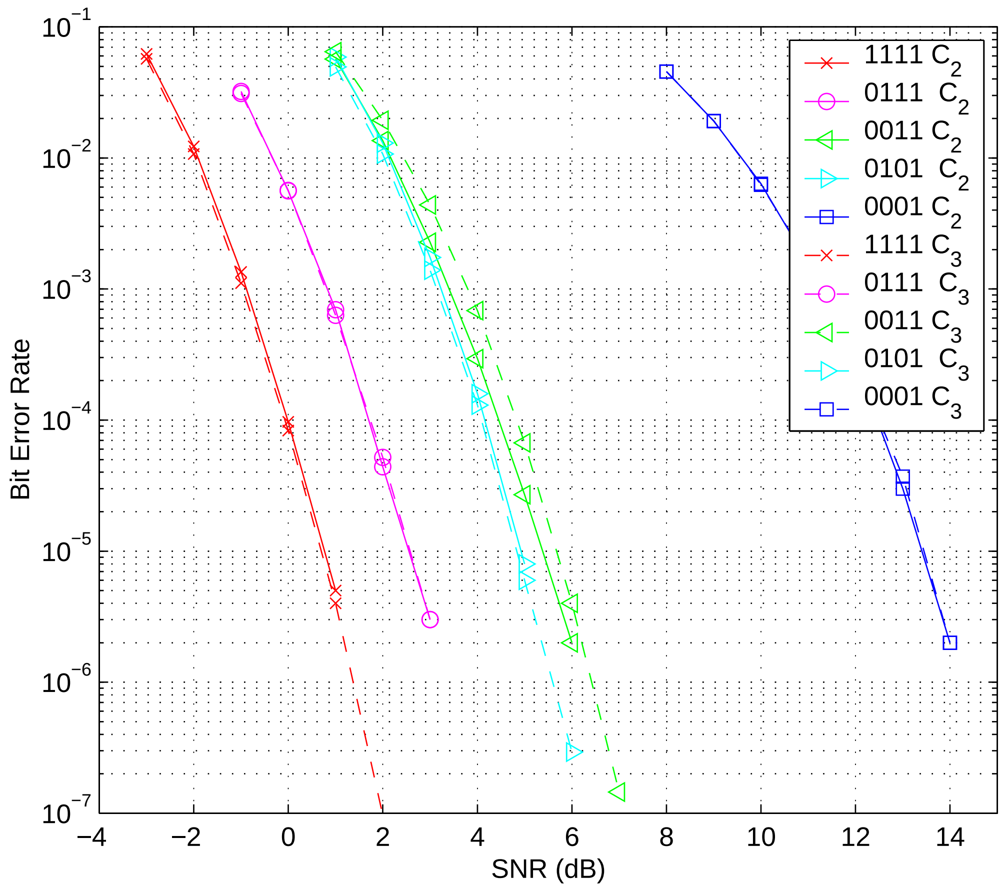


FIGURE 2.9: PERFORMANCE OF CODE C_2 UNDER PERIODIC SYMBOL PUNCTURING WITH PERIOD $p = 4$

C_9 better performance for the $4-2$ puncturing patterns (0101 and 0011). Code C_2 very similar performance to C_3. The Viterbi simulation used traceback depth $L_D = 40$. Fig. 2.11 compares the the performance of codes C_2 and C_9, while Fig. 2.12 compares the the performance of codes C_2 and C_3.

Figures 2.13, 2.14 and 2.15 plot BER vs. mutual information (in bits per channel use) for codes C_2, C_3 and C_9. Mutual information is calculated, for a

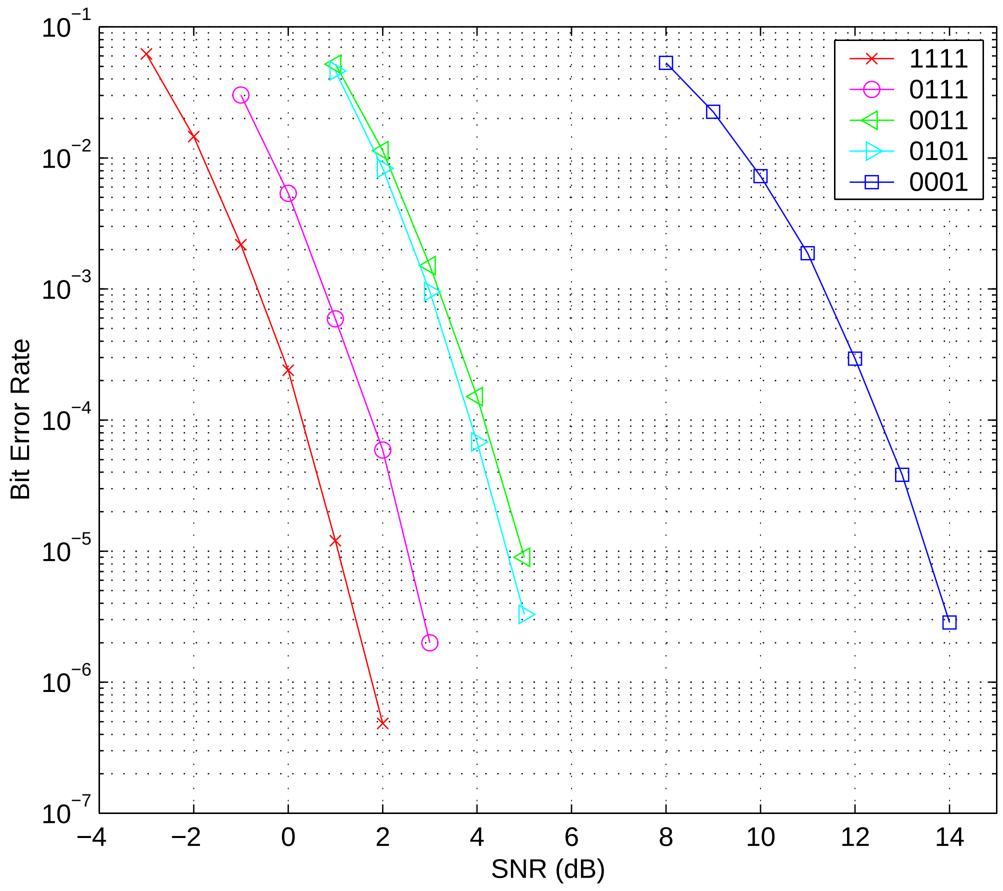


FIGURE 2.10: PERFORMANCE OF CODE C_9 UNDER PERIODIC SYMBOL PUNCTURING WITH PERIOD $p = 4$

$p - q$ puncturing pattern, as [WLS00]:

$$\frac{p-q}{p}\log_2(1+\mathrm{SNR}), \quad q = 0\ldots 3. \tag{2.20}$$

This plot allows one to examine proximity to capacity for each puncturing pattern. For example, code C_2 at BER=10^{-5} requires a consistent excess mutual information between $0.77 - 0.885$ for all five erasure patterns. This amount of excess mutual information is very similar to the one required in [WLS00] for rate 1/3 codes.

Code	1111			0111			0011			0101		
	d_{free}	N_b	L_D	d_{free}^{07}	N_b	L_D	d_{free}^{03}	N_b	L_D	d_{free}^{05}	N_b	L_D
C_1	20	71	11	11	4	9	6	12	10	8	59	15
C_2	20	40	10	11	2	9	6	10	10	8	101	15
C_3	20	24	10	11	2	9	6	28	17	8	99	15
C_4	19	38	9	11	6	9	7	32	11	8	32	12
C_5	19	38	9	11	6	9	7	33	11	8	35	12
C_6	19	32	9	11	2	9	7	49	14	8	68	15
C_7	19	40	9	11	4	9	7	41	14	8	63	15
C_8	19	24	9	11	4	9	7	44	14	8	64	15
C_9	18	8	8	12	2	9	7	18	11	8	28	13
C_{10}	18	8	7	13	20	13	6	8	10	8	71	17
C_{11}	19	40	10	12	10	10	6	4	10	8	55	17
C_{12}	19	24	11	12	14	13	6	8	13	8	48	15
C_{13}	19	40	11	12	20	13	6	9	13	8	57	15
C_{14}	19	24	10	13	32	13	5	2	10	7	12	13
C_{15}	19	16	7	13	42	13	5	6	10	7	8	13
C_{16}	19	24	10	13	29	13	5	4	10	7	12	13

TABLE 2.5: RESIDUAL DISTANCE FOR CODES IN TABLE 2.4 UNDER PERIOD-FOUR ERASURE PATTERNS

$p=4$	1111		0111		0011		0101	
dB	-1.0	0.5	0.5	2.0	2.75	4.25	2.75	4.25
Code	$\times10^{-3}$	$\times10^{-5}$	$\times10^{-3}$	$\times10^{-5}$	$\times10^{-3}$	$\times10^{-5}$	$\times10^{-3}$	$\times10^{-5}$
C_1	1.68	3.4	2.09	6.3	3.28	14.2	2.24	4.3
C_2	1.35	1.6	1.92	3.2	3.72	17.0	2.87	8.1
C_3	1.07	1.6	1.96	5.3	6.27	40.2	2.49	7.9
C_4	2.94	7.7	3.18	7.9	4.85	20.8	3.23	5.7
C_5	3.17	7.7	3.36	8.5	4.99	19.1	3.44	6.0
C_6	2.59	6.7	2.72	6.9	5.32	18.8	2.92	7.6
C_7	2.37	4.3	2.59	9.4	4.68	18.5	2.28	5.5
C_8	2.15	4.9	2.41	7.8	4.24	16.9	2.38	5.6
C_9	2.18	6.0	1.87	5.4	2.63	10.3	1.56	3.1
C_{10}	1.55	3.5	1.97	4.1	3.59	12.7	2.16	5.4
C_{11}	2.28	5.8	2.37	6.4	3.48	12.1	2.25	5.4
C_{12}	2.15	4.6	2.50	8.7	5.01	23.4	2.14	5.0
C_{13}	2.47	5.0	3.01	11.3	5.59	25.5	2.30	5.9
C_{14}	1.91	3.1	2.27	5.7	3.81	15.0	2.63	8.0
C_{15}	2.55	6.4	3.07	5.3	5.22	33.0	3.05	8.5
C_{16}	1.89	4.5	2.19	6.1	3.93	19.2	2.39	8.6

TABLE 2.6: BIT ERROR RATE PERFORMANCE FOR CODES IN TABLE 2.4. SIMULATIONS WITH TRACEBACK DEPTH $L_D = 40$

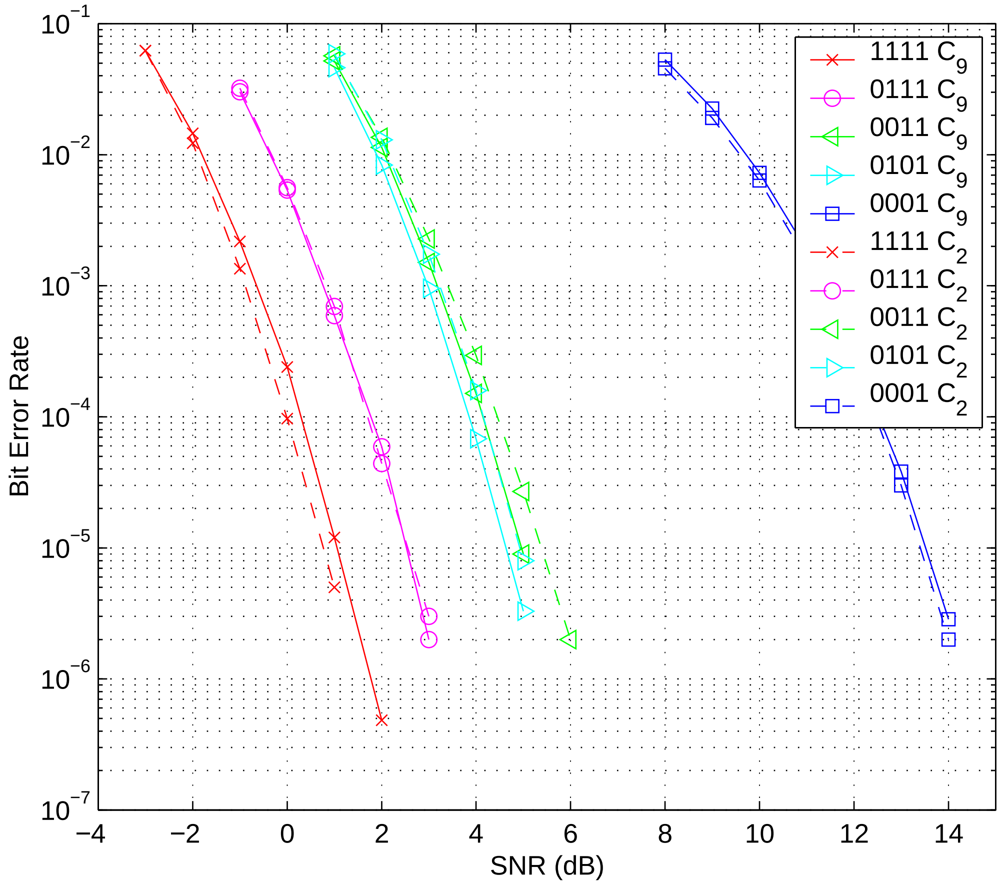


FIGURE 2.11: PERFORMANCE COMPARISON OF CODE C_2 AND C_9

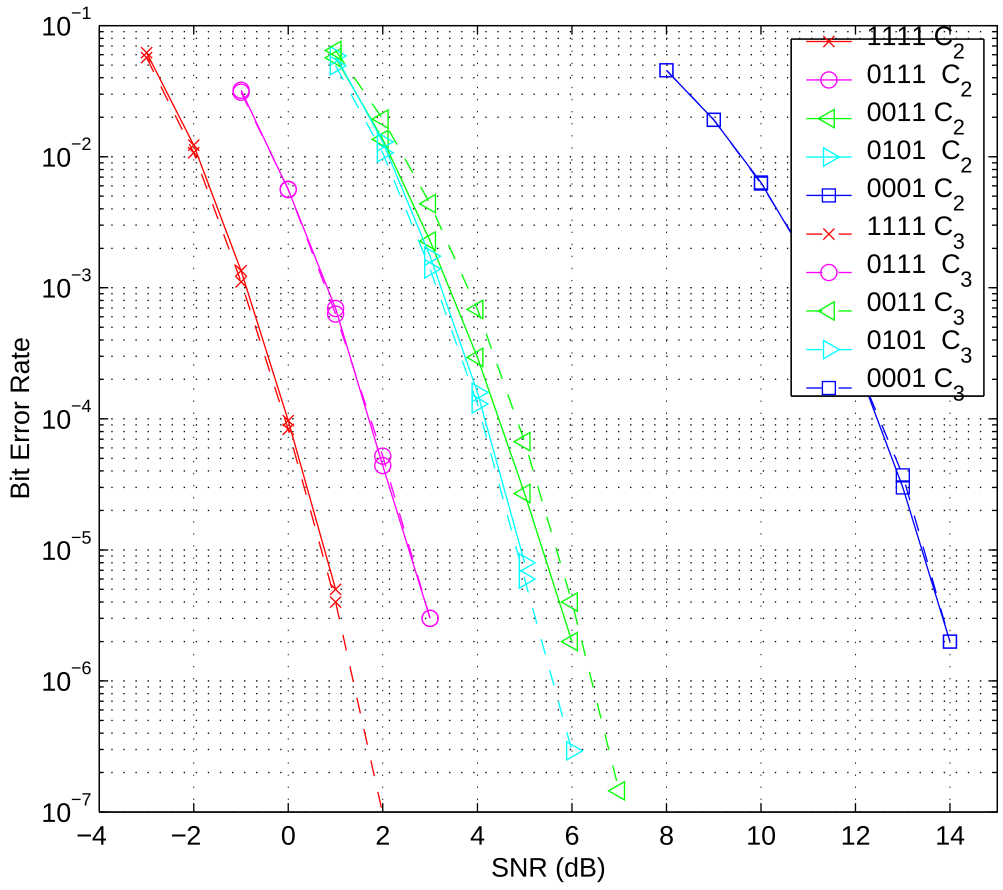


FIGURE 2.12: PERFORMANCE COMPARISON OF CODE C_2 AND C_3

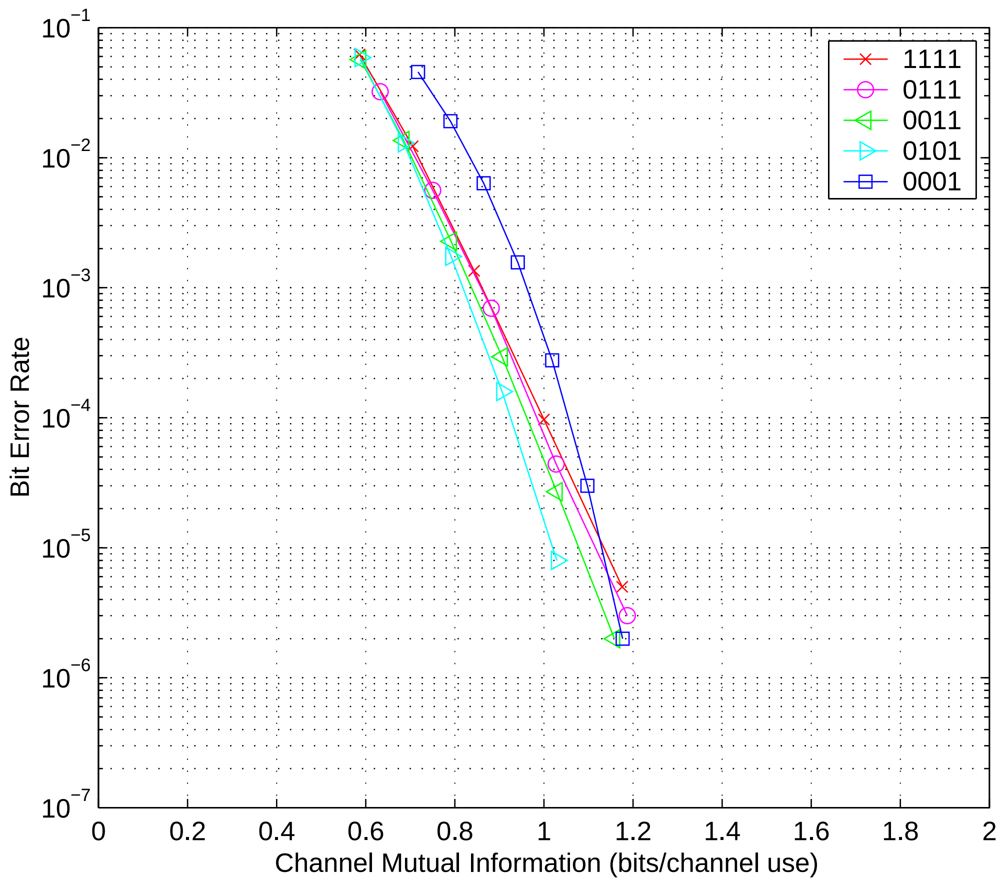


FIGURE 2.13: PERFORMANCE OF CODE C_2 VS. MUTUAL INFORMATION UNDER PERIODIC SYMBOL PUNCTURING WITH PERIOD $p = 4$. TRANSMITTED RATE IS 0.25 BITS PER CHANNEL USE

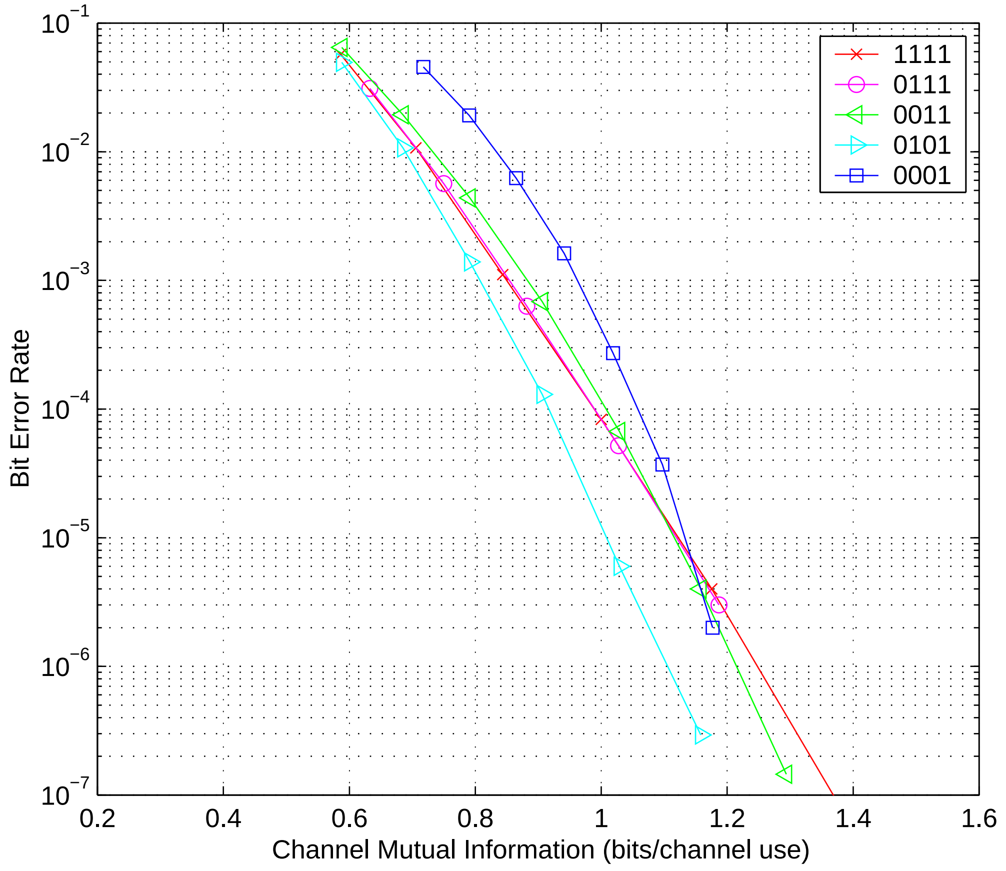


Figure 2.14: Performance of code C_3 vs. mutual information under periodic symbol puncturing with period $p = 4$. Transmitted rate is 0.25 bits per channel use

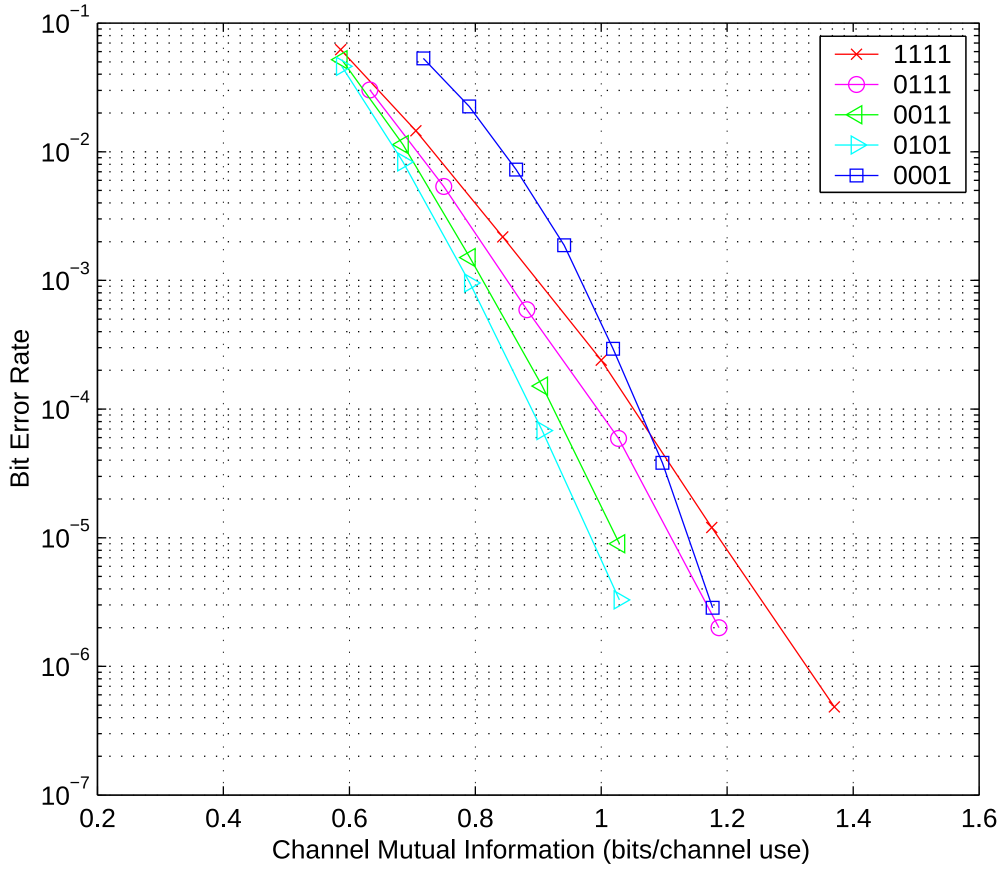


FIGURE 2.15: PERFORMANCE OF CODE C_9 VS. MUTUAL INFORMATION UNDER PERIODIC SYMBOL PUNCTURING WITH PERIOD $p = 4$. TRANSMITTED RATE IS 0.25 BITS PER CHANNEL USE

CHAPTER 3

Proposed Turbo Code Structure and Constituent Encoder Design

This chapter presents a method for parallel concatenated trellis coded modulation (PCTCM) with constituent encoders of rate k/n, $k > 1$. The k binary inputs are one symbol over the extension field $GF(2^k)$. Two main approaches are proposed in the literature for the turbo encoder structure, one employing bit interleaving by Benedetto et al. [BDM96], and the other employing symbol interleaving by Robertson and Wörz [RW96, RW98].

For bit interleaving, k bit interleavers are used to keep the bit streams separate. The first constituent encoder in [BDM96], for k even, has half of the k input bits as systematic outputs and a parity output. The second constituent encoder is the same as the first, but the other half of the k input bits become systematic. Thus the overall turbo encoder is systematic.

For symbol interleaving as described in [RW96, RW98], to have the overall turbo encoder systematic, the interleaver maps even symbol positions to even symbol positions and odd ones to odd. The output of the second encoder is de-interleaved and the output symbols from each encoder are punctured alternatively. The odd-to-odd and even-to-even interleaving was first described by Barbulescu and Pietrobon in [BP94], and is equivalent to using two separate symbol interleavers of half the length, one for the odd positions and another for the even ones. This additional structure of the symbol interleaver reduces the interleaving gain, as is also observed by Ogiwara and Yano in [OY98]. Moreover, puncturing complicates the design of the constituent encoders.

Our proposed approach combines the turbo encoder approach of [BDM96] with a symbol interleaver. Each k/n constituent encoder, for k even, has $k/2$ systematic outputs and $r \geq 1$ parity outputs. The $n = \frac{k}{2} + r$ total output bits of the encoder are mapped to one constellation point. The upper constituent encoder has as systematic outputs the $k/2$ MSB input bits while the lower constituent encoder has as systematic outputs the $k/2$ LSB input bits. Thus the systematic bits are evenly divided between the constituent encoders without puncturing or interleaver constraints as in [RW96] (in [OY98] the interleaver constraints are removed but puncturing is still employed).

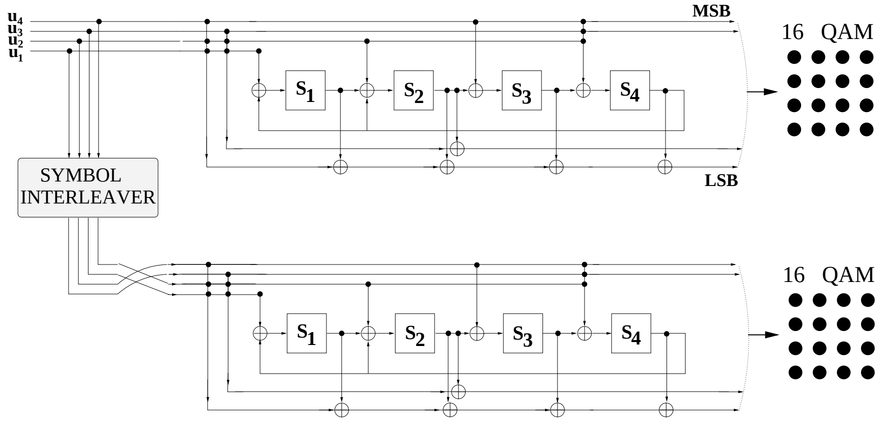


FIGURE 3.1: 2 BITS/SEC/Hz PCTCM TURBO CODE WITH RATE 4/4 CONSTITUENT ENCODERS

Fig. 3.1 shows an example of the proposed parallel turbo code structure that employs 16 QAM modulation in connection with rate 4/4 constituent encoders, each with $\frac{k}{2} = 2$ systematic and $r = 2$ parity outputs. Fig. 3.2 shows a second example of the proposed structure that employs 8 PSK modulation in connection with for rate 4/3 constituent encoders, each with $\frac{k}{2} = 2$ systematic and $r = 1$ parity outputs. The generalization to k/n encoders using 2^n-point constellations is straightforward when k is even.

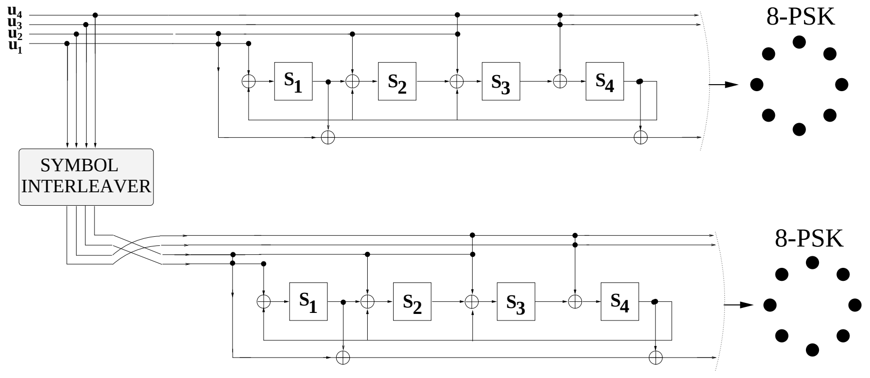


FIGURE 3.2: 2 BITS/SEC/HZ PCTCM TURBO CODE WITH RATE 4/3 CONSTITUENT ENCODERS

Generally, using a symbol interleaver is equivalent to using k bit interleavers that implement the same interleaving pattern. In contrast, interleaving the k bits separately allows spreading of the components of one error event to k-times more error events, typically accumulating more distance. Thus the symbol interleaver imposes a structure that reduces the interleaver gain of a turbo encoder. Despite the loss in interleaver gain, we are motivated to use symbol interleaving because it imposes fewer assumptions on iterative decoding, as discussed below.

Our iterative decoder implements the Soft Input Soft Output (SISO) equations appearing in [BDM97], with input bit probabilities substituted by input symbol probabilities. Let $\mathbf{Y_1}$ be the observed sequence at the SISO module corresponding to the upper constituent encoder, and $\mathbf{u} = \{\mathbf{u_t}\}$ the input symbols sequence we try to estimate. The iterative turbo decoder uses the assumption that the exchanged input symbol probabilities are independent. This is not true because they are conditioned on the observed output sequence $\mathbf{Y_1}$:

$$P(\mathbf{u}|\mathbf{Y_1}) \neq \Pi_t P(\mathbf{u_t}|\mathbf{Y_1}) \tag{3.1}$$

Using bit interleaving leads to the additional assumption that the bits $\{u_{t,i}\}$ within each symbol $\mathbf{u_t}$ are also independent. Again this is not true:

$$P(\mathbf{u_t}|\mathbf{Y_1}) \neq \Pi_i P(u_{t,i}|\mathbf{Y_1}) \tag{3.2}$$

Symbol interleaving avoids this additional assumption of independence (assuming equality in (3.2)).

The use of a symbol interleaver implies that the constituent encoders should be optimized for "symbol effective free distance." This term refers to the minimum output distance when the input symbol sequence has exactly two symbols different from zero as opposed to the usual notion of effective free distance which refers to the minimum output distance for a binary input Hamming distance of two.

In the rest of this dissertation, we use several variations of effective free distance. The superscript refers to the output distance, Hamming (H) or Euclidean (E). The number in the subscript denotes the input weight, whether bit-wise (b) or symbol-wise (s). We always imply squared Euclidean distance. For example, d_{s2}^{E} stands for the output squared Euclidean distance when the symbol-wise input weight is two.

3.1 Desired Constituent Encoder Distance Properties

An analytical upper bound to the bit error probability of turbo codes by Benedetto and Montorsi in [BM96] identified effective free distance as a key parameter. A similar analysis still holds when the input of the constituent encoders is over

GF(2^k), with the slight modification that the input Hamming weight now refers to Hamming weight in the extension Galois field $GF(2^k)$. Repeating the analysis for symbol-wise input along the lines of [BM96] (we do not repeat the exact derivation here), two main guidelines for the design of constituent encoders are derived:

- For a given symbol interleaver length, to achieve interleaver gain, the constituent convolutional encoders must have infinite output weight when the input symbol sequence contains only one symbol different than zero ($d_{s1}^H = \infty$).
- Among the encoders with $d_{s1}^H = \infty$, the ones with the best symbol effective distance (Hamming d_{s2}^H or Euclidean d_{s2}^E depending on the application) optimize the asymptotic turbo code performance.

The first guideline equivalently states that there should be no parallel transitions in the trellis diagram, which was also presented in [RW96].

3.2 Distance Upper Bounds

Consider convolutional codes with k binary inputs, m memory elements, and r parity (not systematic) outputs. Assume that $d_{b1}^H = \infty$, i.e. the impulse response corresponding to every one of the k binary inputs is infinite. Divsalar et al. presented in [DP95b] and [DM96] the following bound on the effective free distance d_{b2}^H that is a key design metric for constituent encoders used with bit interleaving [BM96]:

$$d_{b2}^H \leq \min\left(\left\lceil \frac{2^m}{k} \right\rceil r, 2r + \left\lfloor \frac{2^{m-1}r}{k} \right\rfloor\right) \tag{3.3}$$

where $\lfloor x \rfloor$ denotes the largest integer smaller than x, and $\lceil x \rceil$ denotes the smallest integer larger than x.

For symbol interleaved PCTCM, it is interesting to examine the d_{s2}^H bound. It can be shown that an upper bound to d_{s2}^H, when $d_{s1}^H = \infty$, with r parity (not systematic) outputs and k binary inputs, is given by substituting k with $2^k - 1$ in (3.3):

$$d_{s2}^H \leq \min \left(\left\lceil \frac{2^m}{2^k - 1} \right\rceil r, 2r + \left\lfloor \frac{2^{m-1} r}{2^k - 1} \right\rfloor \right) \tag{3.4}$$

The proof follows along the same lines as the proof of (3.3) given in [DP95b]. The main point is the following: If the feedback polynomial of a convolutional encoder is primitive, then the state diagram has one loop with zero inputs and nonzero outputs. This loop includes all the $2^m - 1$ nonzero states. An input sequence with two non-zero symbols causes the encoder to enter the loop (with the first nonzero input) and exit it (with the second nonzero input). The output weight of any output parity bit going around the whole loop is 2^{m-1}. If k binary inputs exist, there are k ways to enter and leave the loop via single input bits, and thus the minimum output weight of a single parity output, along the part of the loop that it travels before it exits, can be in the best case, $2^{m-1}/k$. Considering symbol inputs, there are instead $2^k - 1$ ways to join/exit this loop, and thus the minimum output weight of a single parity bit can be, in the best case, $2^{m-1}/(2^k - 1)$. This reasoning leads to the second argument of the minimization in (3.4). In general, $2^k - 1$ inputs should be taken into account instead of k, which can be similarly applied to the bit-wise proof in [DM98] for the first argument and for non-primitive feedback polynomials.

The upper bound (3.4) indicates that there is less symbol-wise effective free distance available than bit-wise, as expected. Indeed, grouping any convolutional code's error events symbol-wise instead of bit-wise, can only reduce the effective distance.

3.3 Range of Encoders to Search

This section, relies on the framework developed in Chapter 2 to examine the structural properties of encoders that should be included in an exhaustive search for turbo code constituent encoders.

When searching for encoders that maximize effective distance, we can rule out codes that are input-Hamming-weight equivalent to already examined ones.

Definition 18 Input-Hamming-weight equivalent *Two encoders are called input-Hamming-weight equivalent if they map the same input weight error events to the same output distance.*

If two encoders have the same mapping from input to output error events, then they also have the same mapping from input weight error events to output error events. Thus two algebraically equivalent encoders (definition 5) are also input-Hamming-weight equivalent. The set of input-Hamming-weight equivalent encoders to an encoder $\mathbf{C}$ is a subset of the set of encoders that are algebraically equivalent to $\mathbf{C}$, as Fig. 3.3 illustrates.

Two strictly equivalent encoders (definition 2) are not necessarily input-Hamming-weight equivalent, because two encoders with the same mapping from input sequences to output sequences do not necessarily have the same input weight error events. Also, two input-Hamming-weight equivalent encoders are not necessarily strictly equivalent, because different inputs can have the same input weight.

Chapter 2 introduced the state space description of convolutional codes. For example, the upper constituent encoder in Fig. 3.1 with feedback polynomial

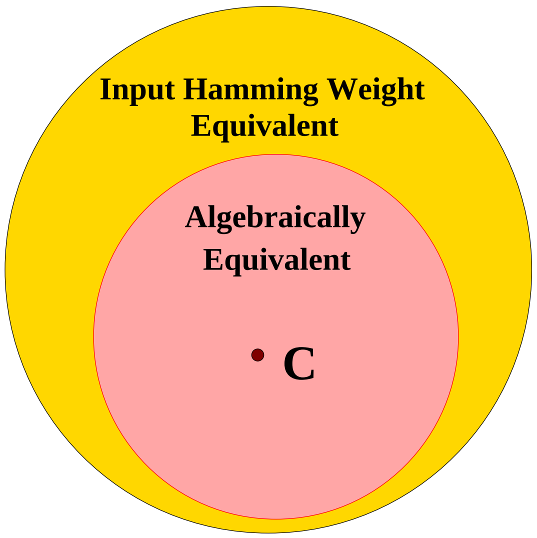


FIGURE 3.3: SET OF INPUT HAMMING WEIGHT EQUIVALENT AND ALGEBRAIC EQUIVALENT ENCODERS TO AN ENCODER **C**

$f(D) = D^4 + D + 1$ has the following state-space description:

$$[s_1\ s_2\ s_3\ s_4]_{j+1} = [s_1\ s_2\ s_3\ s_4]_j \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}}_{\mathbf{A_1}} + [u_1\ u_2\ u_3\ u_4]_j \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}}_{\mathbf{B_1}} \tag{3.5}$$

$$[y_1\ y_2\ y_3\ y_4]_j = [s_1\ s_2\ s_3\ s_4]_j \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{C_1}} + [u_1\ u_2\ u_3\ u_4]_j \underbrace{\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{D_1}} \tag{3.6}$$

Section 2.2.1 established that the range of matrices **A** to consider, that do not lead to algebraic equivalent encoders, can be found from the rational form theorem presented in Horn and Johnson ([HJ85], pg. 154) and the references therein. This theorem states that for any convolutional encoder with m memory elements, no matter how these memory elements are connected, there exists an algebraic equivalent encoder with the memory elements connected in R rows for some R. The following theorem helps to further refine the encoder structures of interest for turbo codes.

Theorem 6 *Consider a convolutional encoder with k inputs, r parity outputs, and m memory elements. For all (k, m, r) values such that*

$$k < \min\left(2^{m-1} - 2, r(2^{m-2} - 1)\right) \tag{3.7}$$

the bound in (3.3) cannot be achieved if the m memory elements are connected in multiple "disconnected" memory rows, that is, multiple distinct memory rows with no common inputs.

Proof
It suffices to show that the use of multiple rows of memories enforces an upper bound on the effective free distance lower than the bound in (3.3).

Assume that the m memory elements are connected in R rows with m_j memory elements in row j, $j = 1 \ldots R$, and $\sum_{j=1}^{R} m_i = m$. Let k_j be the number of inputs in row j, $k_j \leq k$, $\sum_{j=1}^{R} k_j = k$, and r_j be the number of outputs from row j, $r_j \leq r$. In the example of Fig. 3.4, $R = 2$, $m = 4$, $m_1 = m_2 = 2$, $k = 3$, $k_1 = 2$, $k_2 = 1$, $r = 3$ and $r_1 = r_2 = 2$.

For one memory chain j, d_{b2}^{H} is bounded by:

$$d_{b2}^{H} \leq \min\left(\left\lceil \frac{2^{m_j}}{k_j} \right\rceil r_j, 2r_j + \left\lfloor \frac{2^{m_j-1} r_j}{k_j} \right\rfloor\right) \tag{3.8}$$

So for the total encoder it holds that:

$$\begin{aligned} d_{b2}^{H} &\leq \min_{j=1\ldots R}\left(\min\left(\left\lceil \frac{2^{m_j}}{k_j} \right\rceil r_j, 2r_j + \left\lfloor \frac{2^{m_j-1} r_j}{k_j} \right\rfloor\right)\right) \\ &\leq \min_{j=1\ldots R}\left(\min\left(\left\lceil \frac{2^{m_j}}{k_j} \right\rceil r, 2r + \left\lfloor \frac{2^{m_j-1} r}{k_j} \right\rfloor\right)\right) \end{aligned} \tag{3.9}$$

To show that the multiple-rows upper bound is lower, it suffices to show that it is lower for both the two terms of the *min* function in (3.3).

1. For the second term of the bound:

   If the m memories are connected in one row:

$$d_{b2}^{H} \leq 2r + \left\lfloor \frac{2^{m-1}r}{k} \right\rfloor \tag{3.10}$$

   If the m memories are connected in R rows:

$$d_{b2}^{H} \leq \min_{j=1\ldots R} \left(2r + \left\lfloor \frac{2^{m_j-1}r}{k_j} \right\rfloor \right) \tag{3.11}$$

$$\leq 2r + \left\lfloor \frac{\sum_{j=1}^{R} 2^{m_j-1}r}{k} \right\rfloor \tag{3.12}$$

   Note that perhaps $m_j \leq m^*$, but (3.12) is still an upper bound to (3.11).

2. Similarly, for the first term of the bound:

   If the m memories are connected in one row:

$$d_{b2}^{H} \leq \left\lceil \frac{2^m}{k} \right\rceil r \tag{3.13}$$

   If the m memories are connected in R rows:

$$d_{b2}^{H} \leq \left\lceil \frac{\sum_{j=1}^{R} 2^{m_j}}{k} \right\rceil r \tag{3.14}$$

If it holds that $k < 2^m - \sum_{j=1}^{R} 2^{m_j}$ for all possible R and m_j partitions, then (3.13) and (3.14), are related by a strict inequality. However for any integer q, $1 \leq q \leq m$, integers m_j, $0 < m_j < m$, and integer $R > 1$ it holds that:

$$\sum_{j=1}^{R} 2^{m_j} \leq 2^{m-q} + 2^q \leq 2^{m-1} + 2 \tag{3.15}$$

so it suffices that $k < 2^{m-1} - 2$. Similarly, for strict inequality between (3.10) and (3.12), it suffices that $k < r(2^{m-2} - 1)$. □

Disconnected memory rows are described by block diagonal matrices **A** and **B**, so the state equation in (2.1) can be decomposed into a separate state equation for each memory row. Fig. 3.4 shows an example of an encoder with two disconnected memory rows, described by the state equation:

$$[s_1\ s_2\ s_3\ s_4]_{j+1} = [s_1\ s_2\ s_3\ s_4]_j \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{A}} + [u_1\ u_2\ u_3]_j \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}}_{\mathbf{B_1}} \tag{3.16}$$

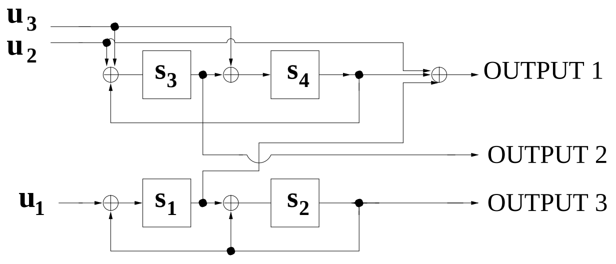


FIGURE 3.4: ENCODER STRUCTURE WITH TWO MEMORY CHAINS

In an exhaustive search, if the memory elements connected in a single row (most "connected" case) give a code with d_{b2}^H that achieves the upper bound, there is no need to expand the search to multiple rows. This is very often the case in practice. The rational form theorem and Theorem 6 usually result in a small number of matrices **A** to examine. However, for each matrix **A** all matrices **B**, **C**, and **D** still have to be examined.

The input vector transformation $\hat{u} = u\mathbf{T}$ for any $k \times k$ matrix **T** that performs row permutation also leads to input-Hamming-weight equivalent encoders. Indeed, the corresponding generator matrices $G(D)$ and $\mathbf{T}G(D)$ still map the same input weight error events to the same output distance. Thus in an exhaustive search there is no need to examine both sets of matrices (**B**, **D**) and (**TB**, **TD**). For example, keep only the matrices **B** where each row (interpreted as a binary number) is greater than the previous row, coupled with all possible matrices $\mathbf{D}$. Similarly output Hamming weight is not affected by permuting the r outputs, which in this case reduces the number of matrices $\mathbf{C}$ to examine.

3.3.1 Application to Bit Interleaving

Table 3.1 provides code fragments with k inputs, r parity outputs, and m memory elements optimized for d_{b2}^{H} and identified through exhaustive search using our proposed structure. Such tables are useful for bit-interleaving coupled with BPSK or 4-PSK modulation, and outperform in terms of d_{b2}^{H} similar code tables provided in [DM96]. Benedetto et al. use a group theoretic approach to propose a different structure in [BGM98] and provide encoder tables that are equally good (but not better) in terms of d_{b2}^{H} than the codes identified in Table 3.1.

To describe an encoder we give in octal notation the feedback polynomial f, the k rows $\{\mathbf{b}_1 \ldots \mathbf{b}_k\}$ of matrix $\mathbf{B}$, the r columns $\{\mathbf{c}_1 \ldots \mathbf{c}_r\}$ of matrix $\mathbf{C}$, and the r columns $\{\mathbf{d}_1 \ldots \mathbf{d}_r\}$ of matrix $\mathbf{D}$ that correspond to the r parity outputs. For example the upper constituent encoder in Fig. 3.1 with state-space equations (3.5), (3.6) is described as:

$$\{f, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{b}_4, \mathbf{c}_1, \mathbf{c}_2, \mathbf{d}_1, \mathbf{d}_2\} = \{023, 010, 05, 01, 03, 017, 04, 015, 016\}$$

Table 3.1 includes for each code fragment the upper bound on the effective distance d_{b2}^{u}, the d_{b2}^{H} distance, the d_{b3}^{H} distance, and the free distance denoted by d_{free}, with the number of nearest neighbors in parentheses. The code fragments can be made systematic by adding k systematic outputs. So although their free distance might be zero, the free distance of the complete code will be positive because of the additional systematic outputs. Codes noted with a * have repeated outputs and do not perform well in simulations [DM96].

(r=1, k=2) code fragments					
m	$\{f,\mathbf{b}_1,\mathbf{b}_2,\mathbf{c}_1,\mathbf{d}_1\}$	$\mathbf{d^u_{b2}}$	d^H_{b2}	d^H_{b3}	d_{free}
1	$\{03, 01, 01, 01, 0\}$	**1**	0(1)	∞	0(1)
2	$\{07, 02, 01, 03, 02\}$	**2**	**2**(3)	0(1)	0(17)
3	$\{011, 02, 05, 03, 03\}$	**4**	**4**(2)	2(4)	0(33)
4	$\{027, 010, 03, 07, 03\}$	**6**	**6**(2)	4(12)	0(65)
5	$\{053, 020, 03, 010, 03\}$	**10**	**10**(2)	5(12)	0(129)

(r=2, k=2) code fragments					
m	$\{f,\mathbf{b_1},\mathbf{b_2},\mathbf{c_1},\mathbf{c_2},\mathbf{d_1},\mathbf{d_2}\}$	$\mathbf{d^u_{b2}}$	d^H_{b2}	d^H_{b3}	d_{free}
1	$\{03, 01, 01, 01, 0, 01, 02\}$	**2**	**2**(5)	∞	2(37)
2	$\{07, 02, 03, 02, 03, 01, 03\}$	**4**	**4**(2)	2(2)	2(34)
$*3$	$\{011, 02, 05, 03, 03, 03, 03\}$	**8**	**8**(2)	4(4)	0(33)
3	$\{015, 04, 05, 04, 07, 03, 02\}$	**8**	7(2)	3(1)	1(1)
4	$\{023, 010, 012, 02, 013, 03, 03\}$	**12**	**12**(2)	3(1)	1(2)

(r=1, k=3) code fragments					
m	$\{f,\mathbf{b}_1,\mathbf{b}_2,\mathbf{b}_3,\mathbf{c}_1,\mathbf{d}_1\}$	$\mathbf{d^u_{b2}}$	d^H_{b2}	d^H_{b3}	d_{free}
1	$\{03, 01, 01, 01, 01, 0\}$	**1**	0(3)	∞	0(3)
2	$\{05, 02, 02, 03, 03, 05\}$	**2**	1(2)	0(2)	0(∞)
3	$\{017, 04, 02, 07, 06, 05\}$	**3**	2(6)	∞	0(∞)
4	$\{033, 014, 02, 05, 013, 07\}$	**4**	**4**(2)	2(7)	0(∞)

Table 3.1: Code fragments optimized for d^H_{b2}

3.3.2 Application to Symbol Interleaving

For PCTCM we are interested in $(k, m, r + k/2)$ constituent encoders with r parity and $k/2$ systematic outputs optimized for d^E_{s2}. Theorem 6 can be extended to symbol-wise inputs by replacing k with $2^k - 1$. Theorem 6 and (3.4) refer to output Hamming distance. For the simulations we want to maximize the output Euclidean distance d^E_{s2}. Although there is no monotone relation between Hamming and Euclidean distance, they are closely related. For example, for 16 QAM and Gray labeling [WL00], it holds that:

$$d^H \leq d^E \leq 2d^H \tag{3.17}$$

Proof

A constellation's edge profile lists the minimum distance edge for each binary symbol error [WLC00]. Each edge can be expressed as a linear combination of the basis vectors that label the constellation. For 16 QAM and Gray labeling, the basis vectors v_1, v_2, v_3, $v4$ in [Wes96] Table 3.2 pg. 47 must be equal the orthonormal basis vectors B={0001, 0100, 0010, 1000} ([Wes96] Chapter 4). Each of the vectors in B corresponds to Hamming weight one. The sum of any two vectors in B corresponds to Hamming weight two. The sum of any two vectors in B corresponds to Hamming weight three. The sum of all four vectors corresponds to Hamming weight four. Thus Table 3.2, which reproduces part of

Edge label	d^H	d^E	$2d^H$
v_1	1	1	2
v_2	1	1	2
$v_2 + v_1$	2	2	4
v_3	1	1	2
$v_3 + v_1$	2	2	4
$v_3 + v_2$	2	4	4
$v_3 + v_2 + v_1$	3	5	6
v_4	1	1	2
$v_4 + v_1$	2	4	4
$v_4 + v_2$	2	2	4
$v_4 + v_2 + v_1$	3	5	6
$v_4 + v_3$	2	2	4
$v_4 + v_2 + v_1$	3	5	6
$v_4 + v_3 + v_1$	3	5	6
$v_4 + v_3 + v_2$	3	5	6
$v_4 + v_3 + v_2 + v_1$	4	8	8

TABLE 3.2: PART OF TABLE 3.2 IN [WES96].

Table 3.2 in [Wes96] for the readers convenience, proves the claim in (3.17). □

Motivated by the previous arguments, and because completely exhaustive searches are beyond our computational capabilities, for the simulation results in Chapter 5 we restrict our attention to encoders with memory elements connected in a single row.

For symbol interleaving, to have $d_{s1} = \infty$, the k rows $\{\mathbf{b}_1 \ldots \mathbf{b}_k\}$ of matrix $\mathbf{B} \epsilon GF(2)^{k \times m}$, have to be linearly independent, i.e. matrix $\mathbf{B}$ has to be full-rank and $k \leq m$. Moreover, the input vector transformation $\hat{u} = u\mathbf{T}$ for any $k \times k$ nonsingular matrix $\mathbf{T}$ does not affect the *symbol-wise* input-Hamming-weight. So there is no need to examine both sets of matrices: $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ and $\{\mathbf{A}, \mathbf{TB}, \mathbf{C}, \mathbf{TD}\}$ for any nonsingular $\mathbf{T}$.

In the special case where $k = m$, the rows/columns of matrix $\mathbf{B}$ form a basis. Any other basis is related with a linear transformation to it. For example, matrix $\mathbf{B_1}$ is related with a linear transformation to matrix $\mathbf{B_2}$:

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}}_{\mathbf{B_1}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}}_{\mathbf{T}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{B_2}} \tag{3.18}$$

So in an exhaustive search it is sufficient to use any one full-rank matrix $\mathbf{B}$, since any other full-rank matrix $\hat{\mathbf{B}}$ is related through a linear transformation $\mathbf{T}$ to it, coupled with all possible matrices $\mathbf{D}$.

For our exhaustive searches we could not examine one matrix $\mathbf{B}$ coupled with all matrices $\mathbf{D}$, because our constituent encoders are half-systematic which implies a special structure on matrix $\mathbf{D}$:

$$\underbrace{\begin{bmatrix} . & . & 0 & 0 \\ . & . & 0 & 0 \\ . & . & 1 & 0 \\ . & . & 0 & 1 \end{bmatrix}}_{\mathbf{D}} \tag{3.19}$$

CHAPTER 4

Interleaver Design for Turbo Codes

The turbo encoder performance depends upon both the constituent encoders and the interleaver it employs. Chapter 2 addressed the constituent encoders design. This chapter addresses the interleaver design, which again can be applied to both bit and symbol interleaving.

Interleaver design for turbo codes is mainly targeted towards lowering the error floor, which is the flattening of the error-rate curve that turbo codes exhibit for moderate and high values of SNR. In this scope, Perez et. al. [PSC96] have shown that the turbo code asymptotic performance approaches the free-distance asymptote. The error floor observed with turbo codes is due to their relatively small free distance and consequently relatively flat free-distance asymptote. Thus to lower the error floor the free-distance of the turbo code must be maximized for a fixed interleaver length. Based on this observation various methods (for example [And97, OS97]) have been proposed to maximize the turbo code output distance by designing the interleaver tailored to specific constituent codes, and trying to break turbo code error events of low output weight. Some methods (for example [DM97, HM97]) also propose interleaver design through cost function minimization.

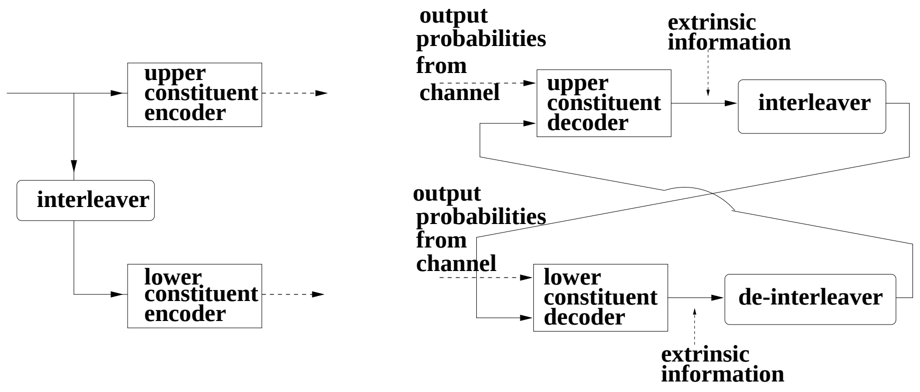
This chapter starts with the spread-interleaver introduced by Divsalar and

Pollara and extends the design criteria to multiple error events based on the interleaver's role in the overall error event distances. The extension helps to explain why the spread interleaver is specifically designed to be semi-random.

An interleaver of length N is completely described by a mutually exclusive and collectively exhaustive listing of the integers from 1 to N. Define $f(i)$ to be the integer in the i^{th} position in the list. The input symbol in position i before interleaving is in position $f(i)$ after interleaving.

4.1 Spread Interleaver

Consider a parallel concatenated turbo code as in Fig. 4.1. The interleaver takes the input block of the upper encoder and produces the input block for the lower encoder.



upper constituent encoder
interleaver
lower constituent encoder
output probabilities from channel
upper constituent decoder
extrinsic information
interleaver
output probabilities from channel
lower constituent decoder
de-interleaver
extrinsic information

Figure 4.1: Parallel Concatenated Turbo Code

During decoding, an error event at the output of the upper decoder will be interleaved and spread to different error events of the lower decoder. The interleaver determines which error events exchange a-priori information during decoding. For simplicity from now on we refer to the interleaver input as input to the upper constituent encoder and to the interleaver output as input to the lower constituent encoder.

The role of the interleaver is to interconnect the error events of the constituent encoders in such a way that the total output weight of a turbo encoder codeword accumulates distance from as many distinct error events as possible from each constituent encoder.

A specific interleaving pattern leads to the partition of the upper and lower encoder inputs into sets of interrelated error events. For example in Fig. 4.2 we can distinguish two such sets for an example error event.

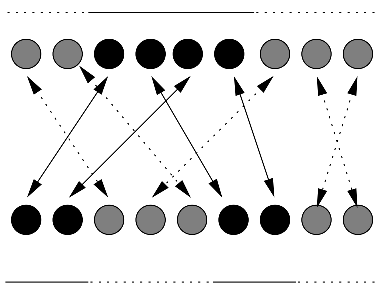


Figure 4.2: Two Sets Of Interrelated Error Events

The output weight associated with a turbo code error event, is the sum of the output weights of all the constituent error events belonging in the same set. So the bigger a set is, the more output weight it accumulates. A constituent error event with many component input symbols although it might itself have very small output distance, through the multiplicity of its inputs will be involved with a large number of error events and thus the overall turbo code error-event will have large distance.

This observation implies that the constituent events with a small number of inputs lead to the lowest output distance. A commonly used example is that when a constituent encoder has a single error event of input weight one, it unavoidably maps to a single error event of weight one in the second constituent encoder. This produces a very small total output weight, unless the constituent encoders have infinite impulse responses.

A second way to have a small number of interconnected error events is depicted in Fig. 4.3 where component symbols of one error event in the upper encoder become part of the same error event in the second encoder. This case can be avoided by using the spread interleaver introduced by Divsalar and Pollara. The spread interleaver is described in [DBP97] as a semi-random interleaver based on the random selection without replacement of N integers from 1 to N under the following constraint.

Corollary 1 *The i^{th} randomly selected integer $f(i)$ must be rejected if there exists $j < i$, such that:*

$$0 < i - j \leq S_1, \qquad |f(i) - f(j)| \leq S_2 \tag{4.1}$$

This constraint guarantees that if two symbols i, j are within distance S_1 in the upper constituent encoder, they cannot be mapped to distance less than S_2 in the lower constituent encoder.

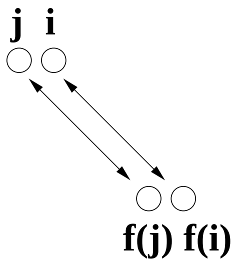


FIGURE 4.3: CASE FOR PARAMETERS S

4.2 Extended Spread Interleaver

An extension of the spread interleaver concept considers multiple error events in the upper encoder. As an example Fig. 4.4 depicts two error events of the upper encoder that interchange their component symbols, and thus the weight accumulation stops in two steps. To avoid this situation we define two more parameters T_1 and T_2 and impose on the construction of the spread interleaver an additional constraint. Again, randomly select without replacement integers from 1 to N, and if the i^{th} selection $f(i)$ satisfies Constraint 1 described previously, check if the following condition is also satisfied.

Corollary 2 *The i^{th} randomly selected integer $f(i)$ must be rejected if there exist $j, k, l < i$, such that:*

$$\begin{aligned} 0 < i - j \leq T_1, \qquad & |f(i) - f(k)| \leq T_2, \\ 0 < |k - l| \leq T_1 \qquad & |f(j) - f(l)| \leq T_2 \end{aligned} \tag{4.2}$$

This constraint guarantees that two relatively close component symbols i and j in the upper encoder, do not have $f(k)$ near $f(i)$ and $f(l)$ near $f(j)$ in the lower encoder, with k and l near each other in the upper encoder. Figs. 4.4 and 4.5 illustrate error events that are avoided.

This procedure can be extended to three error events in the upper encoder. Define parameters X_1 and X_2 and impose on the semi-random interleaver the following additional condition.

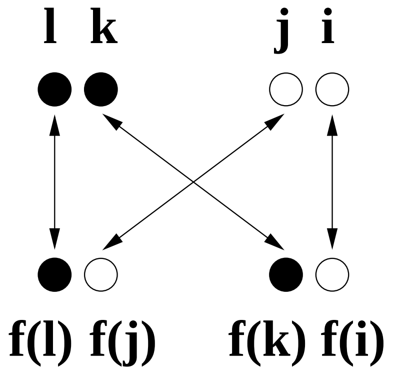


FIGURE 4.4: CASE FOR PARAMETERS T

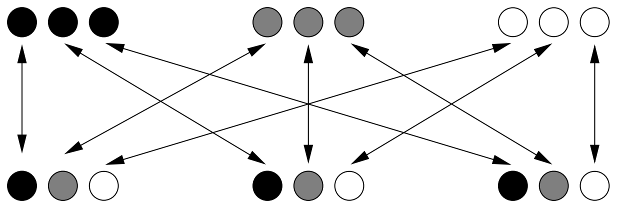


FIGURE 4.5: ANOTHER CASE FOR PARAMETERS T

Corollary 3 *The i^{th} randomly selected integer $f(i)$ must be rejected if there exist $j, k, l, m, n < i$, such that:*

$$\begin{aligned} &0 < i - j \leq X_1, & &|f(i) - f(k)| \leq X_2, \\ &0 < |k - l| \leq X_1 & &|f(j) - f(m)| \leq X_2 \\ &0 < |m - n| \leq X_1 & &|f(n) - f(l)| \leq X_2 \end{aligned} \tag{4.3}$$

Fig. 4.6 illustrates an example of an avoided error event. Extension to more than three error events is usually not of interest, because it leads to increased output weight that does not determine the free distance.

To motivate the introduction of Constraints 2 and 3 consider the following example for a symbol interleaved system with constituent encoders of rate 4/3 employing 8-PSK. The element (w,l) of Table 4.1 is the minimum squared Eu-

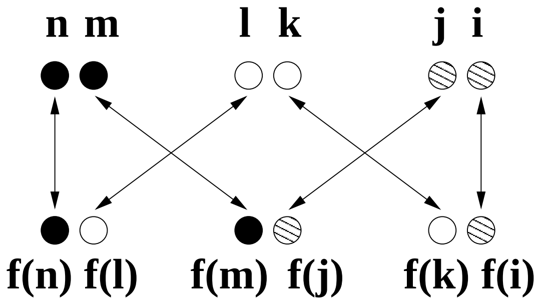


FIGURE 4.6: CASE FOR PARAMETERS X

$W \backslash L$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
2	1.17	1.17	1.76	2.34	2.34	2.34	2.34	2.93	2.93	4.10	4.10	4.10	4.10	4.69	4.69	5.87
3	–	0.59	0.59	0.59	1.17	1.17	1.75	1.75	2.34	2.34	2.34	2.92	2.92	2.92	2.92	3.51
4	–	–	0.59	0.59	0.59	0.59	0.59	0.59	1.17	1.17	1.76	1.76	1.76	2.34	2.34	2.93

TABLE 4.1: SQUARED EUCLIDEAN DISTANCE FOR ERROR EVENTS. ROWS W: SYMBOL-WISE INPUT WEIGHT. COLUMNS L: SYMBOL WISE ERROR EVENT LENGTH.

clidean distance, associated with a constituent encoder error event with input symbol-wise input Hamming weight w, and symbol-wise length l. Observe that when Constraint 1 is satisfied with $(S_1, S_2) = (10, 10)$, the minimum squared Euclidean distance that can be associated with the error event depicted in Fig. 4.3 is $4.10 + 1.17 = 5.27$ (let the upper error event have length 11 and the lower 2). For the case depicted in Fig. 4.4 though, if the constituent error events have length 2 or 3 the associated squared Euclidean distance is $4 \times 1.17 = 4.68$. For the case depicted in Fig. 4.5 the minimum squared Euclidean distance is $6 \times 0.59 = 3.54$. Thus these error events dominate the performance and should be mitigated before further increasing S_1 and S_2. Similarly, the minimum squared Euclidean distance associated with the error event depicted in Fig. 4.6 is $6 \times 1.17 = 7.02$, so this error event also determines the performance for S_1 and S_2 larger than 16.

The symbol interleaved codes in particular might benefit more from the extended interleaver design because of the larger multiplicity of error events with small number of inputs as opposed to the smaller number of such error events for bit interleaved codes.

4.3 Construction Procedure

A uniform interleaver of length N is created by randomly selecting without replacement integers from 1 to N with equal probability. For a semi-random interleaver the randomly selected integers need to satisfy a set of imposed constraints. This subsection presents a technique for constructing such interleavers.

The generation of a length N interleaver consists of N steps, where each step selects an integer for the respective position. At the i^{th} step of the interleaver generation the interleaver contains $i-1$ assigned numbers and there exist $N-i+1$ unassigned numbers. Randomly select one of the $N-i+1$ unassigned numbers with equal probability, for example number j. Check if placing number j at interleaver position i violates any of the imposed constraints. If it does not violate any constraints, then continue with the next step $i+1$. If it does violate a constraint, try to place j in between two other previously assigned indices. Uniformly choose one of the i candidate positions and check if placing j there violates any of the constraints. Continue until either all previously assigned assigned indices have been examined or a suitable position is found. If there does not exist an appropriate position, repeat for a number selected among the unassigned and not already examined $N-i$ numbers k, $k \neq j$.

4.4 Existence of Semi-Random Interleavers

The design procedure does not guarantee that it will identify a semi-random interleaver that meets the constraints, even if such an interleaver exists. Whether such an interleaver exists at all depends upon the interleaver length N and the specific design parameter values S_1, S_2, T_1, T_2 and X_1, X_2.

Corollary 4 *If only Constraint* 1 *is imposed, a necessary and sufficient condition for at least one interleaver of length N to exist is:*

$$N \geq S_1 S_2 \tag{4.4}$$

Proof

To prove the necessary part, consider an interleaver matrix of length N. We cannot have at positions i, j $|i-j| < S_1$, values that differ by $S_2 - 1 \ldots 1$. So obviously $N > S_2$. Define the set of integers:

$$U_d = \{d, d+1, \ldots, d+S_2-1\}, \text{ for } 1 \leq d \leq N - S_2.$$

None of these S_2 integers in U_d may be placed within S_1 to each other. So in the interleaver matrix, between any two of these integers at least $S_1 - 1$ other integers have to be placed (Fig. 4.7) and the interleaver has to have length $(S_2-1)(S_1-1)+S_2$. Moreover, each of the $S_1 - 1$ integers x in the first interval belongs to a different U_x set that has S_2 elements, so the interleaver must have one more integer set of length $S_1 - 1$. In total the interleaver has to contain the elements of the set U_d and at least S_2 distinct integer sets of length $S_1 - 1$, so its length has to be greater than $S_2(S_1-1)+S_2 = S_2 S_1$.

To prove that 4.4 is also a sufficient condition, we show how to construct two different interleavers of length $S_1 S_2$. Consider a block interleaver of dimensions

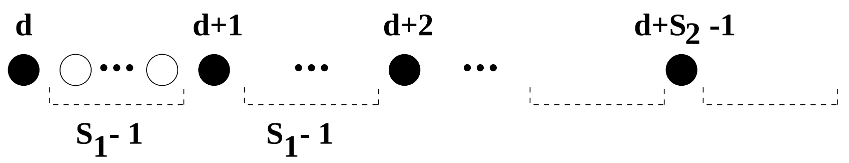


FIGURE 4.7: NECESSARY CONDITION FOR CONSTRAINT 1

$S_2 \times S_1$. Write the numbers $1 \dots N$ by columns. Each column has length S_2 so the elements in each row differ by exactly S_2:

$$\begin{bmatrix} 1 & S_2+1 & \dots & (S_1-1)S_2+1 \\ 2 & S_2+2 & \dots & (S_1-1)S_2+2 \\ \vdots & \vdots & & \vdots \\ S_2 & 2S_2 & \dots & S_1S_2 \end{bmatrix}$$

Reading though the rows starting from the upper left corner we get an interleaver that doesn't satisfy constraint 1:

$$[1\ S_2+1 \dots (S_1-1)S_2+1\ :\ 2\ S_2+2 \dots\ \dots S_1S_2].$$

The number S_2+1 is within distance S_1-1 to number 2 $\Rightarrow |i-j| = S_1-1$ and $|f(i)-f(j)| = S_2-1$. There are at least two ways to avoid this:

1. Start from the lower left end, ie form the interleaver:
   $[S_2\ 2S_2 \dots\ S_1S_2\ :\ S_2-1 \dots\ \dots\ :2\ S_2+2 \dots\ (S_1-1)S_2+2\ :\ 1\ S_2+1 \dots\ (S_1-1)S_2+1]$.
2. Start from the upper right corner, ie form the interleaver:
   $[(S_1-1)S_2+1 \dots\ S_2+1\ 1\ :\ (S_1-1)S_2+2 \dots\ S_2+2\ 2\ :\ \dots\ \dots\ S_2]$.

As has been noted in the literature, this kind of block interleaver does not perform well. Observe that constraint 2 is consistenly not met, since for each neighbor integers k, l the intergers $k+1$ and $l+1$ are also neighbor. Thus the spread interleaver is specifically designed to be semi-random; it is not sufficient to satisfy constraint 1.

Similarly for Constraint 2, we can find a lower bound of the interleaver length as following: Again define the set V_d of integers that are within T_2 to an arbitrary integer d.

$$V_d = \{d, d+1, \dots, d+T_2-1\}, \text{ for } T_2 \le d \le N - T_2.$$

Assume that $T_1 \le 2S_1$, and $T_2 < S_2$ as is usually the case.

Consider Fig. 4.8. Because of Constraint 1, the elements of V_d cannot be placed within S_1 to each other. Let X_d be the set of $2(T_1-1)$ integers placed within T_1 to integer d. Each integer $j \in X_d$ prevents another $2(T_2-1)$ integers from taking part in the sets: $X_{d+1}, \dots X_{d+T_2-1}$, or else Constraint 2 will be violated. Since there exist T_2-1 X-sets the interleaver length has to be at least:

$$N \ge 4(T_2-1)^2(T_1-1) + T_2T_1$$

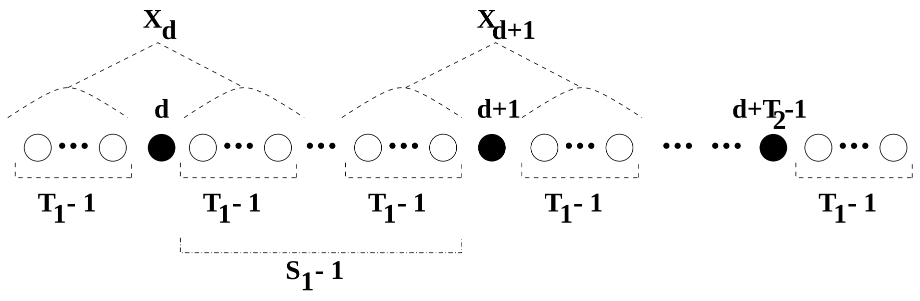


Figure 4.8: Necessary Condition for Constraint 2

CHAPTER 5

Simulation Results

This chapter provides simulation results of the proposed system for 2 bits/sec-/Hz employing 16-QAM and 8-PSK, and 4 bits/sec/Hz employing 64-QAM= $2 \times$ 8-PAM.

The performance metrics are Bit Error Rate, and Block Error Rate vs. SNR (in dB). Bit Error Rate is a typical performance metric for channel codes. Turbo code data transmission and decoding occurs in blocks, thus Block Error Rate promotes understanding of the system's behavior, and is the appropriate metric for systems employing ARQ protocols or some other form of treating block (as opposed to bit) failure.

The turbo code performance curve consists of three parts: the region before convergence, the waterfall region where the code converges and the performance achieves low error rates within tenths of dB, and the error floor where the error curve flattens and follows the free distance asymptote.

Two criteria assess the performance of a turbo code. The first is at what SNR the code converges, which determines the lowest SNR (in dB) at which the code may achieve an acceptable performance (around 10^{-5} for Bit Error Rate below 10^{-2} for Block Error Rate). This SNR is compared against the constrained capacity, which is the mutual information between the channel's input drawn uniformly from a finite constellation and the channel's output [Ung82]. The convergence SNR depends upon the input-output effective distance characteristics of the code.

The second criterion is, at what error rate the curve meets the error floor of the code, which determines the lowest error rate that the code may achieve. The error floor depends upon the free distance of the code, which in turn is related to the interleaver the turbo code employs, as is discussed in Chapter 4.

Table 5.1 contains in octal notation codes identified through computer search, and optimized for normalized d_{s2}^{E} with the edge profile optimal [WKL97, WL00] constellation labelings illustrated in Fig. 5.1.

Our search over all interesting constituent convolutional encoders mapped onto the chosen labeling, in fact produces all interesting constituent codes that could be found with any other labeling related to the chosen one by a binary linear transformation. If two constellations are related with a linear transformation, an exhaustive search would lead to the same encoders with the linear transformation applied to their output. The set of labelings related to each other with a linear transformation is broad enough to include all common labelings. For example the Gray, Natural, and Reordered 8-PSK labeling reported in [BDM96], are included in such a set.

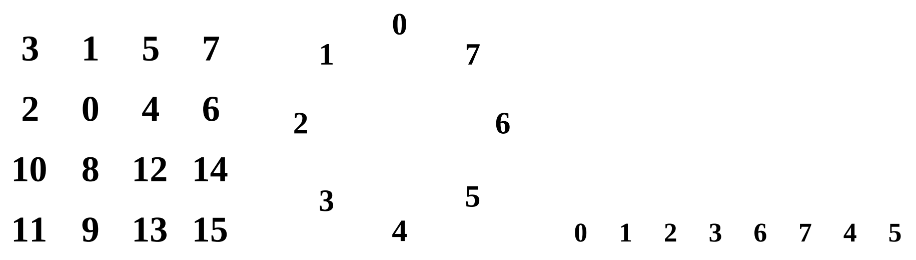
The encoders in Table 5.1 employ the single memory chain structure identified in Chapter 3. Each code has $k/2$ MSB inputs as systematic outputs, and r parity outputs. To describe a code we give in octal notation the feedback polynomial f, the k rows $\{\mathbf{b}_1 \dots \mathbf{b}_k\}$ of matrix $\mathbf{B}$, the r columns $\{\mathbf{c}_1 \dots \mathbf{c}_r\}$ of matrix $\mathbf{C}$ and the r columns $\{\mathbf{d}_1 \dots \mathbf{d}_r\}$ of matrix $\mathbf{D}$ that correspond to the r parity outputs. The simulated codes are shown in bold face. The search identified a large number of codes with the same value of d_{s2}^{E} and comparable number of nearest neighbors.

TABLE 5.1: CODES OPTIMIZED FOR d^E_{s2}

Codes optimized for d^E_{s2} for 16-QAM		
$\{f, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{b}_4, \mathbf{c}_1, \mathbf{c}_2, \mathbf{d}_1, \mathbf{d}_2\}$	d^E_{s2}	d^E_{s3}
{023,010,05,01,03, 04,017,016,015}	3(4)	1(1)
{023,010,05,01,03,04,06,016,011}	3(4)	2(2)
{031,010,011,013,017, 014,06,015,04}	3(5)	1(1)

Codes optimized for d^E_{s2} for 64-QAM=8 $\times$ 8-PAM		
$\{f,\mathbf{b}_1,\mathbf{b}_2,\mathbf{b}_3,\mathbf{b}_4,\mathbf{c}_1,\mathbf{d}_1\}$	d^E_{s2}	d^E_{s3}
{027,01,011,015,07,015,014}	0.3810(3)	0.1905(5)
{035,010,011,013,07,016,010}	0.3810(3)	0.1905(5)
{025,010,014,01,07,05,011}	0.3810(7)	0.3810(66)

Codes optimized for d^E_{s2} for 8 PSK		
$\{f,\mathbf{b}_1,\mathbf{b}_2,\mathbf{b}_3,\mathbf{b}_4,\mathbf{c}_1,\mathbf{d}_1\}$	d^E_{s2}	d^E_{s3}
{027,010,06,01,03,011,014}	1.171573(3)	0.585786(5)
{035,012,016,01,015,011,05}	1.171573(3)	0.585786(5)
{027,015,014,010,07,03,016}	1.171573(3)	0.585786(5)



(a) 16 QAM (b) 8 PSK (c) 8 PAM

FIGURE 5.1: LABELING FOR THE CONSTELLATIONS USED IN THE SIMULATIONS

.

The constituent encoders of rate greater than one (4/3 for example) are catastrophic, but the overall turbo encoder isn't. The upper constituent encoder (for example, in Fig. 3.1) has as systematic bits the $\frac{k}{2}$ MSB input bits, so in the catastrophic loops (i.e. nonzero-input, zero-output loops) only the $\frac{k}{2}$ LSB bits may be nonzero. Similarly the lower constituent encoder has as systematic bits the $\frac{k}{2}$ LSB input bits so in a catastrophic loop only the $\frac{k}{2}$ MSB input bits may be nonzero. Because the input symbols for the upper and lower constituent encoder catastrophic loops are different, the overall turbo encoder does not have an error event that involves catastrophic loops in both the constituent encoders, so the overall turbo encoder is not catastrophic. Each constituent encoder individually implies no coding gain; the coding gain of our system is provided by connecting two constituent encoders in the proposed turbo encoder structure.

The interleavers used in the simulations are uniform random or semi-random, as specified in each case. To describe an interleaver we give the constraint parameters described in Chapter 4 in the following order: (S, T, X), where $S_1 = S_2 = S$, $T_1 = T_2 = T$, and $X_1 = X_2 = X$. A random interleaver can be described by the spread parameters (0, 0, 0).

5.1 2 bits/sec/Hz PCTCM with 16 QAM

For 2 bits/sec/Hz PCTCM with 16 QAM, the constituent encoders implement 4/4 codes with $r = 2$ parity and $\frac{k}{2} = 2$ systematic outputs, and have $m = 4$ memory elements. The whole turbo encoder is depicted in Fig. 3.1. The simulated code is in the first row of Table 5.1.

For interleaver length $8,192$ symbols (input block size in bits: $8,192 \times 4$), the performance is within 0.5 dB of constrained capacity at BER 10^{-5} (Fig. 5.2) with a semi-random interleaver (30-0-0). The same figure also plots the bit-interleaved performance published in [DP95a] for 2 bits/sec/Hz PCTCM with 16 QAM, constituent encoders with 4 memory elements, interleaver length $16,384$ symbols, and input block size in bits: $16,384 \times 2$. We compare systems that do not have the same interleaver length (or interleaver gain), but have the same input block size in bits, which we believe is more interesting from a designer's

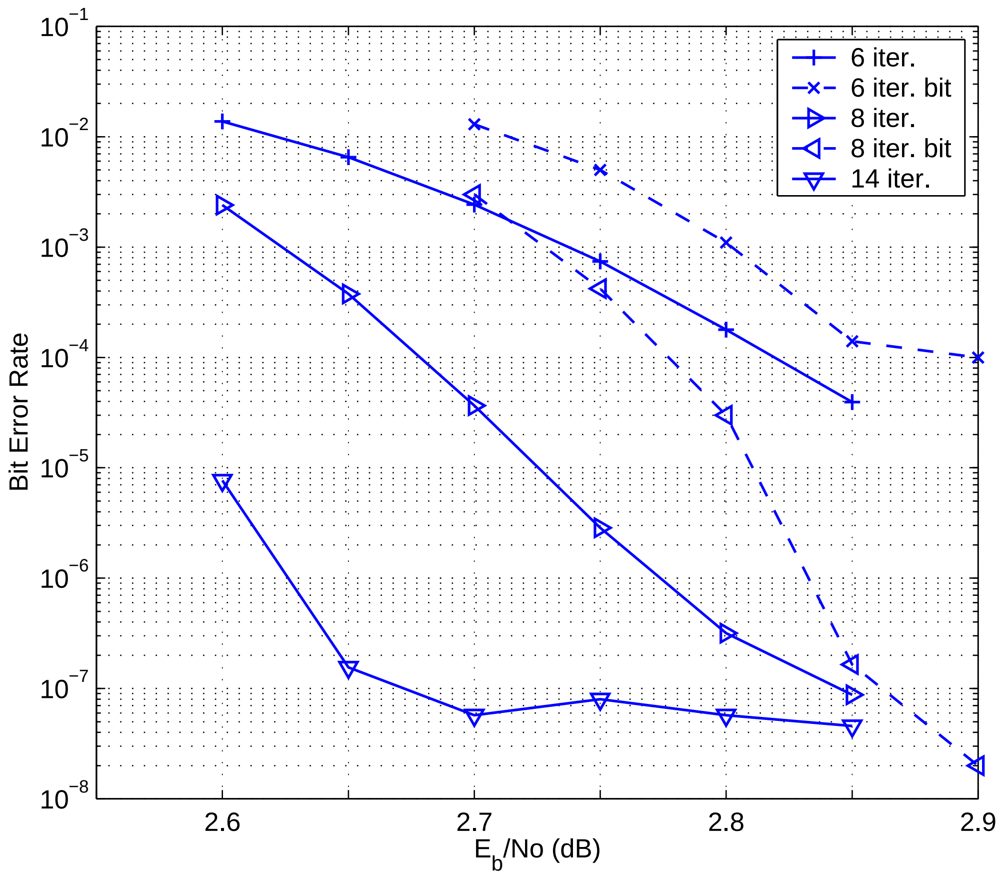


FIGURE 5.2: 2 BITS/SEC/HZ/ TURBO CODE EMPLOYING 16 QAM. CAPACITY=1.76 DB. CONSTRAINED CAPACITY=2.1 DB. INTERLEAVER LENGTH=$8,192$ SYMBOLS. INPUT BLOCK SIZE $8,192 \times 4$ BITS

point of view.

The proposed symbol interleaved system can converge 0.05 dB earlier with 6 decoder iterations, and 0.1 dB earlier with 8 decoder iterations, but has an error floor at around 5×10^{-8} which seems higher than the error floor in [DP95a]. This may be expected when comparing a symbol to a bit interleaved system, assuming both are optimized for d_{free}. The smaller available symbol-wise distance, leads to a smaller achievable d_{free} for the symbol-interleaved system, and thus a higher error floor. The higher error floor becomes more apparent for smaller interleaver lengths.

Fig. 5.2 also shows that the proposed system can converge 0.2 dB earlier by increasing the number of decoder iterations. This property may be desirable for such applications as deep-space communications, where the gain in dB can make worthwhile the extra complexity and time delay that increased decoder iterations involve.

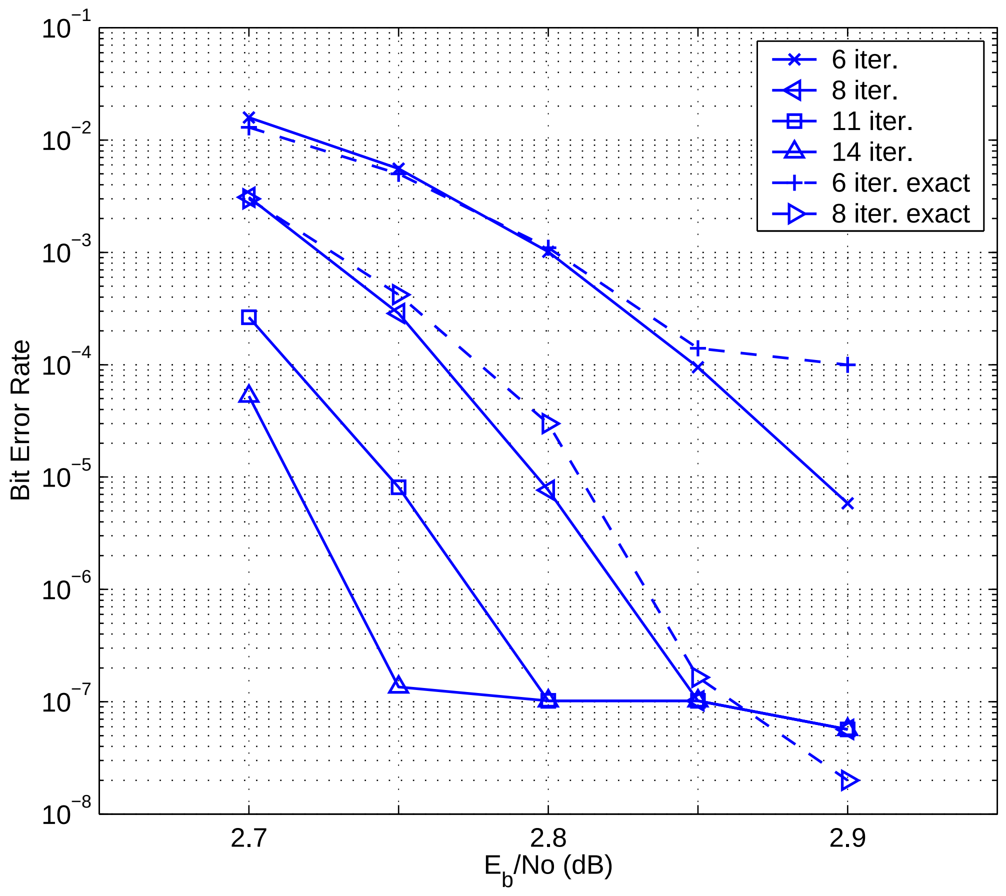


FIGURE 5.3: 2 BITS/SEC/HZ/ TURBO CODE EMPLOYING 16 QAM. REPRODUCTION OF [DP95A] RESULTS. INTERLEAVER LENGTH=16,384 SYMBOLS. INTERLEAVER (30-0-0). INPUT BLOCK SIZE $16,384 \times 2$ BITS

To study the behavior of the system in [DP95a] when increasing the number of decoder iterations, Fig. 5.3 reproduces the curves published in [DP95a] according to the specifications therein. The dashed line plots the published curves (denoted “exact” in the legend), while the solid lines plot the reproduced curves.

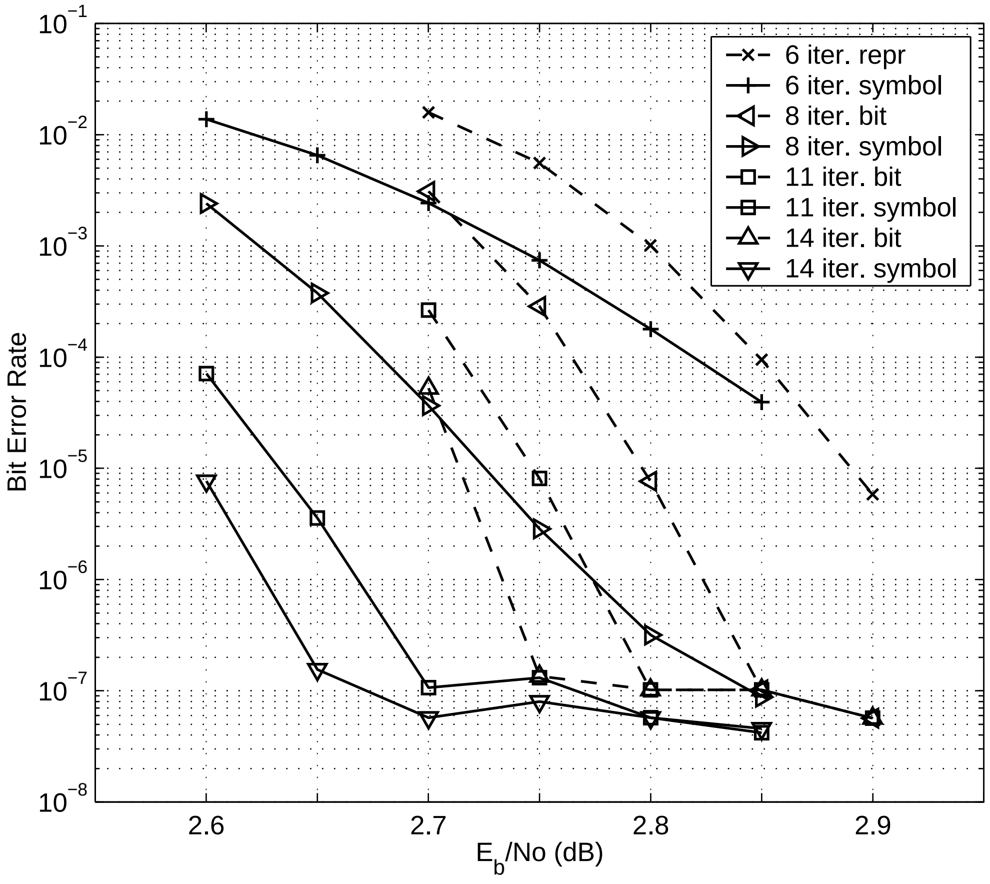


FIGURE 5.4: 2 BITS/SEC/HZ/ TURBO CODE EMPLOYING 16 QAM. COMPARISON OF THE PROPOSED SYSTEM PERFORMANCE, WITH THE SIMULATION REPRODUCED PERFORMANCE OF [DP95A]. BOTH SYSTEMS HAVE THE SAME INPUT BLOCK SIZE IN BITS $16,384 \times 2 = 8,192 \times 4$

Fig. 5.4 compares the proposed symbol interleaved system with the simulated bit interleaved system [DP95a] for different number of decoder iterations. The proposed system can converge around 0.1 dB earlier for 14 decoder iterations. The error floor seems to be the same for both systems.

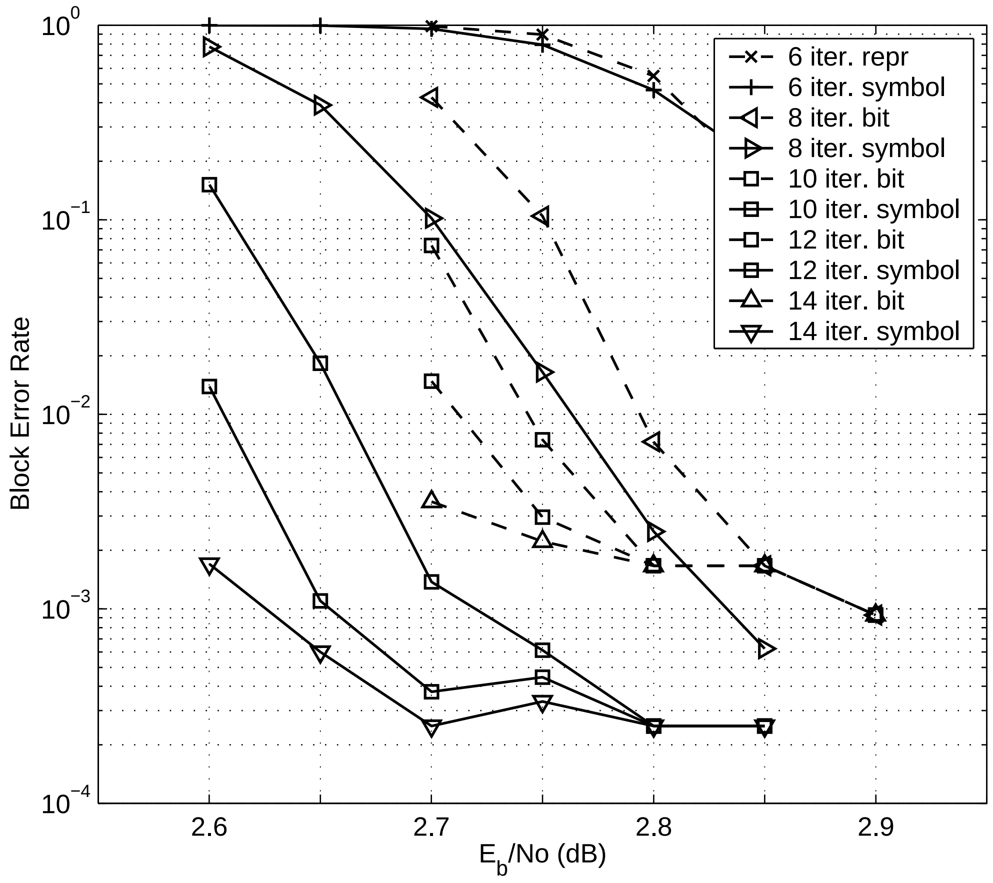


Figure 5.5: 2 bits/sec/Hz/ turbo code employing 16 QAM. Comparison of the Block Error Rate performance of the proposed system and the simulation reproduced system in [DP95a]. Both systems have the same input block size in bits $16,384 \times 2 = 8,192 \times 4$

Fig. 5.5 plots the Block Error Rate performance of the proposed symbol interleaved system. For comparison the same figure also plots the block error rate of the reproduced bit interleaved system in [DP95a]. The symbol interleaved system has a lower block error floor by almost an order of magnitude, which implies that fewer blocks fail, but when they do, they have more bit errors. We believe this difference results from the constituent encoders employed, and is not a general property of the symbol interleaved systems. Indeed, for 8-PAM (see Section 5.3) the symbol interleaved system has a higher block error floor.

5.2 2 bits/sec/Hz PCTCM with 8 PSK

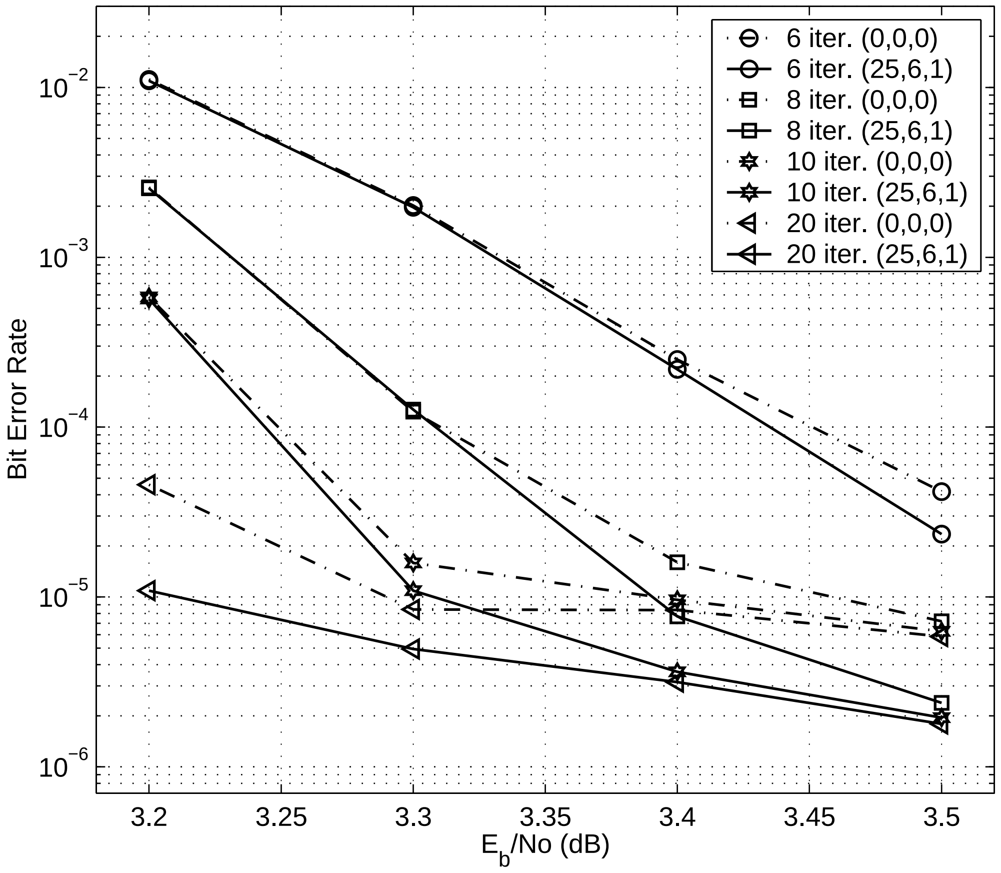


FIGURE 5.6: 2 BITS/SEC/HZ/ TURBO CODE EMPLOYING 8 PSK. CAPACITY=1.76 dB. CONSTRAINED CAPACITY=2.8 dB. INTERLEAVER LENGTH=$5,000$ SYMBOLS. INPUT BLOCK SIZE $5,000 \times 4$ BITS. THE INTERLEAVER (25,6,1) LOWERS THE ERROR FLOOR OF THE RANDOM INTERLEAVER

For 2 bits/sec/Hz PCTCM with 8 PSK, the constituent encoders implement a 4/3 code with $r = 1$ parity and $\frac{k}{2} = 2$ systematic outputs and have $m = 4$ memory elements. The whole turbo encoder is depicted in Fig. 3.2.

For interleaver length $5,000$ symbols (input block size in bits: $5,000 \times 4$), the performance is within 0.6 dB of constrained capacity at BER=10^{-5} (Fig. 5.6). The interleaver (25,6,1) lowers the error floor of the random (0,0,0) interleaver. Fig. 5.7 plots the Block Error Rate performance corresponding to the bit error rate curves in Fig. 5.6. The designed interleaver also lowers the block error floor.

Fig. 5.8 plots the performance for interleaver length $2,500$ symbols (input block size in bits: $2,500 \times 4$), with a random (0,0,0) interleaver. The same figure plots the performance of the symbol interleaved system in [RW96] for interleaver length $5,000$ symbols, input block size in bits: $5,000 \times 2$, 2 bits/sec/Hz PCTCM with 8 PSK, but constituent encoders with $m = 3$ memory elements which leads to roughly half the decoder complexity. The proposed system can converge up to 0.25 dB earlier.

The random interleaver causes a high error floor, which can be lowered by using a more elaborate interleaver. Figures 5.9 and 5.10 show that the interleaver (20,4,1) lowers both the bit error rate and block error rate error floor compared to the random interleaver.

The search for 8 PSK constituent encoders optimized for d_{s2}^{E} produced around $9,500$ encoders with the same value of d_{s2}^{E}. The simulated encoder in Figures 5.8, 5.9 and 5.10 was chosen among them to have the smaller number of d_{s2}^{E} nearest neighbors possible. To study the achievable free distance with this encoder, table 5.2 depicts the smallest output Euclidean weight associated with error events of symbol wise input weight two, and error event length L, denoted as $d_{s2}^{L=l}$.

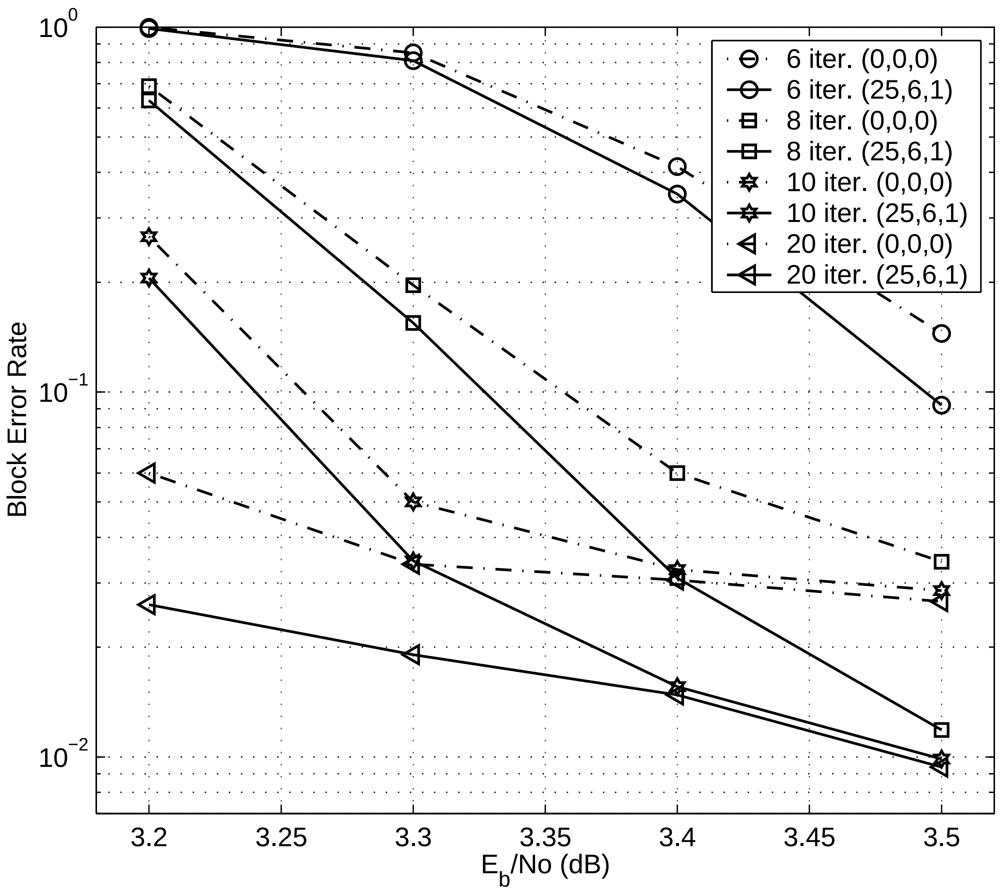


FIGURE 5.7: 2 BITS/SEC/Hz/ TURBO CODE EMPLOYING 8 PSK. CAPACITY=1.76 dB. CONSTRAINED CAPACITY=2.8 dB. INTERLEAVER LENGTH=$5,000$ SYMBOLS. INPUT BLOCK SIZE $5,000 \times 4$ BITS. THE INTERLEAVER (25,6,1) LOWERS THE ERROR FLOOR OF THE RANDOM INTERLEAVER

Observe that $d_{s2}^{L \geq 20} = 6.83$, the output Euclidean weight does not increase with the error event length after length $L = 20$. This implies that the highest output free distance associated with S-type error events (Fig. 4.3) is bounded by $6.83 + 1.17 = 8$ (one error event with length $L = 2$ and one error event with $L \geq 20$). Increasing the S parameter value to more than 20 does not offer any improvement. Moreover this distance upper bounds the output distance of all

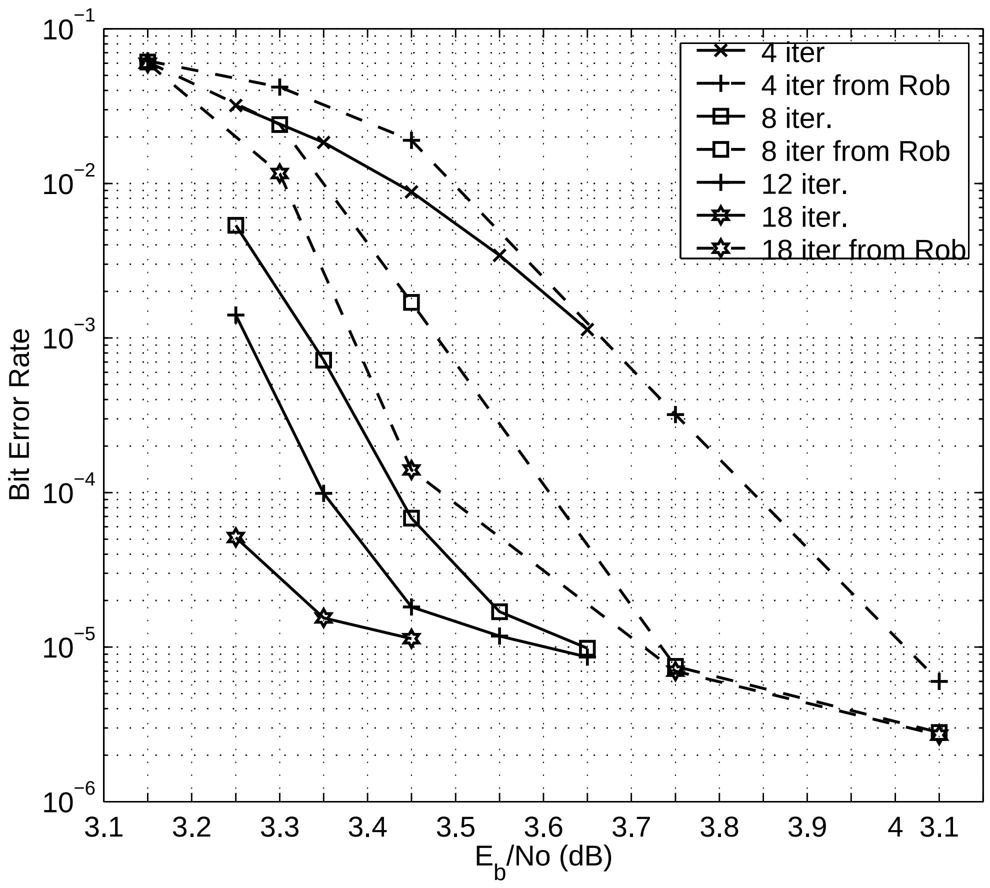


FIGURE 5.8: 2 BITS/SEC/HZ/ TURBO CODE EMPLOYING 8 PSK. CAPACITY=1.76 DB. CONSTRAINED CAPACITY=2.8 DB. INTERLEAVER LENGTH=$2,500$ SYMBOLS. INPUT BLOCK SIZE $2,500 \times 4$ BITS

error events of S type that may occur during decoding. This does not seem a desirable property.

For T type error events (Fig. 4.4) , to achieve free distance equal to 8 (the free distance achieved by S type events), the interleaver needs to satisfy the second constraint with value at least $T = 12$, since $8 - 3 \times 1.17 = 4.49$ (three error events of length two, that correspond to output weight 1.17, and one error event that corresponds to output weight 4.49). Error events of length smaller than 13

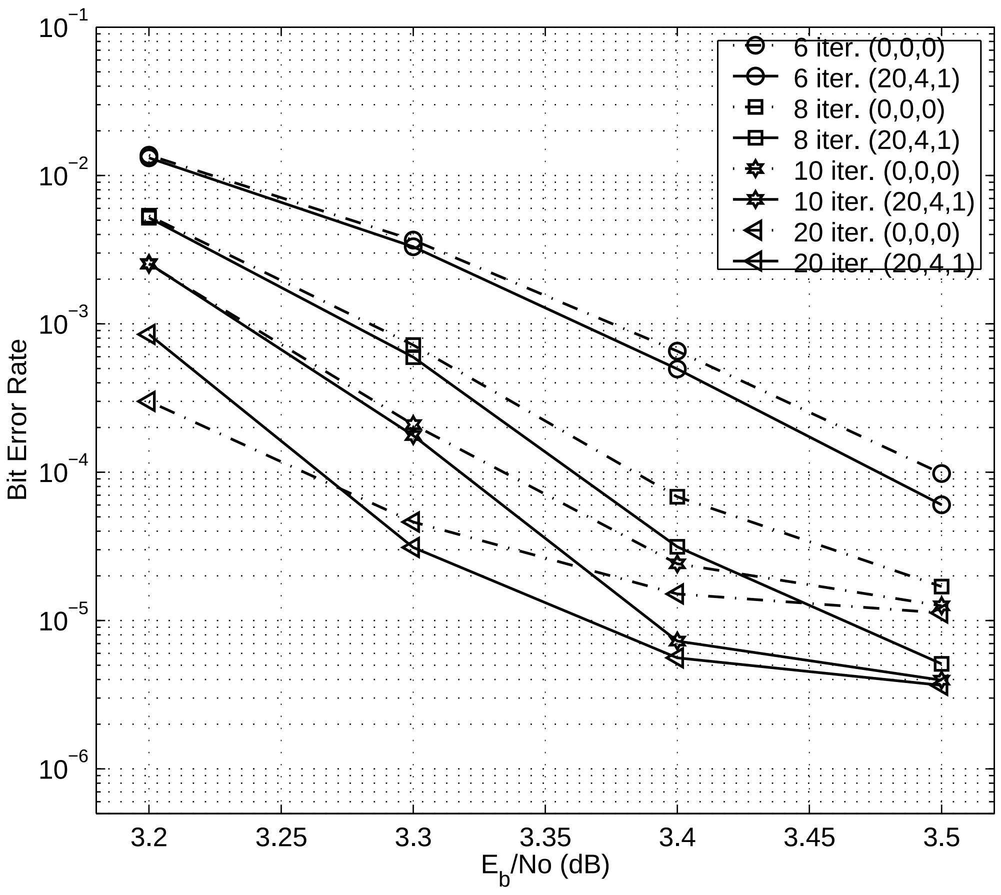


FIGURE 5.9: 2 BITS/SEC/Hz/ TURBO CODE EMPLOYING 8 PSK. CAPACITY=1.76 dB. CONSTRAINED CAPACITY=2.8 dB. INTERLEAVER LENGTH=$2,500$ SYMBOLS. INPUT BLOCK SIZE $2,500 \times 4$ BITS. THE INTERLEAVER (20,4,1) LOWERS THE ERROR FLOOR OF THE RANDOM INTERLEAVER

have associated output weight smaller than 4.49. But, for IL=$2,500$, there does not exist an interleaver with T parameter equal to $T = 12$. For $T = 4$, the minimum output Euclidean distance associated with T type error events is equal to: $1.76 + 3 \times 1.17 = 5.27$. The X type error events (Fig. 4.4) have distance at least $1.17 \times 6 = 7.03$ (six error events of length two). Thus for this encoder, and the (20, 4, 1) interleaver, the T type error event upper bound the free distance

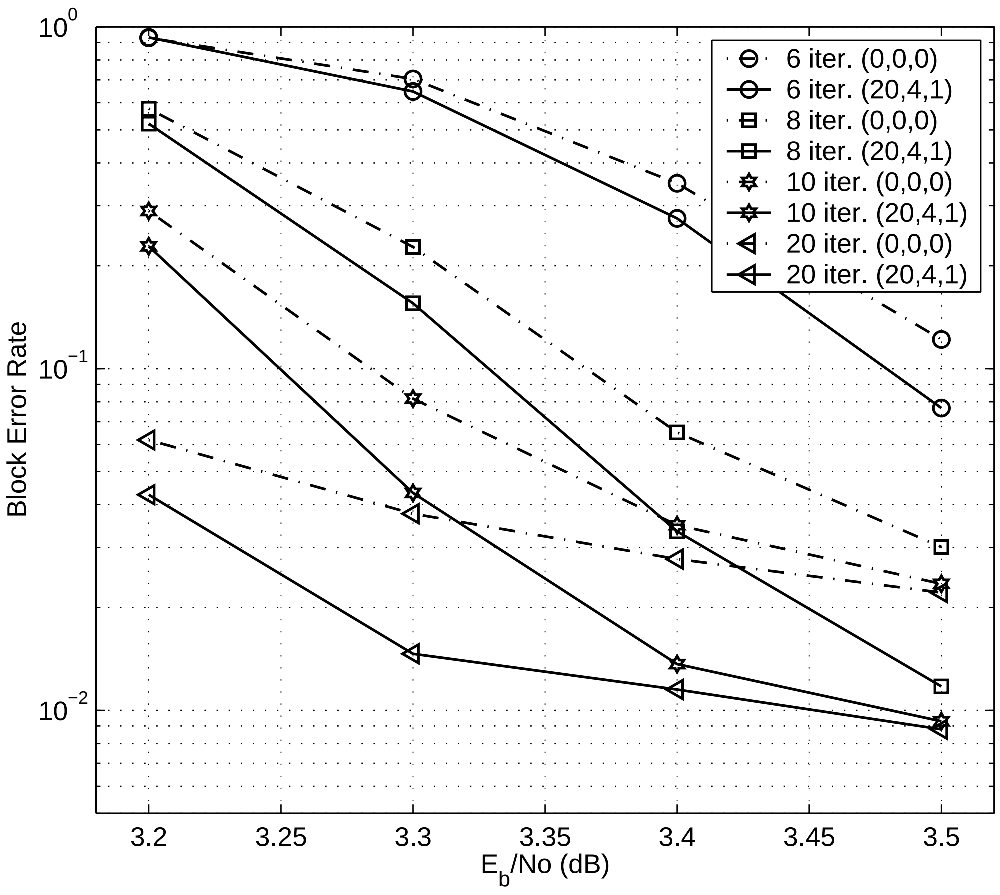


FIGURE 5.10: 2 BITS/SEC/HZ/ TURBO CODE EMPLOYING 8 PSK. CAPACITY=1.76 DB. CONSTRAINED CAPACITY=2.8 DB. INTERLEAVER LENGTH=$2,500$ SYMBOLS. INPUT BLOCK SIZE $2,500 \times 4$ BITS. THE INTERLEAVER (20,4,1) LOWERS THE BLOCK ERROR FLOOR OF THE RANDOM INTERLEAVER

to be at most 5.27.

To overcome this limitation, instead of designing an interleaver to fit an encoder, we followed the reverse procedure. We initially identified an interleaver of length $2,500$ with the higher values of S and T parameters possible. This was an (20-5-0) interleaver. Then we performed a search among the $9,500$ identified

L	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	35
d_{s2}^L	1.17(1)	1.17(1)	1.17(1)	1.76(3)	2.93(1)	2.34(2)	2.34(2)	3.51(1)	3.51(1)	3.51(1)	4.10(3)	5.27(1)	4.69(2)	4.69(2)	5.86(1)	5.86(1)	5.86(1)	6.44(3)	6.83(1)	6.83(1)	6.83(1)	6.83(1)	6.83(1)	6.83(1)	6.83(1)	6.83(1)	6.83(1)	6.83(1)	6.83(1)	6.83(1)

Table 5.2: Squared Euclidean Distance for Error Events for the 8 PSK encoder 027, 010, 06, 01, 03, 011, 014. The nearest neighbors are given in parentheses. L denotes the error event length. d_{s2}^L denotes the smallest output Euclidean distance associated with error events of length L and symbol-wise input weight two.

8 PSK encoders that achieve the highest d_{s2}^E value, with the following criteria:

1. The output Euclidean weight of error events of length two or higher to be as large as possible (useful for all S, T and X error events).

2. The minimum output Euclidean weight of error events of length $T+1$ or higher to be as large as possible (useful for T error events).

3. The minimum output Euclidean weight of error events of length $S+1$ or higher to be as large as possible (useful for S error events).

4. The output weight to increase with the error event length.

Among the good codes the search identified, we simulated the code described by the encoder polynomials: {031, 01, 05, 013, 011, 015, 017}. Table 5.3, similarly to Table 5.2, depicts for this new code the smallest output Euclidean weight associated with error events of symbol wise input weight two, and error event length L, denoted as $d_{s2}^{L=l}$. The output weight increases with the error event length. The free distance achievable with the (20, 5, 0) interleaver is equal to 5.85 ($1.17 + 7.03 = 8.2$ for S events, $3 \times 1.17 + 2.34 = 5.85$ for T events and $3 \times 1.17 = 7.02$ for X events).

L	2	3	4	5	6	7	8	9	10	11
d_{s2}^L	1.17(3)	1.17(2)	1.76(2)	1.76(2)	2.34(1)	2.34(1)	2.34(1)	2.92(1)	3.51(2)	3.51(1)
L	12	13	14	15	16	17	18	19	20	21
d_{s2}^L	3.51(1)	4.10(1)	4.69(2)	4.69(1)	4.69(1)	5.86(3)	5.86(2)	6.44(2)	6.44(2)	7.03(1)
L	22	23	24	25	26	27	28	29	30	35
d_{s2}^L	7.03(1)	7.03(1)	7.62(1)	8.20(2)	8.20(1)	8.20(1)	8.78(1)	9.37(2)	9.37(2)	11.13(2)

TABLE 5.3: SQUARED EUCLIDEAN DISTANCE FOR ERROR EVENTS FOR THE 8-PSK ENCODER {031, 01, 05, 013, 011, 015, 017}. THE NEAREST NEIGHBORS ARE GIVEN IN PARENTHESES. L DENOTES THE ERROR EVENT LENGTH. d_{s2}^L DENOTES THE SMALLEST OUTPUT EUCLIDEAN DISTANCE ASSOCIATED WITH ERROR EVENTS OF LENGTH L AND SYMBOL-WISE INPUT WEIGHT TWO.

This encoder is specifically designed to perform well in conjunction with an (20, 5, 0) interleaver. Fig. 5.11 plots the bit error rate for both the simulated 8 PSK encoders. By choosing the appropriate encoder, the designer has the choice either to achieve convergence at a lower SNR, or to have a lower error floor. Fig. 5.12 compares the block error rate of the two designed encoders.

Fig. 5.13 plots the bit error rate performance for the new encoder {031, 01, 05, 013, 011, 015, 017} when used with a random (0,0,0) and the designed (20,5,0) interleaver. Fig. 5.14 plots the corresponding block error rate performance.

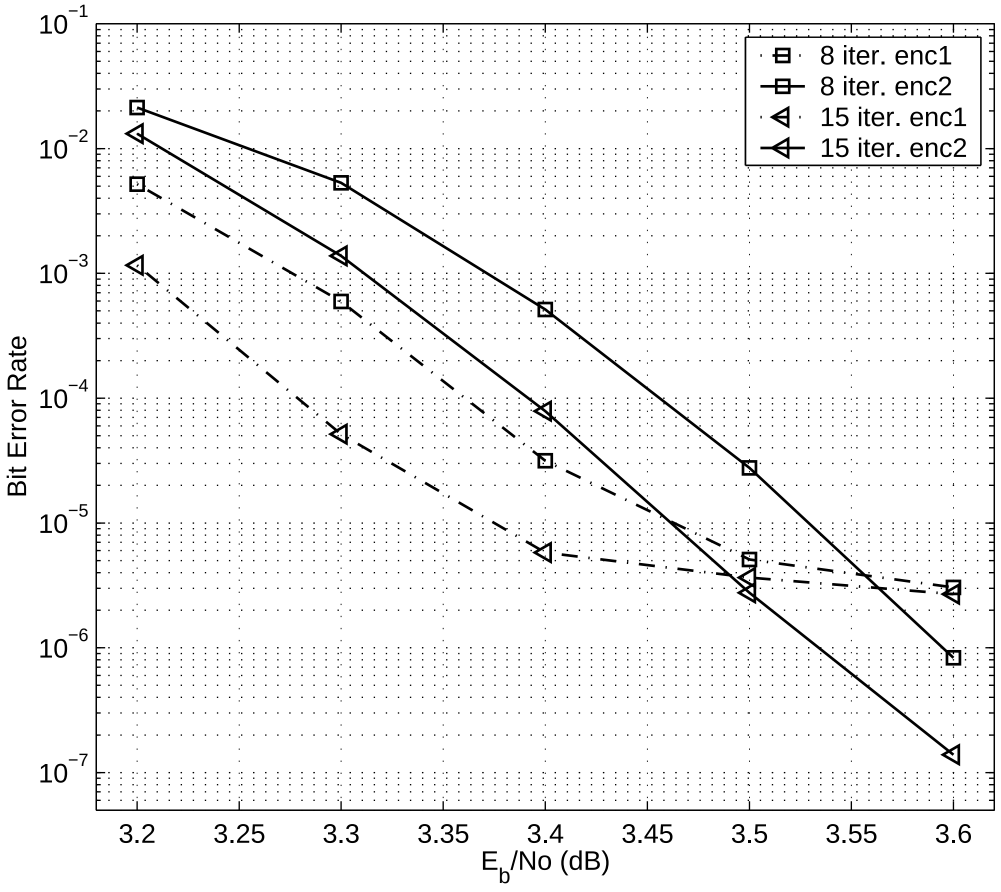


FIGURE 5.11: 2 BITS/SEC/HZ/ TURBO CODE EMPLOYING 8 PSK. COMPARISON OF THE 8-PSK ENCODERS: ENC1= {027, 010, 06, 01, 03, 011, 014} EMPLOYING AN (20, 4, 1) INTERLEAVER AND ENC2={031, 01, 05, 013, 011, 015, 017} EMPLOYING AN (20, 5, 0) INTERLEAVER). INTERLEAVER LENGTH=$2,500$ SYMBOLS. INPUT BLOCK SIZE $2,500 \times 4$ BITS

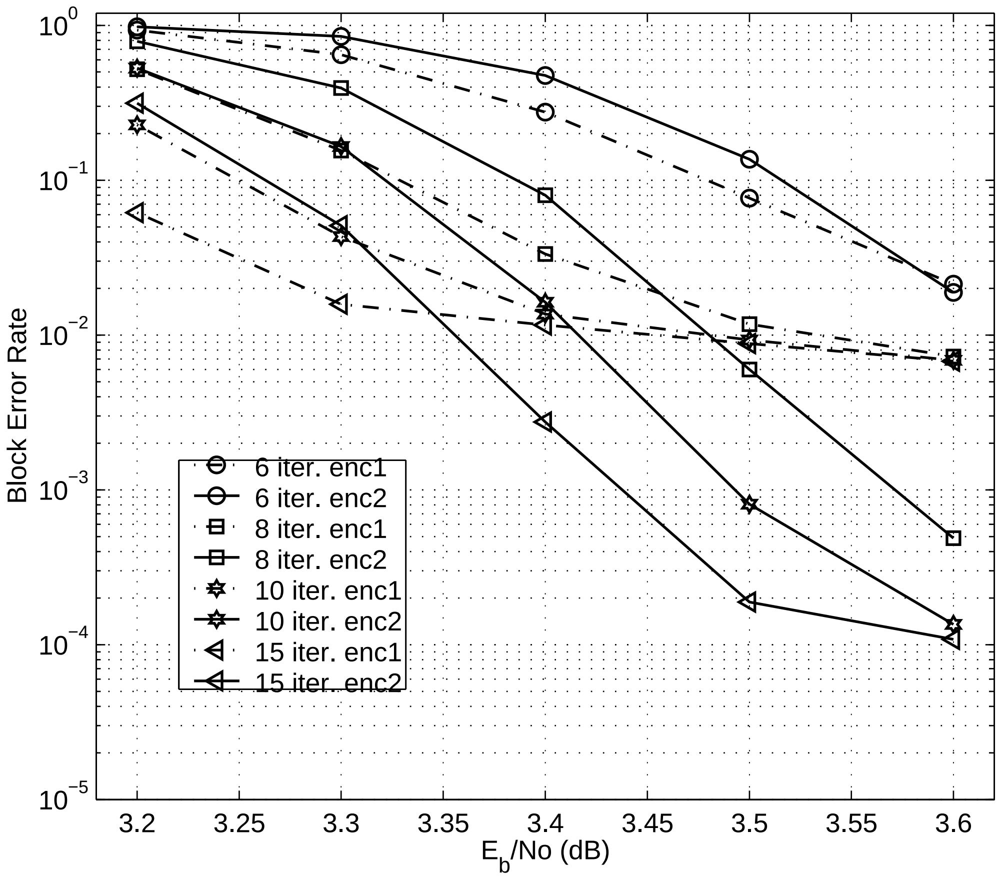


FIGURE 5.12: 2 BITS/SEC/HZ/ TURBO CODE EMPLOYING 8 PSK. BLOCK ERROR RATE FOR TWO 8 PSK ENCODERS ENC1={027, 010,-06, 01, 03, 011, 014} EMPLOYING AN (20, 4, 1) INTERLEAVER AND ENC2={031, 01, 05, 013, 011, 015, 017} EMPLOYING AN (20, 5, 0) INTERLEAVER). INTERLEAVER LENGTH=2,500 SYMBOLS. INPUT BLOCK SIZE $2,500 \times 4$ BITS

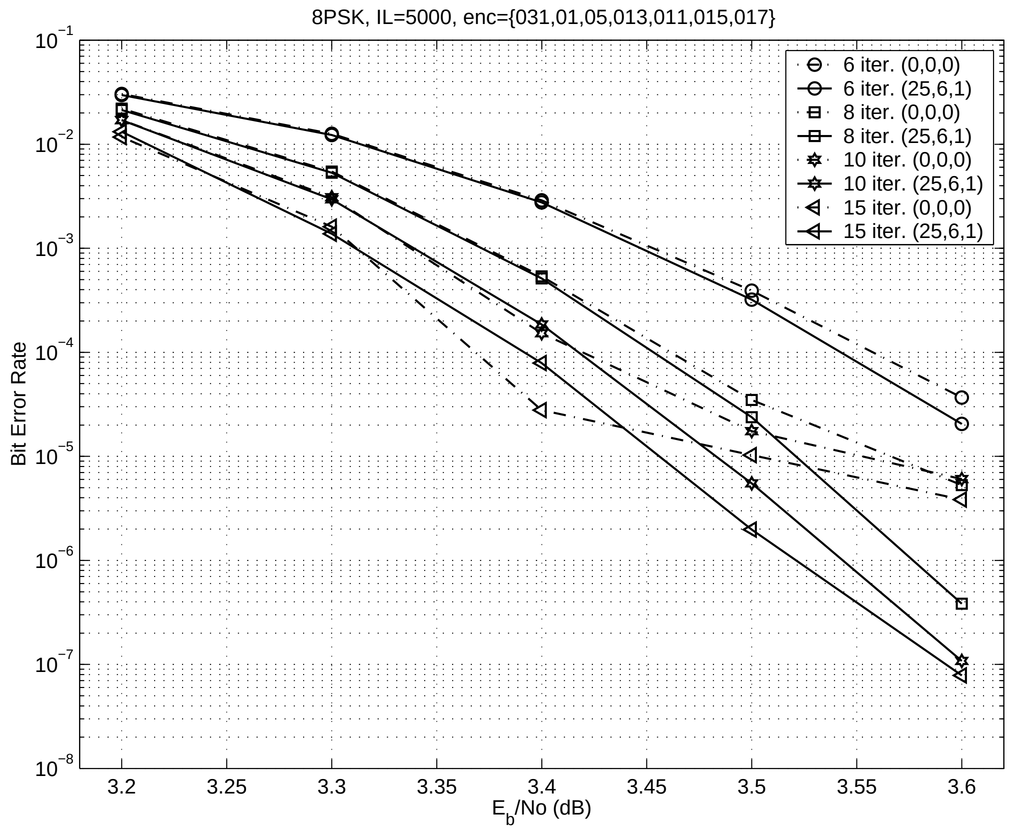


FIGURE 5.13: 2 BITS/SEC/HZ/ TURBO CODE EMPLOYING 8 PSK. ENCODER EMPLOYED: ENC2={031, 01, 05, 013, 011, 015, 017}. INPUT BLOCK SIZE $2,500 \times 4$ BITS. THE INTERLEAVER (20,5,0) LOWERS THE ERROR FLOOR OF THE RANDOM (0,0,0) INTERLEAVER

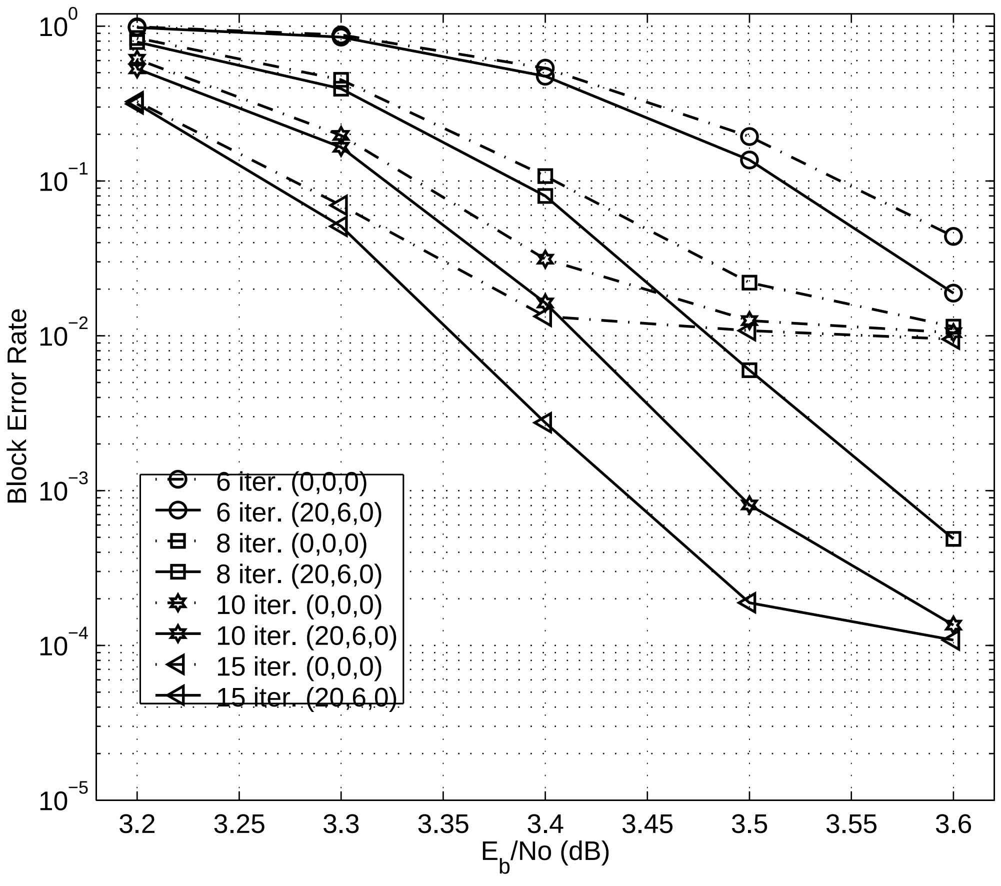


FIGURE 5.14: 2 BITS/SEC/Hz/ TURBO CODE EMPLOYING 8 PSK. ENCODER EMPLOYED: ENC2={031, 01, 05, 013, 011, 015, 017}. INPUT BLOCK SIZE $2,500\times4$ BITS. THE INTERLEAVER (20,5,0) LOWERS THE BLOCK ERROR FLOOR OF THE RANDOM INTERLEAVER

5.3 4 bits/sec/Hz PCTCM with 8×8 PAM=64 QAM

For 4 bits/sec/Hz PCTCM with 8×8 PAM=64 QAM (Fig. 5.15), the constituent encoders implement a 4/3 code with $r = 1$ parity and $\frac{k}{2} = 2$ systematic outputs, and have $m = 4$ memory elements.

For interleaver length $4,096$ symbols (input block size in bits: $4,096 \times 4$), and (30, 0, 0) interleaver, the performance at BER 10^{-6} is within 0.6 dB of constrained capacity. Compared with the bit-interleaved system performance in

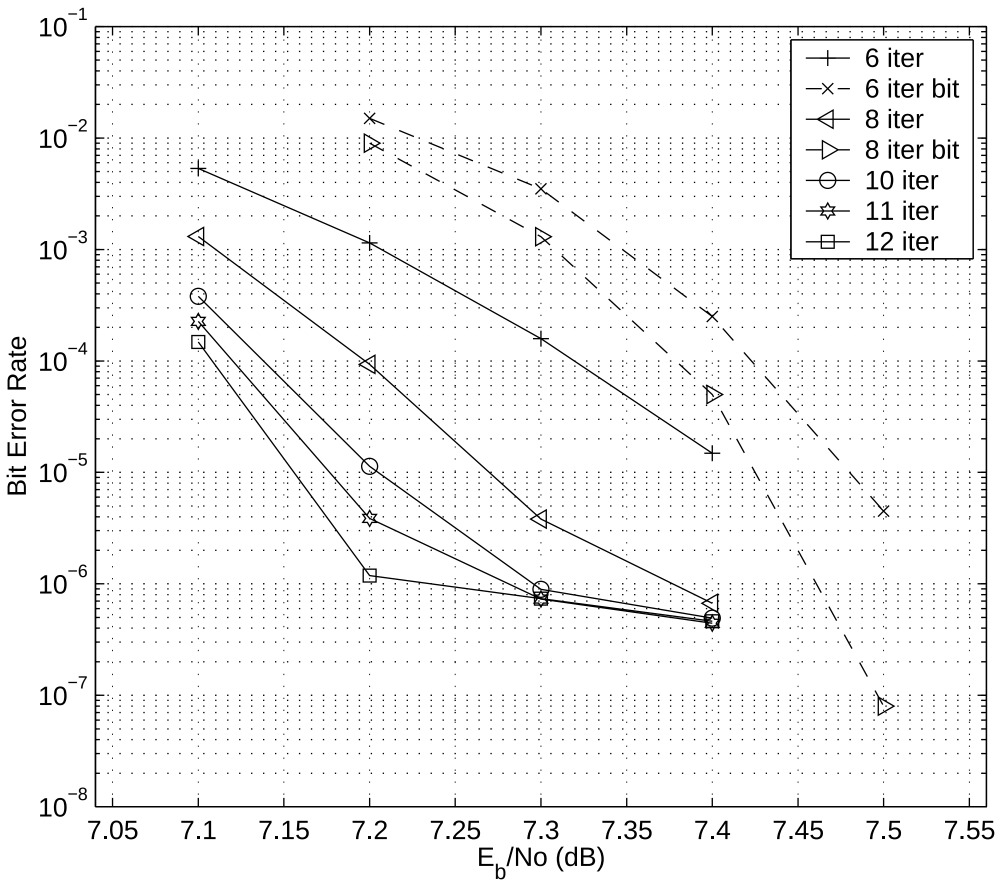


FIGURE 5.15: 4 BITS/SEC/HZ/ TURBO CODE EMPLOYING 8 PAM. CAPACITY 5.74 DB. CONSTRAINED CAPACITY 6.6 DB. INTERLEAVER LENGTH $4,096$ SYMBOLS. INPUT BLOCK SIZE $4,096 \times 4$ BITS

[BDM96] for 4 bits/sec/Hz PCTCM with 64 QAM, 4 memory elements, interleaver length $4,096$ symbols (input block size in bits: $4,096 \times 4$), and $(30-0-0)$ interleaver, the proposed symbol interleaved system can converge earlier (e.g., approximately 0.1 dB for 8 iterations) but has an error floor around an order of magnitude higher. The proposed system also has the property of converging at a lower SNR as the number of decoder iterations increase (up to a point).

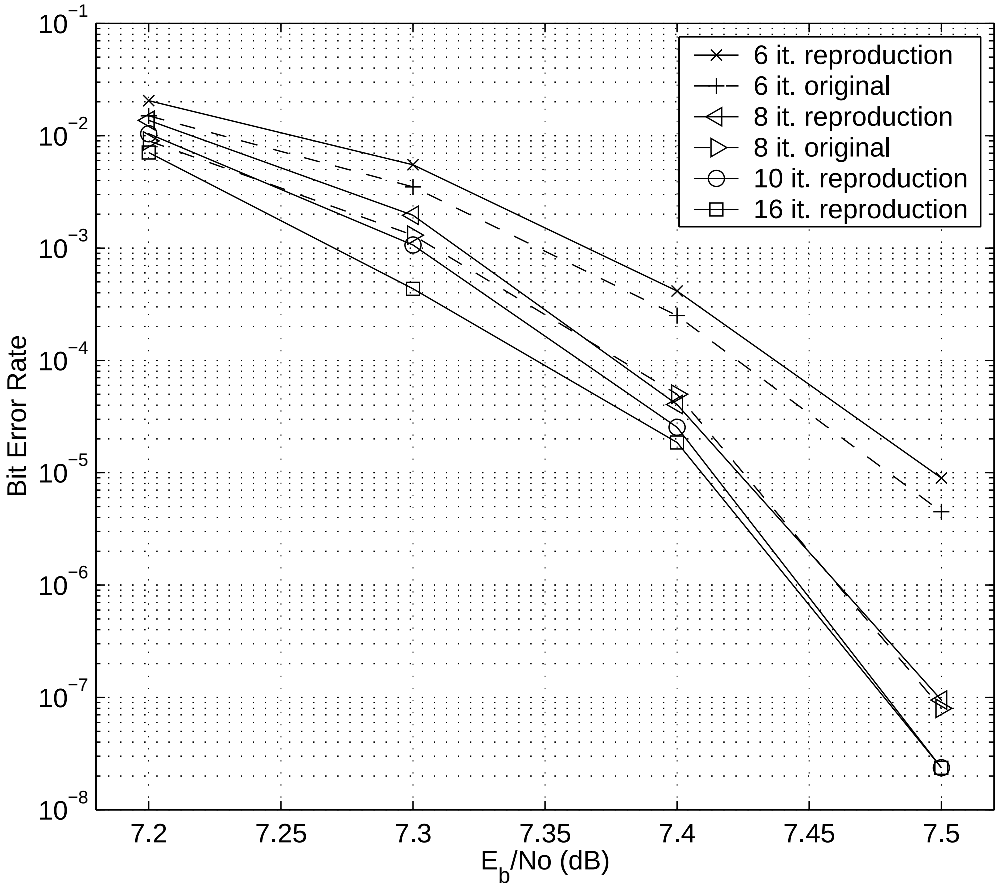


FIGURE 5.16: 4 BITS/SEC/HZ/ TURBO CODE EMPLOYING 8 PAM. SIMULATIONS REPRODUCTION OF BIT INTERLEAVED SYSTEM IN [BDM96]. INTERLEAVER LENGTH $4,096$ SYMBOLS. INPUT BLOCK SIZE $4,096 \times 4$ BITS

Fig. 5.16 reproduces the system in [BDM96] according to the specifications therein. The employed code does not seem to offer better performance when increasing the number of decoder iterations.

Fig. 5.17 plots the block error rate of the proposed system. The same figure also plots the block error rate of reproduced through simulations system in [BDM96]. The system in [BDM96] has a lower block error floor by an order of magnitude. To improve the error floor of the proposed system, an interleaver-encoder optimization procedure may follow the steps presented in detail for 8-PSK encoders in Section 5.2.

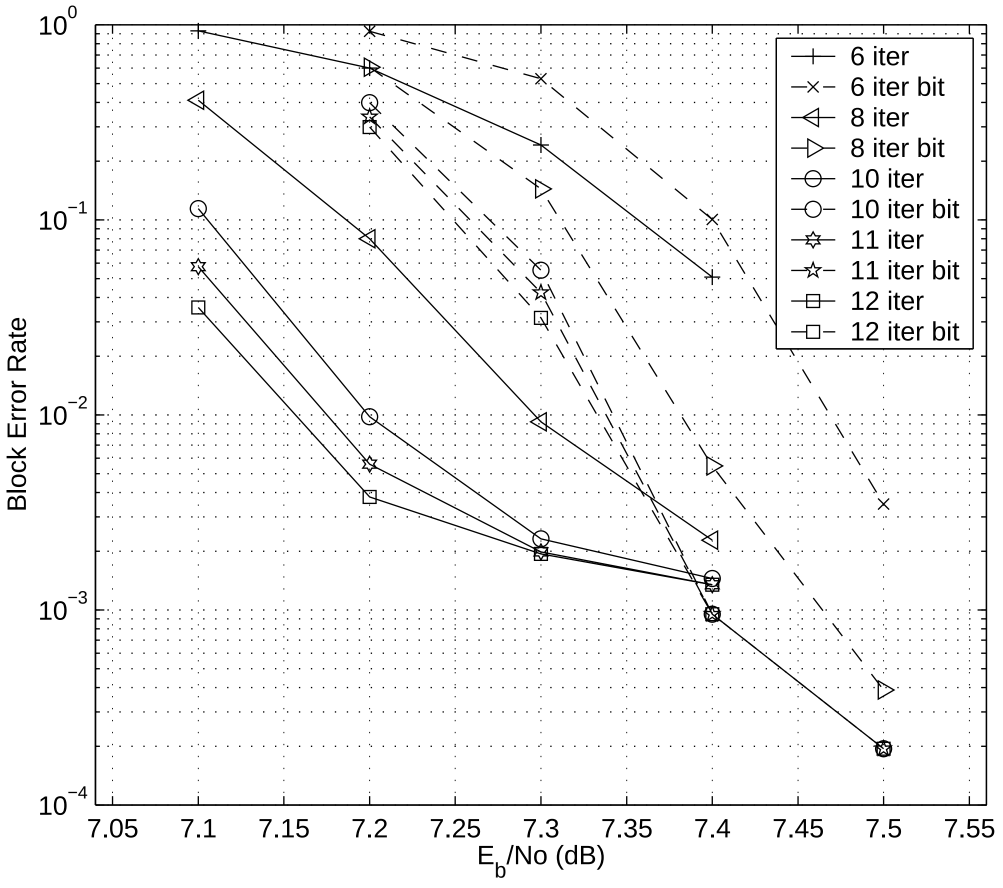


FIGURE 5.17: 4 BITS/SEC/Hz/ TURBO CODE EMPLOYING 8 PAM. BLOCK ERROR RATE PERFORMANCE OF THE PROPOSED SYSTEM AND THE SIMULATIONS REPRODUCED SYSTEM IN [BDM96]. INTERLEAVER LENGTH $4,096$ SYMBOLS. INPUT BLOCK SIZE $4,096 \times 4$ BITS

CHAPTER 6

Conclusions

This thesis examined a method to achieve high spectral efficiency using symbol interleaved parallel concatenated trellis coded modulation. The turbo encoder design consists of two parts, constituent encoder design and interleaver design, examined in Chapters 3 and 4 respectively.

Chapter 2 introduced the state-space representation of convolutional codes and the notion of algebraic equivalence. The rational form theorem shows that in order to examine all algebraically distinct encoders, it is sufficient to consider only the canonical memory structures with R rows. An algebraic criterion, derived from application of control theory notions to convolutional code design, allows to check minimality among range equivalent encoders. As an application, we proposed a method to check minimality under periodic puncturing, and assess the performance of non minimal encoders. Simulation results and code tables for $1/4$ codes optimized for Hamming weight under both bit-wise and symbol-wise puncturing are included.

The constituent encoder design (Chapter 3) determined the optimization criteria and extended the effective distance bound to symbol-wise inputs. Based on the framework developed in Chapter 2 we identified the canonical structures for the constituent encoder search space. In many cases of practical interest, the optimal structure for these constituent encoders connects the memory elements in a single row. This single row generally applies to turbo-code constituent encoders for parallel concatenation and is not restricted to symbol interleaving or trellis coded modulation. As an example, we presented codes optimized for bit-wise effective distance coupled with BPSK.

The interleaver design (Chapter 4) extended the spread interleaver design to take into account multiple error events, and proposed a semi-random interleaver construction method.

Simulation results (Chapter 5) for 2 bits/sec/Hz with 16-QAM, 2 bits/sec/Hz with 8-PSK, and 4 bits/sec/Hz with 64-QAM show that the proposed symbol interleaved system, as compared to the systems reported in the literature, can converge at lower SNR. To lower the error floor we proposed a two step method for interleaver - constituent encoder optimization (Chapter 5). Using the appropriate encoder optimization criteria allows to trade-off convergence at a lower SNR for a lower error floor.

References

[And97] Andersen J. " Interleaver design For turbo coding ." In *International Symposium on Turbo Codes*, 1997.

[Ash65] Ash R. *Information Theory.* Dover Publications, NY, 1965.

[BCT93] Berrou C., Clavieux A., and Thitimajshima P. "Near Shannon limit error-correcting coding: turbo codes ." In *IEEE International Conference on Communications*, pp. 1064–1070, Geneva, Switzerland, May 1993.

[BDM91] Biglieri E., Divsalar D., McLane P. J., and Simon M. *Introduction to Trellis-Coded Modulation.* Macmillan Publishing Company, New York, 1991.

[BDM96] Benedetto S., Divsalar D., Montorsi G., and Pollara F. "Parallel concatenated trellis coded modulation." In *IEEE International Conference on Communications*, volume 2, pp. 974–978, Dallas, TX, USA, June 1996.

[BDM97] Benedetto S., Divsalar D., Montorsi G., and Pollara F. "A soft-input soft-output APP module for the iterative decoding of concatenated codes." *IEEE Communications Letters*, **1**(1):22–24, January 1997.

[BGM98] Benedetto S., Garello R., and Montorsi G. "A search for good convolutional codes to be used in the construction of turbo codes." *IEEE Transactions on Communications*, **46**(9):1101–1105, September 1998.

[BM96] Benedetto S. and Montorsi G. "Unveiling turbo codes:some results on parallel concatenated coding schemes." In *IEEE Transactions on Info. Theory*, volume 42, pp. 409–429, March 1996.

[BP94] Barbulescu A. S. and Pietrobon S. S. "Interleaver design for turbo codes." In *Electronics Letters*, volume 30, pp. 2107–2108, 8 December 1994.

[Che99] Chen C. "Linear System Theory and Design." *Oxford University Press*, 1999.

[CT91] Cover T. M. and Thomas J. A. *Elements of Information Theory.* Wiley Series in Telecommunications, USA, 1991.

[DBP97] Divsalar D., Benedetto S., Pollara F., and Montorsi G. " Turbo codes: principles and applications." In *Lecture Notes*, UCLA EXTENSION, Octomber 1997.

[DM96] Divsalar D. and McEliece R. J. "Effective free distance of turbo codes ." *Electronics Letters*, **32**(5):445–446, 29 February 1996.

[DM97] Daneshgaran F. and Mondin M. "Design of interleavers for turbo codes based on a cost function." In *International Symposium on Turbo Codes*, 1997.

[DM98] Divsalar D. and McEliece R. J. "On the design of generalized concatenated coding systems with interleavers." In *TMO PR 42-134*, pp. 1–22, April-June 1998.

[DP95a] Divsalar D. and Pollara F. "Multiple turbo codes." In *MILCOM 95 Universal Communications*, volume 1, pp. 279–285, San Diego, CA, November 1995.

[DP95b] Divsalar D. and Pollara F. "On the design of turbo codes ." In *TMO PR 42-123*, pp. 99–121, July-September 1995.

[FJW96] Forney G. D., Johannesson R., and Wan Z. "Minimal and canonical rational generator matrices for convolutional codes." *IEEE Transactions on Info. Theory*, **42**(6):1865–1880, November 1996.

[For70] Forney G. D. "Convolutional codes I: algebraic structure." *IEEE Transactions on Info. Theory*, **26**(6):720–738, November 1970.

[FW99] Fragouli C. and Wesel R. "Convolutional codes and control theory." In *ComCon99*, Athens, Greece, June 1999.

[FW00] Fragouli C. and Wesel R. D. "Turbo encoder design for high spectral efficiency." In *IEEE Transactions on Communications*, Accepted, 2000.

[Gal68] Gallager R. G. *Information Theory and Reliable Communications.* Wiley Series in Telecommunications, USA, 1968.

[HJ85] Horn R. A. and Johnson C. R,. "Matrix Analysis." *Cambridge University Press*, 1985.

[HM97] Hokfelt J. and Maseng T. " Methodical Interleaver Design For Turbo Codes ." In *International Symposium on Turbo Codes*, 1997.

[J 95] J. Proakis . *Digital Communications.* McGraw-Hill, Third Edition, 1995.

[JW93] Johannesson R. and Wan Z. "A linear algebra approah to minimal convolutional encoders." *IEEE Transactions on Info. Theory*, **39**(4):1219–1233, July 1993.

[Lap94] Lapidoth A. "The performance of convolutional codes on the block erasure channel using various finite interleaving techniques." *IEEE Transactions on Info. Theory*, **40**:1459–1473, September 1994.

[LFM94] Loeliger H., Forney G. D., Mittelholzer T., and Trott M. D. "Minimality and observability of group systems." *Linear Algebra And Its Applications*, **205-206**:937–963, July 1994.

[OS97] Oberg M. and Siegel P. " Lowering the error floor for turbo codes ." In *International Symposium on Turbo Codes*, 1997.

[OY98] Ogiwara H. and Yano M. "Improvement of turbo trellis-coded modulation system." *IEICE*, **E81-A**(10):2040–2046, Oct 1998.

[PSC96] Perez L., Seghers J., and Costello D. " A distance spectrum interpretation of turbo codes." In *IEEE Transactions on Info. Theory*, November 1996.

[RW96] Robertson P. and Worz T. "A novel bandwidth efficient coding scheme employing turbo codes." In *IEEE International Conference on Communications*, volume 2, pp. 962–967, Dallas, TX, USA, June 1996.

[RW98] Robertson P. and Worz T. "Bandwidth-efficient turbo trellis-coded modulation using punctured component codes." *IEEE Journal in Selected Areas in Communications*, **16**(2):206–218, February 1998.

[Sha48] Shannon C. E. "A mathematical theory of communication." In *Bell Sys. Tech. Journal*, pp. 379–423 623–656, 1948.

[Sha57] Shannon C. E. "Certain results incoding theory for noisy channels." In *Information and Control*, volume 1, pp. 6–25, 1957.

[Ung82] Ungerboeck G. "Channel coding with multilevel/phase signals." *IEEE Transactions on Info. Theory*, **28**(1):55–67, Jan 1982.

[Wes96] Wesel R. D. "Trellis Code Design For Correlated Fading And Achievable Rates For Tomlinson-Harashima Precoding." *PhD Dissertation*, August 1996.

[Wes98] Wesel H. *Wireless Multimedia Communications.* Addison Wesley Longman, USA, 1998.

[Wes99] Wesel R. D. "Reduced complexity transfer function computation." In *ICC99*, Vancouver, Canada, June 1999.

[Wes00] Wesel R. D. "EE213: Channel and Source coding class notes." *EE Department, UCLA*, Spring 2000.

[WKL97] Wesel R. D., Komninakis C., and Liu X. "Towards optimality in Constellation Labeling." In *Communications Theory Mini Conference, GlobeCom 97*, pp. 23–27, Phoinix, AZ, November 1997.

[WL98] Wesel R. D. and Liu X. "Analytic techniques for periodic trellis codes." In *36th Annual Allerton Conference on Communication, Control, and Computing*, September 1998.

[WL00] Wesel R. D. and Liu X. "Edge-profile optimal constellation labeling." In *ICC 2000*, pp. 1198–1202, New Orleans, LA, June 2000.

[WLC00] Wesel R. D., Liu X., Cioffi M., and Komninakis C. "Constellation Labeling for Linear Encoders." *IEEE Transactions on Info. Theory*, 2000.

[WLS97] Wesel R. D., Liu X., and Shi W. "Periodic symbol puncturing of trellis code." In *Thirty-first Asilomar conference on signals, systems and computers*, Nov. 1997.

[WLS00] Wesel R. D., Liu X., and Shi W. "Trellis codes for periodic erasures." *IEEE Transactions on Communications*, **48**(6):938–947, June 2000.