https://ntrs.nasa.gov/search.jsp?R=19910013158 2020-03-19T17:49:43+00:00Z

SEMI-ANNUAL STATUS REPORT

Error Control Techniques for Satellite and Space Communications
NASA Grant Number NAG5-557

Principal Investigator:
Daniel J. Costello, Jr.

March 1991

Summary of Progress

During the period August 1, 1990 – January 31, 1991, significant progress was made in a number of areas. In this report, we will focus on the results included in the Ph.D. dissertation of Mr. Steven S. Pietrobon, a Ph.D. student supported by the grant. Mr. Pietrobon completed his dissertation in December, 1990 and will formally receive his Ph.D. degree in May, 1991. A copy of the dissertation is included as an Appendix to this report. One journal paper has already been published based on this research [1], and two more are being submitted to the IEEE Transactions on Information Theory this month [2,3]. In addition, a number of conference presentations have resulted from this work [4-11]. The following sections contain a brief summary of the important aspects of this dissertation.

1) Trellis Coded Multidimensional Phase Modulation

Since the publication of the paper by Ungerboeck [12], **trellis-coded modulation** (TCM) has become a very active research area. The basic idea of TCM is that by trellis coding onto an expanded signal set (relative to that needed for uncoded transmission), both power and bandwidth efficient communication can be achieved.

TCM can be classified into two basic types, the lattice type (e.g., M-**pulse amplitude modulation** (PAM) and M-**quadrature amplitude shift keying** (QASK)) and the constant amplitude type (e.g., **multiple phase shift keying** (MPSK)). Constant amplitude modulation schemes have a lower power efficiency compared with lattice type modulation schemes but are more suitable for certain channels, e.g., satellite channels containing nonlinear amplifiers such as **traveling wave tubes** (TWT's).

In any TCM design, partitioning of the signal set into subsets with increasing minimum intrasubset distances plays a central role. It defines the signal mapping used by the modulator and provides a tight bound on the **minimum free Euclidean distance** (d_{free}) between code sequences.

We have investigated a class of trellis-coded **multidimensional** (multi-D) MPSK modulation schemes. Signals from a **2L-dimensional** (2L-D) MPSK signal set (which we denote as L $\times$ MPSK) are transmitted over a **two-dimensional** (2-D) modulation channel by sending L consecutive signals of an MPSK signal set. Therefore, the L $\times$ MPSK signal set is the cartesian product of L 2-D MPSK signal sets.

An efficient method of partitioning multi-D MPSK signal sets has been developed that leads to easily implemented multi-D signal set mappers. When these signal sets are combined with trellis codes, significant asymptotic coded gains in comparison to an uncoded system are achieved. These codes provide a number of advantages compared to trellis codes with 2-D signal sets which make them particularly attractive for NASA satellite communication systems: 1) flexibility in achieving a variety of fractional information rates, 2) codes which are partially or totally transparent to discrete phase rotations of the signal set, 3) suitability for use as inner codes in a concatenated coding system, and 4) higher decoding speeds resulting

PRECEDING PAGE BLANK NOT FILMED

from the high rate codes used (rate $k/(k+1)$ with k up to 15 for some codes).

For trellis coded L $\times$ MPSK modulation, with $M = 2^I$, effective information rates of $I - (j/L)$ bits/symbol, for $j = 1, 2, \cdots, L$, can be achieved. This allows a system designer a greater choice of data rates than is available with 2-D signal sets ($L = 1$) without sacrificing data quality.

An analytical description of multi-D signal sets in terms of block code cosets, and the use of systematic convolutional encoding, results in an encoder design (from the differential encoder to the 2-D signal set mapper) that allows many good codes to be found. This approach also leads to the construction of signal sets that allow codes to be transparent to multiples of 360°/M phase rotations. Finally, due to the way the signal sets are mathematically constructed, a signal set mapper can be easily implemented by using basic logic gates and L-bit binary adders.

A systematic code search based on maximizing d_{free} (and thus the asymptotic coding gain) as well as minimizing the number of nearest neighbors for various degrees of phase transparency was performed. For L $\times$ 4PSK, asymptotic coding gains up to 7.8 dB compared to an uncoded system were obtained. For L $\times$ 8PSK and L $\times$ 16PSK, codes exhibiting asymptotic coding gains up to 5.85 dB were found.

Since a Viterbi decoder processes k bits in each recursion of the algorithm, the large values of k for codes using multi-D signal sets allows very high bit rates to be achieved (compared to convolutional codes that map only into a 2-D signal set). The large number of branch metric computations can be reduced either through the use of a modified Viterbi algorithm or large lookup tables. Finally, a method has been developed that uses the redundancy in some signal sets to achieve symbol synchronization at the decoder for codes that are not fully transparent.

Rate $k/(k+1)$ trellis codes with L $\times$ MPSK modulation also have the advantage of being useful as inner codes in a high rate concatenated coding system with **Reed-Solomon** (RS) outer codes over $GF(2^k)$. In the inner decoder makes errors, one trellis branch error will exactly match one symbol in the outer RS codeword. The symbol oriented nature of trellis coded L $\times$ MPSK inner codes can provide an improvement of up to 1 dB in the overall performance of a concatenated coding system when these codes replace bit oriented trellis coded 1 $\times$ MPSK inner codes of the same rate. This can be an extremely important advantage in achieving high bandwidth and power efficiency in concatenated coding systems such as NASA's TDRSS.

2) Trellis Coding with Multidimensional QAM Signal Sets

We have also performed a systematic code search for trellis codes with multi-D QAM signal sets. The 2-D signal sets used in the construction of the multi-D signal sets range from 16 to 512 points and were designed to have minimum energy, to be 90° rotationally symmetric, and to be suitable for partitioning. Where possible, the signal set selected is the same as that commonly used in the literature and in practical implementations of TCM schemes.

Rate $k/(k+1)$ codes were used in the code search. The codes presented all have signal

sets with 2^I signal points, where I is a positive integer. Some of the advantages of multi-D signal sets are: possible 90° phase invariance, lower coding complexity, non-integer information rates, and suitability for use in a concatenated system. The multi-D QAM signal sets were constructed through the use of cosets by a method similar to that used for multi-D MPSK.

The code search found the codes having the largest **minimum free Euclidean distance** (d_{free}) and the smallest number of nearest neighbors. This maximizes the asymptotic coding gain and minimizes the bit error probability at high SNR. There are usually two different possible phase transparencies for a linear code, and the best codes for each phase transparency were found. The information rates ranged from 3 to 8 bits/symbol, with signal sets up to eight dimensions for the 16QAM and 32 CROSS constellations, six dimensions for the 64 CIRC constellation, four dimensions for the 128 CROSS and 256 CIRC constellations, and two dimensions for the 512 STAR constellation. (The 64 CIRC, 256 CIRC, and 512 STAR constellations are new 2-D signal sets.) Codes were found having asymptotic coding gains up to 6 dB.

The codes constructed for the small size signal sets (especially 16 QAM) may be useful in NASA's satellite communication systems where high bandwidth efficiency is required at the expense of more linear amplifiers. The codes constructed for the larger size signal sets may be useful for high capacity microwave links and telephone modems where high data rates are required on bandwidth limited channels.

3) Rotationally Invariant Trellis Codes

One aspect of trellis coding that has come under increasing study is the search for codes that are invariant to phase rotations of the received signal set. The rotations under consideration are those caused by a demodulator in a communication system. When the signal set has rotational symmetries, e.g., MPSK or 16QAM, the demodulator has no knowledge of which of the symmetries was transmitted. Thus, the demodulator selects one of the symmetries with which to demodulate the received signal, regardless of whether it is the correct or incorrect symmetry.

In uncoded systems, this problem is easily corrected by differentially encoding (*precoding*) the data before transmission. After demodulation, differenital decoding (*postdecoding*) of the received data is then used to return the data to its original form. Precoding of the data also allows the recovery of data altered by phase slips within the demodulator. This occurs when noise in the received signal causes the demodulator to lose lock and results in another of the signal set symmetries being selected.

For trellis coding the situation is much more complicated. Here, we are dealing with sequences of symbols in the code space rather than independent symbols, as in the uncoded case. In fact, convolutional and trellis codes can be thought of as subclasses of *sequence codes.* Unlike block codes, sequence codes have code words of infinite length, consisting of sequences of symbols taken from a finite or infinite size signal set. Any finite or infinite set of sequences can be considered as a sequence code. If a coded sequence has been rotated, the resulting

code sequence may or may not be in the code space.

The *transparency* or *rotational invariance* of a sequence code is the minimum non-zero phase rotation for which all code sequences in the code space can be rotated such that the rotated sequences are still in the code space of the sequence code. A sequence code is *rotationally invariant* or *transparent* if the invariance of the code is equal to the minimum non-zero phase symmetry of the **two-dimensional** (2-D) signal set. If there are some sequences which are not in the code space after a phase rotation, a decoder will produce erroneous data if the received sequence has been rotated by this amount.

A good example of this is the NASA standard (2, 1, 6) convolutional code with Gray mapped QPSK modulation. This code is not 90° transparent (and is therefore not rotationally invariant), but it is 180° transparent. A decoder will produce erroneous data after a 90° or −90° rotation. To overcome this, the decoder needs to recognize that a 90° rotation has occurred and rotate the received sequence. This process can be slow, resulting in many errors being produced before the decoder is properly synchronized. A rotationally invariant code, however, will only produce a small number of errors after a phase rotation, since there is no need to detect and then correct for a phase rotation.

In order to describe and study rotationally invariant sequence codes, we use the **parity check equations** (PCE) of a code. For rate $k/(k+1)$ codes, a single PCE fully describes the relationship between the 2-D symbols in a code sequence. However, the PCE gives no information about the input/output relationship of an encoder, i.e., it is independent of the encoder implementation. This allows us to minimize the number of variables in finding good rotationally invariant codes, thus simplifying the code search.

A systematic method of obtaining rotationally invariant trellis codes for a variety of 2-D signal sets has been developed. Since codes based on linear PCE's cannot be rotationally invariant for 2-D signal sets with more than two points, an alternative general nonlinear PCE was found. This nonlinear PCE allows the construction of invariant codes for 2-D signal sets that are "naturally" mapped.

A general method of combining the precoder with a systematic encoder without increasing the encoder memory was also discovered. This eliminates the need for a postdecoder, since the precoder is part of the encoder trellis.

When a signal set has 90° rotational symmetries or only one input bit is checked by the encoder, the general PCE is relatively simple, with only one non-linear term. The best rotationally invariant nonlinear codes found for QPSK, 8PSK, and 16PSK signal sets have smaller free distances than the best corresponding linear codes. However, their low number of nearest neighbors may result in good performance at moderate E_b/N_0 ratios. The QAM codes found were very good. Most of these codes had the same free distance as the best corresponding linear codes. In particular, the new 90° rotationally invariant rate 3/4, 64 state, 16-QAM code with a 5.44 dB asymptotic coding gain is being considered for adoption by CCITT as the V.FAST coding standard for Two-Wire High-Speed Modems. This code transmits 3 bits/symbol and achieves an almost 5 dB real coding gain at a BER of 10^{-5} over uncoded 8PSK.

4) Implementation of a Bandwidth Efficient Coding Scheme for the Hubble Space Telescope

A trellis coding scheme using 8PSK modulation has been designed for use on NASA's **Hubble Space Telescope** (HST). By using a four dimensional signal set (i.e., the cartesian product of two 8PSK symbols) and a rate 5/6 encoder, it is possible to obtain a bandwidth efficiency of 2.5 bits/symbol. This implies that the data rate can be increased from the current 1 Mbit/s to 7.5 Mbit/s without any increase in bandwidth. The code selected has 16 states and gives a real coding gain of 3.1 dB compared with uncoded 2.5 bits/symbol 8PSK and 1.5 dB compared with uncoded QPSK at a bit error rate of 10^{-5}. Due to the multidimensional signal set, this code is also fully rotationally invariant. A 2 Mbit/s serial implementation of a Viterbi decoder is being implemented for this code. This work is due to be completed by the end of the grant period.

References

[1] D. J. Costello, Jr., S. S. Pietrobon, R. H. Deng, A. LaFanechere, and G. Ungerboeck, "Trellis Coded Multi-Dimensional Phase Modulation", *IEEE Trans. Inform. Th.*, IT-36, pp. 63-89, January 1990.

[2] S. S. Pietrobon and D. J. Costello, Jr., "Trellis Coding With Multidimensional QAM Signal Sets", submitted to *IEEE Trans. Inform. Th.*, March 1991.

[3] S. S. Pietrobon, G. Ungerboeck, and D. J. Costello, Jr., "Rotationally Invariant Trellis Codes", submitted to *IEEE Trans. Inform. Th.*, March 1991.

[4] D. J. Costello, Jr., S. S. Pietrobon, G. Ungerboeck, R. H. Deng, and A. LaFanechere, "Channel Coding with Multi-Dimensional Trellis Coded Phase Modulation", IEEE Int. Symp. on Inform. Th., Kobe, Japan, June 1988.

[5] D. J. Costello, Jr., S. S. Pietrobon, and G. Ungerboeck, "Trellis Coded Phase Modulation", Proc. Int. Forum on Inform. Th. and Its Applications, pp. LW3.1-LW3.11, Tokyo, Japan, July 1988 (Invited Paper).

[6] S. S. Pietrobon, D. J. Costello, Jr., and G. Ungerboeck, "A General Parity Check Equation for Rotationally Invariant Trellis Codes", 1989 IEEE Information Theory Workshop, Cornell University, Ithaca, NY, June 1989 (Invited Paper).

[7] S. S. Pietrobon and D. J. Costello, Jr., "Trellis Coding Using Multi-dimensional QAM Signal Sets", 1990 IEEE International Symposium on Information Theory, San Diego, CA, January 1990.

[8] D. J. Costello, Jr., S. S. Pietrobon and G. Ungerboeck, "Rotationally Invariant Trellis Codes", *IEEE Int. Symp. on Inform. Th.*, San Diego, CA, January 1990.

[9] S. S. Pietrobon, G. Ungerboeck, and D. J. Costello, Jr., "Nonlinear Rotationally Invariant Trellis Coding", 1991 IEEE Communication Theory Workshop, Rhodes, Greece, July 1991.

[10] D. J. Costello, Jr., G. Ungerboeck, and S. S. Pietrobon, "New Rotationally Invariant QAM Trellis Codes", 5th Tirrenia International Workshop on Coded Modulation and Bandwidth Efficient Transmission, Tirrenia, Italy, September 1991.

[11] S. S. Pietrobon, D. J. Costello, Jr., and M. J. Miller, "A Bandwidth Efficient Coding Scheme for the Hubble Space Telescope", 2nd Space Communications Technology Conference, Cleveland, Ohio, October 1991.

[12] G. Ungerboeck, "Channel Coding with Multilevel/Phase Signals", *IEEE Trans. Inform. Theory*, IT-28, pp. 55-67, January 1982.

Appendix

Trellis Coding With Multidimensional Signals Sets And Rotationally Invariant Trellis Codes

TRELLIS CODING WITH MULTIDIMENSIONAL SIGNAL SETS AND ROTATIONALLY INVARIANT TRELLIS CODES

A Dissertation

Submitted to the Graduate School
of the University of Notre Dame
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

by

Steven Silvio Pietrobon, B.Eng., M.Eng.

Daniel J. Costello Jr.
Director

Department of Electrical Engineering
Notre Dame, Indiana

December, 1990

TRELLIS CODING WITH MULTIDIMENSIONAL SIGNAL SETS AND ROTATIONALLY INVARIANT TRELLIS CODES

Abstract

by

Steven Silvio Pietrobon

Shannon's capacity bound shows that coding can achieve large reductions in the required E_b/N_0 in comparison to uncoded schemes. For bandwidth efficiencies of 2 bit/sym or greater, these improvements have been obtained through the use of Trellis Coded Modulation (TCM) and Block Coded Modulation (BCM). A method of obtaining these high efficiencies using multidimensional MPSK and QAM signal sets with trellis coding is described. These schemes have advantages in decoding speed, phase transparency, and coding gain in comparision to other trellis coding schemes. Finally, a general parity check equation for rotationally invariant trellis codes is introduced from which non-linear codes for two dimensional MPSK and QAM signal sets are found. These codes are fully transparent to all rotations of the signal set.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

ACKNOWLEDGEMENTS

I would like to thank the National Aeronautics and Space Administration (grant number NAG5-557), National Science Foundation (grant number NCR89-03429), Overseas Telecommunications Commission (Australia), and the Digital Communications Group at the South Australian Institute of Technology for their support.

In Chapter 2, I wish to acknowledge the previous work of Dr. Robert H. Deng and Mr. Alain Lafanechère. They provided the initial basis of using block codes to partition a multidimensional MPSK signal set. Also, I would like to thank Dr. Gottfried Ungerboeck for providing valuable advice in the preparation of this chapter and for inspiring me to find a general derivation for rotationally invariant trellis codes.

Finally, I would to thank Dr. Jospeh M. Nowack for reviewing this dissertation and my advisor, Prof. Daniel J. Costello, Jr., for his advice, comments, and assistance in my research.

CHAPTER ONE

INTRODUCTION

In nearly all communication systems which are modeled with an Additive White Gaussaian Noise (AWGN) channel, there are three main parameters that determine the performance of the system. These are the information bit error rate (P_b), the signal to noise ratio per information bit (E_b/N_0 where E_b is the energy per bit and $N_0/2$ is the double sided noise density) and the bandwidth efficiency (K, the number of information bits transmitted in each signalling interval of T seconds). In the communication schemes we are considering, a two-dimensional symbol is transmittted in each signalling interval. We use the unit bit/sym for K.

Assuming a flat channel, these three parameters can be related to each other through Shannon's famous capacity bound [62]

$$C = B \log_2(1 + E_s/N_0), \tag{1.1}$$

where C is the capacity of the channel (in bit/sec), B is the bandwidth (in Hz), and E_s/N_0 is the signal to noise ratio (E_s is the energy per symbol). We have taken the base two logarithm since bits are used as the basic unit of information. Shannon's noisy channel coding theorem effectively states that P_b can be made as small as desired as long as the transmission rate does not exceed C for the given B and E_s/N_0. Conversely, reliable communication is not possible if C is exceeded.

We can modify (1.1) so that the capacity bound relates K and E_b/N_0 instead of C, B and E_s/N_0. Assuming perfect Nyquist signalling (i.e., the bandwidth expansion factor is one) we have that

$$K \leq C/B, \tag{1.2}$$

i.e., K must be less than C/B for reliable transmission. We also have

$$E_s/N_0 = K\ E_b/N_0. \tag{1.3}$$

Thus, substituting (1.2) and (1.3) into (1.1) we obtain the bound

$$E_b/N_0 \geq \frac{2^K - 1}{K}. \tag{1.4}$$

This bound is plotted in Figure 1.1. Note that the minimum E_b/N_0 that can be acheived is ln 2 or -1.59 dB. However, to achieve this minimum, K must be infinitely small (i.e., approaching 0). This implies that very large bandwidths will be required. As K increases, the minimum E_b/N_0 also increases. This fact is very important since it tells us the price we have to pay (in terms of E_b/N_0) in order to achieve larger K and thus greater bandwidth efficiency.

Also shown in Figure 1.1 are points for uncoded Binary Phase Shift Keying (BPSK) and Quadrature Phase Shift Keying (QPSK) at a P_b of 10^{-5}. Note the large improvements in E_b/N_0 that can be achieved, even if we maintain the same value of K. To achieve these gains, *coding* must be used.

Traditionally, this has been achieved through the use of block codes, convolutional codes, or a combination of the two. These schemes involve the addition of redundant bits to the information bits. These

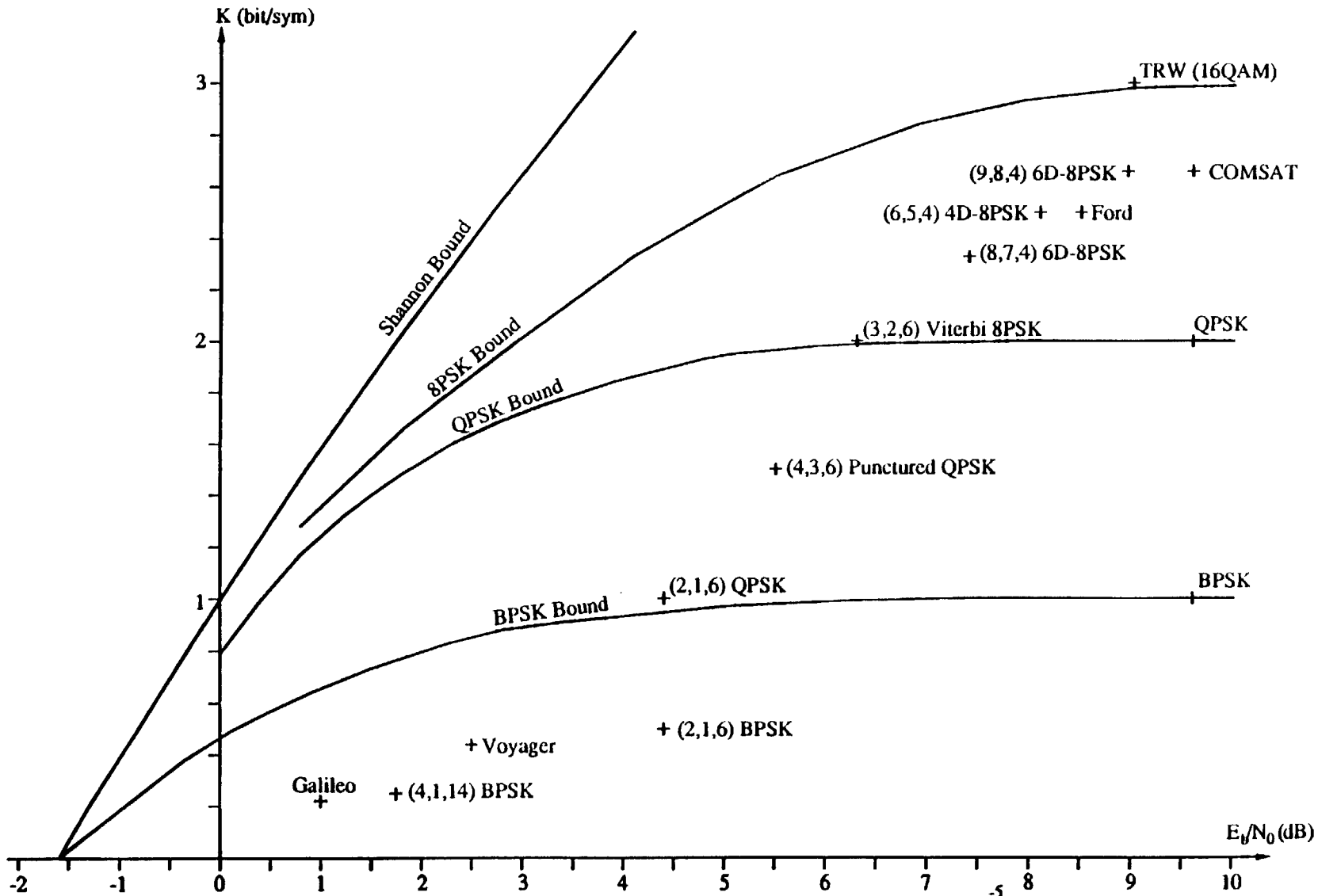

Figure 1.1: Plot of bandwidth efficiency K (bit/sym) verses E_b/N_0 for $P_b = 10^{-5}$.

redundant bits are used to increase the *minimum free Hamming distance* (d^H_{free}) of a code compared to an uncoded scheme where Hamming distance is defined as the number of differing bits between two codewords. The d^H_{free} of a code is the minimum Hamming distance between all non-equal codewords.

BPSK or (Gray mapped) QPSK modulation is used with these codes because the squared Euclidean distance between two codewords is proportional to the Hamming distance between the codewords. Thus, the minimum squared Euclidean distance between non-equal codewords (d^2_{free}) is proportional to the d^H_{free} of a code. Therefore, one can find a code without considering the signal set being used. The limitation of these codes is that K cannot be greater than or equal to two bit/sym.

The parameter K is important because the required bandwidth for a communication system is inversely proportional to K. When bandwidth is limited such that the required K is 2 bit/sym or greater, alternative coding techniques are required. Since we cannot increase bandwidth (as the traditional schemes do), the only way to obtain redundancy is through an expanded signal set. The basis for this technique was first described systematically by Ungerboeck [65].

Ungerboeck obtained bounds of K versus E_s/N_0 for various one dimensional (1-D) and 2-D signal sets. Some of these bounds are reproduced in Figure 1.1 for BPSK, QPSK, and 8PSK modulation (E_b/N_0 is used as the reference here, though). As can be seen, the potential coding gain of using 8PSK for K = 2 bit/sym compared with uncoded QPSK is 6.6 dB (this is only 1.2 dB less than the theoretical minimum for K = 2 bit/sym). This would require an infinite amount of coding effort though, and so a practical coding system will have a coding gain

somewhat less than 6.6 dB.

The codes that Ungerboeck obtained in [65] are based on convolutional codes with k input bits and k+1 output bits, which are then mapped into a signal set. Convolutional coding schemes with K of 2 bit/sym or greater have become known as Trellis Coded Modulation (TCM). The difference between these schemes and traditional convolutional codes is that the codes are based on d^2_{free} only and not on the d^H_{free} of the code.

More recently, there has been active research in using Block Coded Modulation (BCM) to obtain high bandwidth efficiency [5,11,13,26,27,31,34-36,39,41,59,61,63,80]. The basic method for this was first outlined in [33] and is also known as multi-level coding.

1.1 Partitioning

An important class of signal sets is Multiple Phase Shift Keying (MPSK) modulation. These signal sets have the property that each 2-D signal is of equal amplitude. This is useful in nonlinear channels such as in communication satellites with travelling wave tube amplifiers where the signal set remains largely unaffected (although spectral regrowth can occur causing an increase in bandwidth). Quadrature Amplitude Modulation (QAM) signal sets have the advantage of greater efficiency (due to their denser packing), but usually require a linear or near-linear channel to avoid any distortion of the signal point levels.

An important part of either TCM or BCM is in the partitioning of the signal set being used. This is related to partitioning a set into

subsets. Each partition divides a previous set into two subsets, with an equal number of points in each subset. So, starting with the original signal set, we divide this set into two subsets. Each of these two subsets are divided into two and so on, until only one point remains in each subset (we assume that the number of points in the signal set is a power of two).

The partition is usually made such that the minimum squared Euclidean distance between all non-equal points in each subset is as large as possible. The minimum of these distances over all the subsets is called the *Minimum Squared Subset Distance* (MSSD) at *partition level* p (δ_p^2, the partition level starts at 0 for the full signal set and increments by one for each two-level partition). Usually, due to symmetry in the signal set, δ_p^2 is the same in each subset. If δ_p^2 is the same as the previous partition level, we try to minimize the average number of nearest neighbors in each coset.

This partition leads to a mapping of n bits into each of the 2^n points (the 2^n subsets at partition level n). It is this mapping and the δ_p^2's of the signal set that lead to construction of good TCM or BCM schemes.

1.2 Trellis Coding with Multidimensional Phase Modulation

There are some limitations of only mapping into a 2-D signal set with linear convolutional codes. One of them is that this class of codes has an integer value of K (since k = K for a rate k/(k+1) code). To obtain fractional rates, schemes have been developed that use periodically time varying trellis codes (PTVTC) [30]. For example, to

obtain K = 2.5 bits with 8PSK modulation, two trellis codes are used. The first code transmits 2 bits for the first symbol, followed by 3 bits for the next symbol. This process then repeats.

For a specific PTVTC it has been shown that one can obtain a single encoder that modulates L 2-D signal sets, where L is the period of the code [48]. There appears to be no reason why this is not true in general (on the condition that the PTVTC convolutional encoder is linear). This would indicate that it might be better to find a code that considers the L 2-D signal sets as a whole, rather than one at a time. Indeed, this appears to be true in terms of coding gain and a number of other criteria as well.

This concatenation of L signal sets is called a *multi-dimensional* (or *multi-D)* signal set. Other work on trellis codes that have used multi-D signal sets are described in [3,4,6,14-17,19,37,48,50,51,56,75,76]. Usually, the concatenation is with 2-D signal sets, giving a 2L dimensional signal set.

A major part of obtaining good codes with multi-D signal sets is in finding good partitions as described above. If there are I bits for each 2-D signal set, the total number of points in each signal set is 2^{IL}. Even for small I and L (e.g., I = 3 and L = 2) the number of multi-D points becomes very large and thus finding a good partition (or partitions) by hand becomes very difficult.

A solution to this problem is to use the partition of the 2-D signal set. For example, we can use the partition (and notation for the subsets) for 8PSK given in [65] (reproduced in Figure 1.2). A 4-D 8PSK signal set would consist of the set A0×A0 where × denotes the cartesian product. The first partition could consist of the cosets B0×B0 ∪ B1×B1

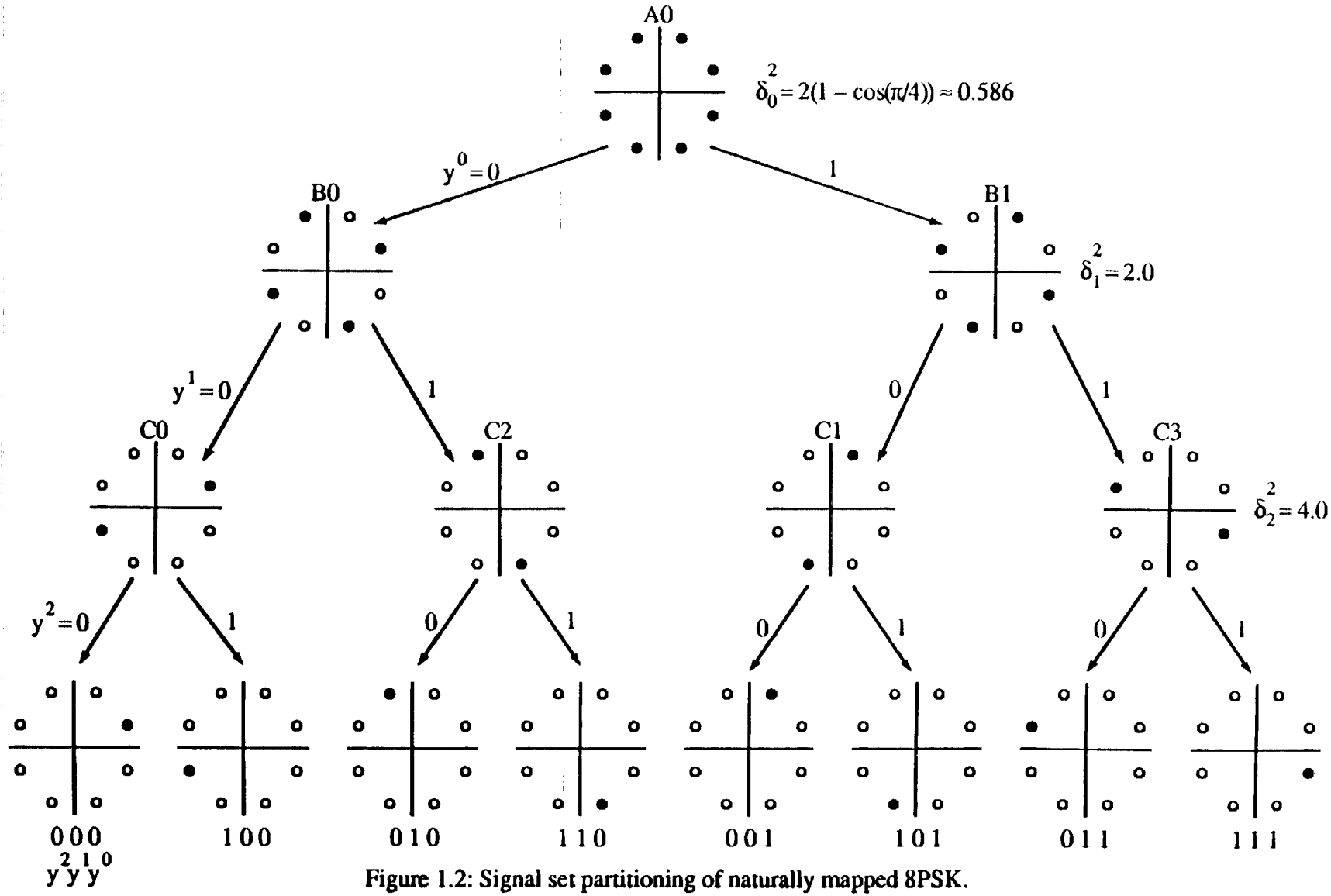

Figure 1.2: Signal set partitioning of naturally mapped 8PSK.

and B0×B1 ∪ B1×B0. It is also not too difficult to see what the MSSD is for these two levels ($\Delta_0^2 = 0.586$ and $\Delta_1^2 = 2\Delta_0^2 = 1.172$, where we use Δ to indicate the MSSD's for multi-D signal sets). The rest of the partition can follow in a similar manner.

Although this method gives a good intuitive feel of how multi-D partitions are made, it has its limitations when larger values of L are considered. In fact, the problem becomes very similar to finding good block codes (and their cosets), except that we are dealing with more than one "level" of coding (also called coding level). Each level of coding effectively codes one of the partition levels of the 2-D signal set. To distinguish one level from another, powers of 2 are used in the codewords. For example, a codeword at partition level 0 (of the 2-D signal set) might have a code word of [1 1], at level 1, a codeword of [2 2] and so on. This can be seen in the previously described example. At partition level one of the 4-D signal set, we can see that the first coset has two code words of length 2 at code level 0, corresponding to the (2,1) block code with a d^H_{free} of 2. The other two code levels are uncoded, i.e., they use the (2,2) block code which has a d^H_{free} of one. Another method for constructing multi-D signal sets is Forney's 2-construction or 3-construction [23].

In Chapter 2, we present in detail the construction of trellis codes that use multi-D MPSK or L×MPSK. The encoder is broken into three parts, the differential encoder, a systematic convolutional encoder and the signal set mapper.

In Chapter 2, the results of a systematic code search for a wide variety of rates and signal sets are presented. The 2-D signal sets that were used are QPSK, 8PSK and 16PSK, with the number of 2-D signal

sets (L) varying from one to four. Codes with the following values of K were found, 1.0, 1.25, 1.33, 1.5, 1.67, 1.75, 2.0, 2.25, 2.33, 2.5, 2.67, 2.75, 3.0, 3.25, 3.33, 3.5, 3.67, and 3.75 bits. The complexity of the codes (the number of checked bits, $\tilde{k}$, plus the memory of the encoder, ν) ranges up to a value of 10.

1.2.1 Code Transparency

The construction of the signal sets has been described by use of coset representatives or generators. This method has also been described in [22]. The generators allow the signal set mapper to be easily implemented using exclusive OR gates and $I = \log_2 M$ bit binary adders.

A significant advantage of using generators for this mapping is that it determines a simple relationship of how the n mapping bits are affected when the signal set has been rotated by $\Psi = 360/M$ degrees, e.g., 8PSK being rotated 45° (this can be caused by phase slips within a demodulator). With a 2-D signal set, nearly all linear convolutional codes are not fully transparent to phase rotations of Ψ (an exception is BPSK). This implies that a Viterbi decoder needs to synchronize with the received signal set. These synchronizers are usually based on a measure of the performance of the decoder. Due to the synchronizers random nature, it may take many symbols before a decoder recognises an out of synch condition and attempts to lock on to the correct phase. With MPSK modulation, this can be a significant problem as the points are very close in phase (45° for 8PSK, 22.5° for 16PSK). Demodulators are thus more likely to have phase slips than if QPSK or BPSK were used.

With a transparent code, a phase rotation has no affect on the coded sequences and so a Viterbi decoder does not need to have a synchronising circuit, simplifying the design and possibly improving performance. The coded sequences may remain the same on a phase rotation but the relation between the information sequence and coded sequence will change with a phase rotation. To overcome this, differential encoding (or precoding) before the encoder and differential decoding (or postdecoding) after the decoder are used.

With modulo-M or a combined modulo-M and modulo-2 addition of the generators, it is shown in Chapter 2 that at most I bits out of the $n \leq IL$ bits used in the mapping are affected by a phase rotation of Ψ. Since these bits are usually evenly spread out through the mapping bits, many codes can be found that are fully transparent to phase rotations of Ψ (this is because only two to four of the mapping bits are actually coded, the rest remaining uncoded). This mapping also allows the design of general precoders and postdecoders.

1.2.2 Decoder Speed

Another advantage of using multi-D signal sets is the large values of k that can be used. This can result in very high decoding speeds, possibly approaching 1 Gbit/s for a single codec. This is because each iteration of the Viterbi algorithm decodes as many as IL-1 bits (the maximum value of k), while the decoder complexity, in terms of the number of coded bits and encoder memory, is about the same as for trellis codes which use only a 2-D signal set. However, the branch metric calculator is usually much more complicated, due to the greater number of parallel transitions between the trellis.

As an example, a decoder that has been built for a PTVTC has an internal clock speed of 75 MHz [18]. The code is of rate 8/9 with 8PSK modulation giving K = 2.67 bit/sym. Since the 2-D symbol rate is the same as the decoder internal clock speed, the bit rate is limited to 200 Mbit/s. However, the equivalent multi-D scheme (which uses a 3×8PSK signal set) has each decoder iteration decoding 8 bits instead of 2.67 bit/sym on average. Thus, this same decoder (with about 50% extra hardware due to the parallel transitions that need to be decoded) could be made to have a bit rate of 600 Mbit/s! This code also has other advantages due to the multi-D signal set. It is fully transparent (compared with only 180° transparency for the PTVTC) and it has a asymptotic higher coding gain due to its larger d^2_{free}.

In Figure 1.1 we have plotted the points for the COMSAT code, as well as the point for the equivalent multi-D code [46] at a P_b of 10^{-5}. As can be seen, the multi-D code achieves an extra 0.5 dB coding gain. Also shown in Figure 1.1 are various other coding schemes.

1.2.3 Decoder Implementation

The practical side of trellis codes is examined in Chapter 5. Here, the implementation of one of the 262 codes from Chapter 2 that were found in the search is described. A soft decision Viterbi decoder and encoder for this code has been designed. Initially, the code that was chosen was a rate 7/8 code with 3×8PSK modulation, K = 2.33, two checked bits and 16 states.

This code was chosen mainly due to a previous PTVTC code that was implemented which has similar properties to the code we have chosen [30] (called the COMSAT code, after the company that developed the

decoder). Note that this COMSAT code is not the same code as that shown in Figure 1.1. This COMSAT code has a rate of 7/9, uses 8PSK modulation, has K =2.33, a maximum of two checked bits and 16 states. The main differences between our code and the COMSAT code is that for an equivalent hardware complexity, our code is about L = 3 times faster in terms of bit rate and has a phase transparency of 90° verses 180° for the COMSAT code [48]. Our code also has a larger d^2_{free} (4.0 instead of 3.515). Figure 1.1 plots the simulated performance of our code [46], showing a 2.2 dB coding gain in comparison to uncoded QPSK at a P_b of 10^{-5}.

Recent developments have led us to change our code to a 2.5 bit/sym, 4-D 8PSK code (its simulated performance [46] is also plotted in Figure 1.1). This code also has 16 states and two checked bits, implying minimal changes to the already exisiting design. It is also 45° invariant. The reason for the change are due to INTELSAT's new SONET standard which requires a 155.52 Mbit/s bit rate through a 70 MHz channel. A 2.5 bit/sym code will be able to meet this standard and so our low data rate design could be a "proof of concept" for the SONET system.

The codec is to be implemented using standard TTL logic (mainly low power and advanced Schottky). The internal clock speed of the Viterbi decoder is 10 MHz. Each of the 16 add-compare-select (ACS) operations (one for each state), is to be performed serially. Thus, only one ACS circuit is required. A total of approximately 23 clock cycles are required for each iteration of the Viterbi algorithm. This gives a maximum bit rate of 3.04 Mbit/s and 2.17 Mbit/s for the 2.33 bit/sym and 2.5 bit/sym codes, respectively.

1.3 Trellis Coding with Multidimensional QAM signal sets

The methods that are described in Chapter 2 for MPSK modulation can also be applied to QAM constellations. Some work has already been done in this area [5,8-10,12,22,24,64,68,74] but nearly all of it has been ad hoc in nature (some codes were even designed by "hand"). In Chapter 3, we apply the systematic multi-D construction and code search methods developed for MPSK signal sets to that of QAM signal sets. A "natural" mapping of each 2-D 90° rotationally symmetric QAM signal set (ranging from 16 to 512 points) is presented. This allows us to easily determine the phase properties of the codes.

The code search used 2-D to 8-D signal sets with K ranging from 3.0 to 8.0 bit/sym. Some of the larger size signal sets were further limited in the number of dimensions due to computational limitations. The benefits of the code search have resulted in new codes being found that are 90° invariant and that have the fewest number of nearest neighbors (N_{free}). Also, a 16 state, 3 bit/sym 8-D 16QAM code was found that has 6.02 dB asymptotic coding gain (γ). This code can also be used in larger size signal sets for the same γ (although N_{free} may not be optimum).

1.4 Rotationally Invariant Trellis Codes

As has been described earlier, a desirable property of trellis codes is to have full transparency. One way to achieve this is through a multi-D signal set. If $L \geq 2$, the decoder will need to synchronize

onto the L 2-D symbols. This may be a problem in some channels where symbol loss is likely to occur, e.g., in fading channels. Thus, codes are desired which have only a 2-D signal set (to avoid symbol synchronisation) and that are fully transparent. Linear codes cannot achieve these requirements. This is because at least two bits are always affected in the signal set mapping. The parity check equation for linear codes (which describes the relationship between the coded bits) will always change on a phase rotation for all non-trivial codes.

To obtain phase transparency, the code has to be made *non-linear*. That is, there are logical AND operations in the code. Rotationally invariant codes have been previously found in [3,10,19,40,75] with multi-D signal sets and in [1,3,32,44,52,54,73,74,81] for 2-D signal sets. The work in [48,49,70] concentrated on finding rotationally invariant rate 1/2 codes with QPSK modulation. We have extended this work into finding rotationally invariant codes for rate 2/3 8PSK and rate k/(k+1) QAM.

To obtain the codes in [48] we used a parity check equation that was found by Ungerboeck [66]. This parity check equation was designed such that the equation remained the same after a 90° phase rotation. To achieve this, a non-linear term was added into the equation. With rate 2/3 8PSK the situation becomes much more complicated. The parity check equation now has to compensate for three terms being affected on a phase rotation, not just two as in the QPSK case. To find the various forms of the parity check equation for rate 2/3 8PSK, a different approach was taken.

This was achieved by finding a general parity check equation for any rate k/(k+1) rotationally invariant trellis code that has a natural

type mapping (it is really two equations, one with modulo-2 arithmetic and the other with modulo-M arithmetic). From this relatively simple general equation, one can derive the rate 1/2 general parity check equation that was found by Ungerboeck. Also, the complicated rate 2/3 equations can also be derived. With these parity check equations, a code search for rate 2/3 8PSK rotationally invariant trellis codes was performed.

The rate 2/3 codes with two checked bits were found to have 0 dB asymptotic coding gain, but with very small N_{free}. This small N_{free} may make these codes suitable for fading channels in comparison to uncoded QPSK. Other rate 2/3 codes with one checked bit were also found. However, their parallel transitions in a fading channel is a significant disadvantage (although the codes have γ = 3.0 dB).

A surprising result was found for the rotationally invariant QAM codes. For all except one code, the d^2_{free} of the invariant codes are the same as the best linear codes, with the invariant code having *fewer* N_{free} than the linear codes. The results of a code search for rate 1/2 QPSK codes are also presented.

CHAPTER TWO

TRELLIS-CODED MULTIDIMENSIONAL PHASE MODULATION

In this chapter, we investigate a class of trellis coded multidimensional (multi-D) MPSK modulation schemes. Signals from a 2L-dimensional (2L-D) MPSK signal set (which we shall denote as L×MPSK) are transmitted over a 2-D modulation channel by sending L consecutive signals of an MPSK signal set. Therefore, the L×MPSK signal set is the Cartesian product of L 2-D MPSK signal sets. Trellis coded multi-D phase modulation (TC-L×MPSK) provides us with a number of advantages that usually can't be found with TC-MPSK: (i) flexibility in achieving a variety of fractional information rates, (ii) codes which are partially or totally transparent to discrete phase rotations of the signal set, and (iii) higher decoder speeds resulting from the high rate codes used (rate k/(k+1) with k up to 15 for some codes).

In Section 2.1, we introduce a block coding technique for partitioning L×MPSK signal sets. Section 2.2 describes how the encoder system, comprising a differential precoder, a systematic convolutional encoder, and a multi-D signal set mapper, is obtained for the best codes found in a systematic code search. The signal sets are designed such that the codes can become transparent to integer multiples of 360°/M rotations of the MPSK signal set. Also, due to the way in which they are mathematically constructed, a signal set mapper can be easily implemented by using basic logic gates and L bit binary adders. The

systematic code search is based on maximizing the minimum free Euclidean distance (d_{free}) and thus the asymptotic coding gain, as well as minimizing the number of nearest neighbors (N_{free}) for various degrees of phase transparency. TC-L×4PSK, TC-L×8PSK, and TC-L×16PSK codes for L = 1 to 4 are found. For TC-L×8PSK and TC-L×16PSK, asymptotic coding gains up to 5.85 dB compared to an uncoded system are obtained. The TC-L×4PSK codes exhibit asymptotic coding gains up to 7.8 dB. Among the L = 1 codes listed are some new codes which have improvements in N_{free} and phase transparency compared to codes found previously [43,65,68,76]. Viterbi decoding of TC-L×MPSK is also discussed, concentrating on maximum likelihood decoding of the parallel transitions within a code trellis.

2.1 Multi-D Signal Set Partitioning

In order to describe set partitioning we will start with the familiar partitioning of the 8PSK signal set. This is followed with an example of multi-D signal set partitioning using the 2×8PSK signal set. Generalizations will be gradually introduced, so that by the end of this section the reader should become thoroughly familiar with the concepts involved.

2.1.1 Partitioning the 8PSK Signal Set

In partitioning the 8PSK signal set, or 1×8PSK, we form a minimum squared subset distance (MSSD) chain of $\delta_0^2 = 0.586$, $\delta_1^2 = 2$, $\delta_2^2 = 4$, and $\delta_3^2 = \infty$ (assuming that the average signal energy is one). Figure 1.2 illustrates this partitioning, in which each subset is equally divided

into two smaller subsets such that the MSSD in each smaller subset is maximized. Partitioning continues in this manner until we have eight subsets, each containing a single point, hence $\delta_3^2 = \infty$.

2.1.2 Partitioning 2×8PSK

A 2×8PSK signal set (L = 2) is illustrated in Figure 2.1. We use integers y_1 to indicate the first 8PSK point and y_2 for the second 8PSK point, where $y_1, y_2 \in \{0,1,\ldots,7\}$. Natural mapping is used to map the integer y_j into each complex valued 8PSK signal, i.e., $y_j \mapsto \exp[\sqrt{-1}\, y_j \pi/4]$, for $j = 1,2$. We can also represent y_1 and y_2 in binary form as the vector $y_j = [y_j^2, y_j^1, y_j^0]$, with $y_j^i \in \{0,1\}$, and where $y_j = 4y_j^2 + 2y_j^1 + y_j^0$, for $j = 1,2$. That is, the least significant bit (lsb) of y_j corresponds to the right most bit and the most significant bit (msb) to the left most bit. We will use this convention throughout the chapter.

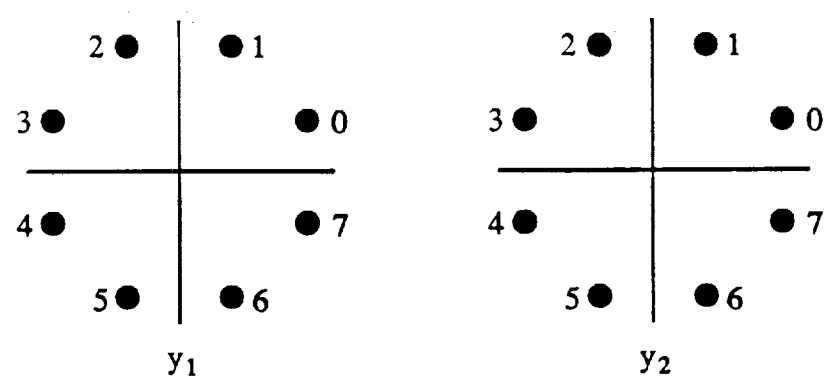

Figure 2.1: The 2×8PSK signal set.

To represent a 2×8PSK signal point we form the 2×3 binary matrix,

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} y_1^2 & y_1^1 & y_1^0 \\ y_2^2 & y_2^1 & y_2^0 \end{bmatrix}.$$

Since there are a total of six bits used to describe a signal point, the unpartitioned signal set (indicated by Ω^0) has a total of $2^6 = 64$ points. We also say that Ω^0 is at partition level $p = 0$. It can easily be seen that the MSSD at partition level $p = 0$ is $\Delta_0^2 = \delta_0^2 = 0.586$ (we use large Δ to indicate the MSSD's for $L > 1$ and small δ for $L = 1$). The next partition (at partition level $p = 1$) divides Ω^0 into two subsets of 32 points each. We call Ω^1 the subset that contains the all zero element (i.e., $y_1 = y_2 = 0$). The other subset of 32 points is its coset, labeled $\Omega^1(1)$. In forming these two subsets, we would like their MSSD, Δ_1^2, to be larger than Δ_0^2. If this were not possible, then we should find a partitioning that leads to a maximum reduction in the number of nearest neighbors within the smaller subsets (i.e., the average number of signal points that are distance Δ_1^2 away from any point). In principle, the partitioning could be carried out in this heuristic manner.

A more efficient way of partitioning Ω^0 is to require the column vectors of y, i.e., $y^i = [y_1^i, y_2^i]^T$, for $0 \le i \le 2$, to be codewords in a block code. This representation using block codes is also known as multilevel coding (first described by Imai and Hirakawa [33] and later applied to QAM by Cusack [13]). To express this mathematically, we need to introduce some further notation. We define C_{m_i} as that block code which contains the column vectors y^i, for $0 \le i \le 2$. Thus, C_{m_0} contains the least significant bits of y_1 and y_2, C_{m_1} contains the middle bits

of y_1 and y_2, and so on. The actual value of m_i indicates which block code is being used. For L = 2 there are only three block codes that are of interest to us: C_0, which is the (2,2) block code with Hamming distance $d_0 = 1$ (and code words $[0\ 0]^T$, $[0\ 1]^T$, $[1\ 1]^T$, and $[1\ 0]^T$), C_1, which is the (2,1) block code with Hamming distance $d_1 = 2$ (and code words $[0\ 0]^T$ and $[1\ 1]^T$), and C_2, which is the (2,0) block code having only one code word, $[0\ 0]^T$ and Hamming distance $d_2 = \infty$.

Also, since C_{m_i} denotes a block code with 2^{L-m_i} code words, we can write that the partition level p is the sum of all the m_i's that produce the subset Ω^p, i.e., $p = \sum_{i=0}^{2} m_i$. Since there are $I = \log_2 M$ bits needed for each MPSK point, p can range from 0 to IL (0 to 6 in this case). A shorthand way of writing which column vectors y^i belong to which block codes is $\Omega(C_{m_2}, C_{m_1}, C_{m_0})$. Thus, we can write $\Omega^0 = \Omega(C_0, C_0, C_0)$. Since C_0 contains all possible length two binary vectors, then Ω^0 is generated.

To obtain the next partition (at level p = 1), we let $\Omega^1 = \Omega(C_0, C_0, C_1)$. This partition satisfies our previous comments on partitioning. That is, there are only two code words in C_1 (reducing the number of points to 32), and C_1 contains the all zero code word. In partitioning, we also require the property that all the points in Ω^1 belong to Ω^0 (written as $\Omega^1 \subset \Omega^0$). For this example, since $C_1 \subset C_0$, this property is satisfied. This can be stated more generally as $\Omega^{p+1} \subset \Omega^p$, for $0 \le p \le IL-1$. Thus, if we have two partition levels p and p′, and p′ = p+1, then $C_{m_i'} \subseteq C_{m_i}$ for $0 \le i \le I-1$.

The partition Ω^1 is equivalent to forcing the lsb's of y_1 and y_2 to be either both zero or both one. By inspection of Figure 2.1 we can

thus see that $\Delta_1^2 = 2\delta_0^2 = 1.172$. In fact, we can use a more general expression which gives a lower bound on the MSSD. From [27,61] we have,

$$\Delta_p^2 \geq \min\ (\delta_{I-1}^2 d_{m_{I-1}}, \ldots, \delta_1^2 d_{m_1}, \delta_0^2 d_{m_0}), \tag{2.1}$$

where d_{m_i} is the Hamming distance of the code C_{m_i}, for $0 \leq i \leq I-1$. From (2.1), we obtain for 2×8PSK,

$$\Delta_p^2 \geq \min\ (4d_{m_2}, 2d_{m_1}, 0.586d_{m_0}). \tag{2.2}$$

For $p = 0$ and 1, we can see that (2.2) is satisfied with equality. In fact, due to the symmetry of the 8PSK signal set, (2.2) is an equality for all values of p. It can be seen that in partitioning Ω^0 into Ω^1 and its coset $\Omega^1(1)$, we could have formed $\Omega(C_0,C_1,C_0)$ or $\Omega(C_1,C_0,C_0)$ instead of $\Omega(C_0,C_0,C_1)$. However, both these other partitions have $\Delta_1^2 = 0.586$, and are therefore not good partitions, since we want Δ_1^2 to be as large as possible. This is because d_{free}^2 can be lower bounded by $2\Delta_1^2$ for many trellis codes [65].

Ignoring for the moment how the cosets are formed, we can partition Ω^1 into Ω^2 and its coset $\Omega^2(2)$, and so on. (The value within the brackets of the coset will be explained in Section 2.1.3.) Every time we partition, we want to make Δ_p^2 as large as possible. To do this we use the following rule. The C_{m_i} that we partition (into C_{m_i+1}) from level p to level p+1 should be the i corresponding to the smallest $\delta_i^2 d_{m_i}$ at partition level p. If there are two or more $\delta_i^2 d_{m_i}$ that have the smallest value, we choose the one with the smallest i.

Note that once C_{m_i} has been partitioned to C_2 (or C_{I-1} in general), then that particular block code cannot be further partitioned (since it contains only one code word). Table 2.1 illustrates the partitioning of the 2×8PSK signal set. The arrows show which C_{m_i}'s are being partitioned as p is increased. The values of Δ_p^2 are also shown. Note that at p = 3, we have $\delta_i^2 d_{m_i} = 4$ for both i = 1 and 2. As indicated by the above rule, i = 1 is chosen to be partitioned to form Ω^4. Even though $\Delta_4^2 = \Delta_3^2 = 4$, partition level 4 is still useful for coding since the number of nearest neighbors for Ω^4 is less than for Ω^3. This will become more apparent when the actual codes are found.

TABLE 2.1

2×8PSK SIGNAL SET PARTITION

Partition Level (p)	Ω^p	Minimum squared subset distance (Δ_p^2)	Generator $(t^p)^T$
0	$\Omega(C_0, C_0, \underset{\downarrow}{C_0})$	min(4, 2, 0.586) = 0.586	[0 1]
1	$\Omega(C_0, C_0, \underset{\downarrow}{C_1})$	min(4, 2, 1.172) = 1.172	[1 1]
2	$\Omega(C_0, \underset{\downarrow}{C_0}, C_2)$	min(4, 2, ∞) = 2.0	[0 2]
3	$\Omega(C_0, \underset{\downarrow}{C_1}, C_2)$	min(4, 4, ∞) = 4.0	[2 2]
4	$\Omega(\underset{\downarrow}{C_0}, C_2, C_2)$	min(4, ∞, ∞) = 4.0	[0 4]
5	$\Omega(\underset{\downarrow}{C_1}, C_2, C_2)$	min(8, ∞, ∞) = 8.0	[4 4]
6	$\Omega(C_2, C_2, C_2)$	min(∞, ∞, ∞) = ∞	-

The above rule usually works quite well. For L = 3, though, some of the best partitions do not follow this rule. Instead, we can allow a Δ_p^2 to be smaller than the rule proposes, in order to obtain a larger $\Delta_{p'}^2$ for some p′ > p, than is possible by following the rule.

2.1.3 Formation of Cosets

Now consider partition level $p = 1$. We have shown that there are two subsets, namely Ω^1 and its coset $\Omega^1(1)$. To obtain $\Omega^1(1)$, we must look at how coset codes are derived from block codes. Recall that C_1 is the (2,1) block code with Hamming distance $d_1 = 2$. The coset of this code, $C_1(1)$, is formed by adding modulo-2 a non-zero code word that belongs to C_0, but does not belong to C_1 (called the *generator* τ^0), to all the code words in C_1. We illustrate this with an example. C_0 has code words $[0\ 0]^T$, $[0\ 1]^T$, $[1\ 0]^T$, and $[1\ 1]^T$ (remember that these code words correspond to column vectors of **y**) and C_1 has code words $[0\ 0]^T$ and $[1\ 1]^T$. Therefore, the generator τ^0 could equal $[0\ 1]^T$ or $[1\ 0]^T$. We arbitrarily choose $\tau^0 = [0\ 1]^T$. Thus, $C_1(1) = C_1 \oplus \tau^0 = \{[0\ 1]^T, [1\ 0]^T\}$. (In this chapter the symbol $\oplus$ will be used to denote modulo-2 (exclusive-OR) arithmetic and + to denote integer or modulo-M, $M > 2$, arithmetic.) Note that if $\tau^0 = [1\ 0]^T$, the same coset vectors would have been found, except that they would have been in a different order. Also note that the Hamming distance between codewords in $C_1(1)$ is equal to d_1.

We can also write a general expression for the cosets at partition level $p = 1$ as

$$C_1(\zeta^0) = C_1 \oplus \zeta^0\tau^0, \tag{2.3}$$

where $\zeta^0 \in \{0,1\}$. Thus, when $\zeta^0 = 0$, we obtain $C_1(0) \equiv C_1$ and when $\zeta^0 = 1$, we obtain the coset of C_1, $C_1(1)$. In a similar way we can divide C_1 into C_2 and its coset $C_2(2)$, and $C_1(1)$ into cosets $C_2(1)$ and $C_2(3)$. Figure 2.2 gives an illustration of this partition. For the second generator, there is only one choice, i.e., $\tau^1 = [1\ 1]^T$. The

general expression for the cosets at partition level p = 2 becomes

$$\begin{aligned} C_2(2\zeta^1 + \zeta^0) &= C_2 \oplus \zeta^1\tau^1 \oplus \zeta^0\tau^0 \\ &= \zeta^1\begin{bmatrix}1\\1\end{bmatrix} \oplus \zeta^0\begin{bmatrix}0\\1\end{bmatrix}, \end{aligned} \tag{2.4}$$

where C_2 is the all zero vector and $\zeta^m \in \{0,1\}$, for $0 \le m \le 1$. We also note that $C_2 \subset C_1 \subset C_0$ and that $\tau^m \in C_m$, but that $\tau^m \notin C_{m+1}$, for $0 \le m \le 1$.

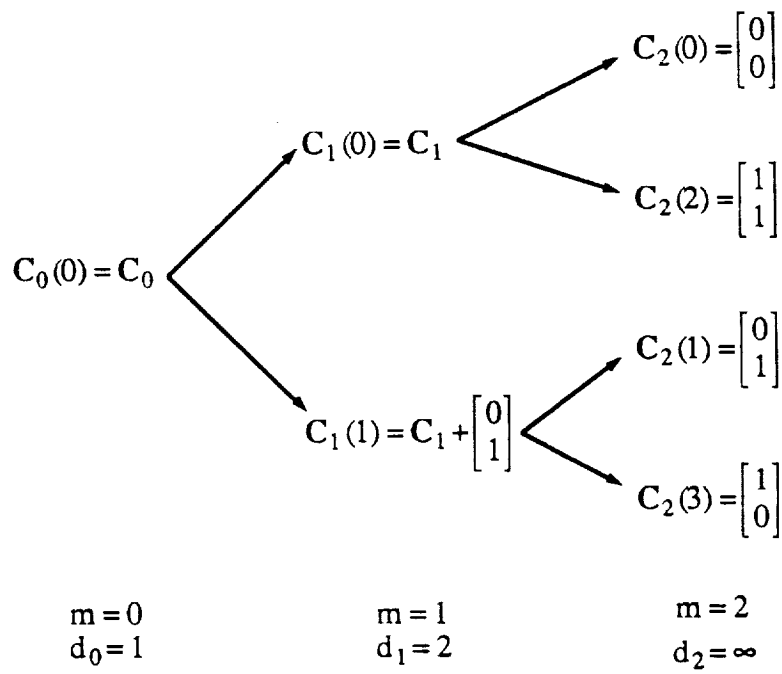

Figure2.2: Partitioning of the L = 2 binary vector space.

Since we have shown how the cosets of C_m are formed, we can now show how the cosets of Ω^p are formed. We start with the simplest case, the single coset of Ω^1 namely $\Omega^1(1)$. In the same way as the block codes are partitioned, we must find a 2×3 matrix that belongs to Ω^0 but does not belong to Ω^1. This is called the *generator* of Ω^1 and is

labeled $\mathbf{t}^0$. Since C_{m_0} is partitioned in going from Ω^0 to Ω^1, this implies that $\mathbf{t}^0 = [\mathbf{0},\mathbf{0},\tau^0]$, where $\mathbf{0}$ is the all zero vector $[0\ 0]^T$, i.e.,

$$\mathbf{t}^0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

An alternate notation for $\mathbf{t}^0$ (using the symbol t^0), is to treat $\mathbf{t}^0$ as if it represented two integer values, y_1 and y_2. Thus, $\mathbf{t}^0$ in integer form is $t^0 = [0\ 1]^T$.

To form the coset $\Omega^1(1)$, all that is required is to add $\mathbf{t}^0$ modulo-2 to all the signal points in Ω^1. We write this as

$$\Omega^1(z^0) = \Omega^1 \oplus z^0\mathbf{t}^0, \tag{2.5}$$

where $z^0 \in \{0,1\}$ indicates which of the two subsets is being selected. We can see that in coset $\Omega^1(1)$, the lsb's of y_1 and y_2 are either 0 and 1 or 1 and 0, respectively. Thus this coset has the same MSSD as Ω^1, i.e., $\Delta_1^2 = 1.172$. Alternately, t^0 can be added modulo-M (modulo-8 in this case) to the signal points in Ω^1. With modulo-8 arithmetic, the lsb's of y_1 and y_2 are still added modulo-2, but the lsb's now produce carries which affect the middle and most significant bits. This is denoted as

$$\Omega^1(z^0) = \Omega^1 + z^0t^0 \pmod 8. \tag{2.6}$$

For example, a signal $y = [1\ 3]^T$ (where $y = [y_1\ y_2]^T$) in Ω^1 becomes $[1\ 2]^T$ with modulo-2 addition of $\mathbf{t}^0$ to $\mathbf{y}$ or $[1\ 4]^T$ with modulo-8 addition of t^0 to y. Using either type of arithmetic, we still obtain

the required partition, although the ordering of signal points within each coset is different. In constructing rotationally invariant trellis codes, we will find that there is a distinct advantage to using modulo-M arithmetic over modulo-2 arithmetic.

Continuing with the set partitioning, it should be obvious that the next generator is $t^1 = [1\ 1]^T$. From Table 2.1, we see that t^1 corresponds to the generator of $\mathbf{C}_1$. The expression for the cosets of Ω^2 is

$$\Omega^2(2z^1 + z^0) = \Omega^2 + z^1\begin{bmatrix} 1 \\ 1 \end{bmatrix} + z^0\begin{bmatrix} 0 \\ 1 \end{bmatrix} \pmod 8, \tag{2.7}$$

where $z^i \in \{0,1\}$, for $0 \le i \le 1$. For partition level $p = 3$, we choose $t^2 = [0\ 2]^T$, with $z^2 \in \{0,1\}$ used to select t^2. Continuing in the same way, we can partition the signal set until we obtain only a single (4-D) signal point. Thus we can form the equation (using the generators from Table 2.1)

$$\begin{aligned} y(z) &= \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \Omega^6(z) \\ &= z^5\begin{bmatrix} 4 \\ 4 \end{bmatrix} + z^4\begin{bmatrix} 0 \\ 4 \end{bmatrix} + z^3\begin{bmatrix} 2 \\ 2 \end{bmatrix} + z^2\begin{bmatrix} 0 \\ 2 \end{bmatrix} + z^1\begin{bmatrix} 1 \\ 1 \end{bmatrix} + z^0\begin{bmatrix} 0 \\ 1 \end{bmatrix} \pmod 8, \end{aligned} \tag{2.8}$$

where $z = \sum_{i=0}^{5} 2^i z^i$, with $z^i \in \{0,1\}$, for $0 \le i \le 5$, and $y(z)$ gives the integer representations of the two 8PSK signal points. The signal set mapping given by z can now be directly used by a convolutional encoder. Since y_1 and y_2 can be described in terms of z, the signal set mapper can be implemented using simple logic circuits (exclusive-OR circuits for modulo-2 addition and binary adders for modulo-M addition). Alternatively, since z can be represented with only six bits, one can

use a small ROM. Figure 2.3 illustrates two possible signal set mappers for 2×8PSK. Figure 2.3(a) shows a mapper using modulo-2 arithmetic, and Figure 2.3(b) shows a mapper using modulo-8 arithmetic.

In general, we can write (2.8) as

$$y(z) = \begin{bmatrix} y_1 \\ \vdots \\ y_L \end{bmatrix} = \Omega^{IL}(z) = \sum_{i=0}^{IL-1} z^i t^i, \tag{2.9}$$

where $z = \sum_{i=0}^{IL-1} 2^i z^i$, with $z^i \in \{0,1\}$, for $0 \le i \le IL-1$. The addition in (2.9) is not specified, but may be modulo-2 (using the binary matrix generators), modulo-M (using the integer generators), or a combination of modulo-2 and modulo-M. Figure 2.4 illustrates the partitioning of Ω^0 into Ω^3 and its cosets $\Omega^3(4z^2 + 2z^1 + z^0)$ for the 2×8PSK signal set using modulo-8 addition.

2.1.4 Partitioning 3×MPSK and 4×MPSK Signal Sets

In a similar fashion to 2×8PSK, to partition L×8PSK (for $L > 2$) requires the partitioning of length $L > 2$ block codes. We again look for partitions that have an increasing Hamming distance. For $L = 3$, there are two partitions that are interesting.

The first partition has Hamming distances $d_0 = 1$, $d_1^1 = 2$, $d_2^1 = 2$, and $d_3 = \infty$. These Hamming distances correspond to the (3,3), (3,2), (3,1), and (3,0) block codes C_0, C_1^1, C_2^1, and C_3, respectively, where $C_3 \subset C_2^1 \subset C_1^1 \subset C_0$. Table 2.2(a) gives the three generators, τ_1^0, τ_1^1, and τ_1^2, that were chosen, along with the Hamming distances (d_m) and the number of nearest neighbors (N_m) at each partition level m. The choice was not completely arbitrary, since one of the generators must be the all ones vector (which in this case is τ_1^0). The reason for this will be

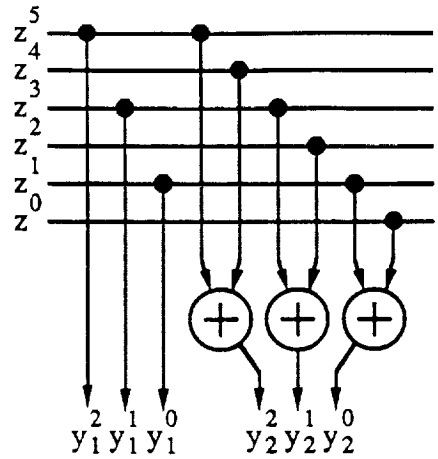

Figure 2.3(a): 2×8PSK signal set mapper with modulo-2 addition.

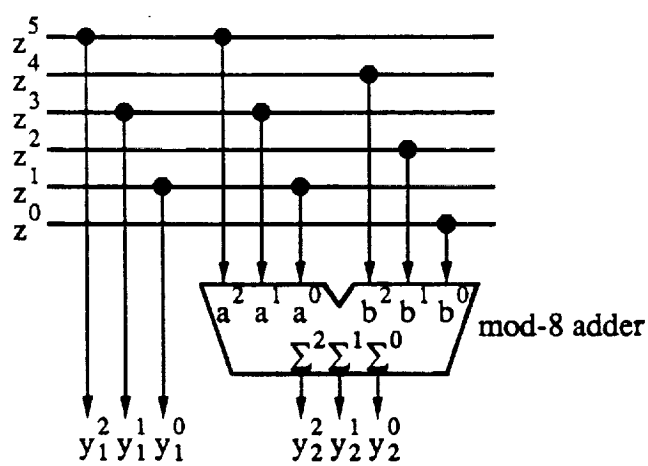

Figure 2.3(b): 2×8PSK signal set mapper with modulo-8 addition.

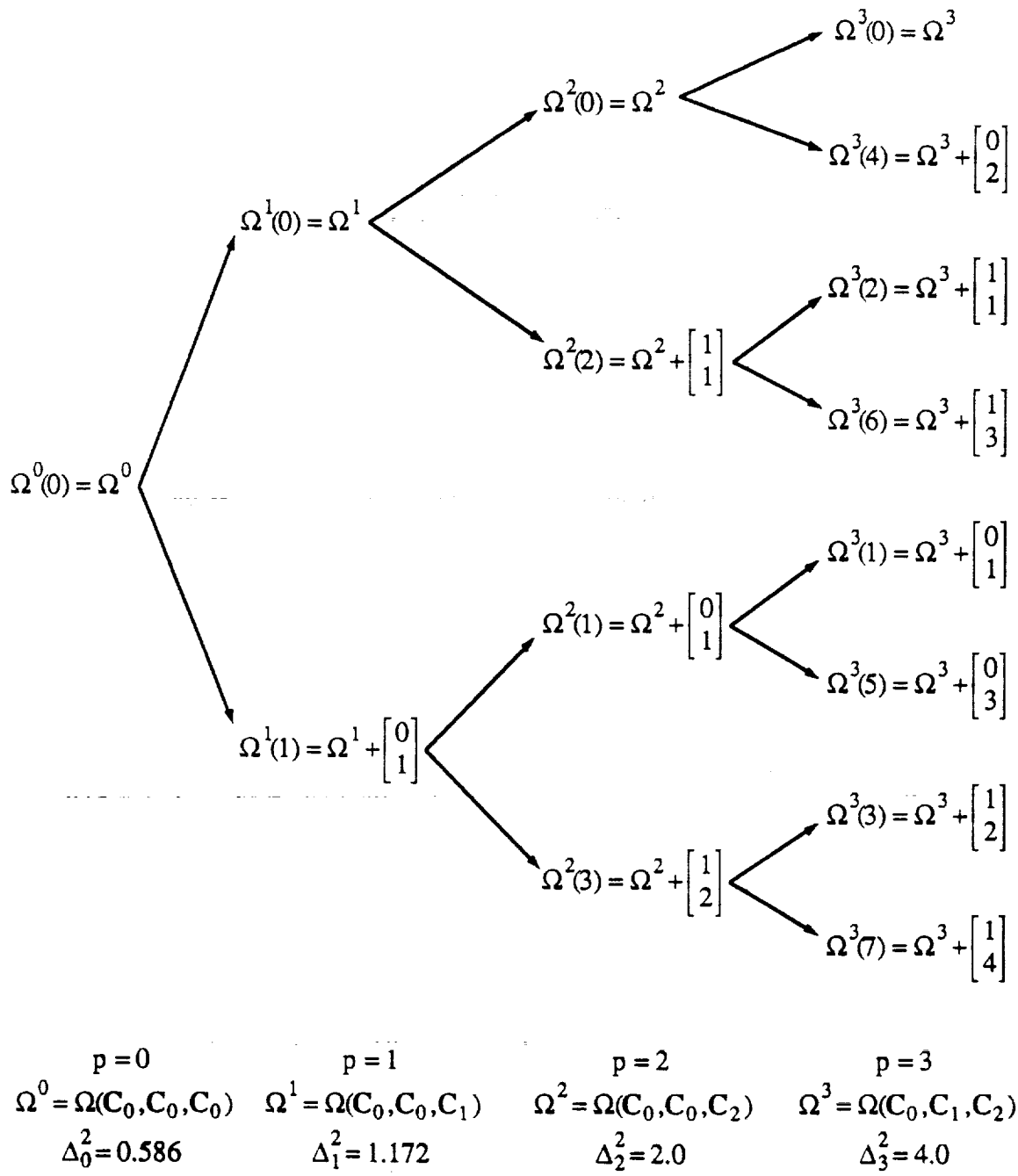

Figure 2.4: A three level 2×8PSK signal set partition.

TABLE 2.2

BINARY GENERATORS FOR L = 3 AND 4

(a) L = 3 (I)

m	d_m	N_m	$(\tau_1^m)^T$
0	1	3	[1 1 1]
1	2	3	[1 1 0]
2	2	1	[0 1 1]

(b) L = 3 (II)

m	d_m	N_m	$(\tau_2^m)^T$
0	1	3	[0 0 1]
1	1	1	[0 1 1]
2	3	1	[1 1 1]

(c) L = 4

m	d_m	N_m	$(\tau^m)^T$
0	1	4	[0 0 0 1]
1	2	6	[0 0 1 1]
2	2	2	[0 1 0 1]
3	4	1	[1 1 1 1]

explained in Section 2.2.

It is interesting to note that the generator matrix for these block codes can be formed from the generators. In general, a generator matrix $\mathbf{G}_m$ for an (L,L-m) block code $\mathbf{C}_m$, for $0 \le m \le L-1$, can be formed from the generators τ^m to τ^{L-1}, i.e., $\mathbf{G}_m = [\tau^m, \tau^{m+1}, \ldots, \tau^{L-1}]^T$. For example, for the L = 3 block codes given in Table 2.2(a),

$$\mathbf{G}_0^1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \ \mathbf{G}_1^1 = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \ \mathbf{G}_2^1 = [\ 0\ 1\ 1\ ].$$

For the other L = 3 partition, we have $d_0 = 1$, $d_1^2 = 1$, $d_2^2 = 3$, and $d_3 = \infty$. These distances correspond to block codes $\mathbf{C}_0$, $\mathbf{C}_1^2$, $\mathbf{C}_2^2$, and $\mathbf{C}_3$, where $\mathbf{C}_3 \subset \mathbf{C}_2^2 \subset \mathbf{C}_1^2 \subset \mathbf{C}_0$. Table 2.2(b) shows the generators for these codes. Note that τ_2^2 is the all ones vector in this case. The advantage of this partition is that $d_2^2 = 3$ is larger than $d_2^1 = 2$.

However, $d_1^2 = 1$ is less than $d_1^1 = 2$.

The partitions of 3×8PSK that will be useful for trellis coding are given in Table 2.3. Table 2.3(a) corresponds to the first partition where we try to maximize Δ_p^2 at each partition level. In Tables 2.3(b) and 2.3(c), the second set of block codes are used to increase Δ_2^2 to 1.757 while Δ_1^2 decreases to 0.586. In Table 2.3(c), Δ_6^2 increases to 6.0 and Δ_4^2 decreases to 2.0. Note how $\Delta_6^2 = 6.0$ is obtained in Table 2.3(c). At $p = 4$ we have $\Delta_4^2 = \min(4.0, 2.0, \infty)$ and at the next partition level, $\Delta_5^2 = \min(4.0, 6.0, \infty) = 4.0$. Now C_{m_2} is partitioned to give $\Delta_6^2 = \min(8.0, 6.0, \infty) = 6.0$. In the next level, we partition C_{m_1} to obtain $\Delta_7^2 = 8.0$. In Section 3, the reasons why these latter two partitions are used will be seen more clearly.

TABLE 2.3(a)

3×8PSK SIGNAL SET PARTITION (I)

Partition Level (p)	Ω^p	Minimum squared subset distance (Δ_p^2)	Generator $(t^p)^T$
0	$\Omega(C_0, C_0, \underset{\downarrow}{C_0})$	$\min(4, 2, 0.586) = 0.586$	[1 1 1]
1	$\Omega(C_0, C_0, \underset{\downarrow}{C_1^1})$	$\min(4, 2, 1.172) = 1.172$	[1 1 0]
2	$\Omega(C_0, C_0, \underset{\downarrow}{C_2^1})$	$\min(4, 2, 1.172) = 1.172$	[0 1 1]
3	$\Omega(C_0, \underset{\downarrow}{C_0}, C_3)$	$\min(4, 2, \infty) = 2.0$	[2 2 2]
4	$\Omega(C_0, \underset{\downarrow}{C_1^1}, C_3)$	$\min(4, 4, \infty) = 4.0$	[2 2 0]
5	$\Omega(C_0, \underset{\downarrow}{C_2^1}, C_3)$	$\min(4, 4, \infty) = 4.0$	[0 2 2]
6	$\Omega(\underset{\downarrow}{C_0}, C_3, C_3)$	$\min(4, \infty, \infty) = 4.0$	[4 4 4]
7	$\Omega(\underset{\downarrow}{C_1^1}, C_3, C_3)$	$\min(8, \infty, \infty) = 8.0$	[4 4 0]
8	$\Omega(\underset{\downarrow}{C_2^1}, C_3, C_3)$	$\min(8, \infty, \infty) = 8.0$	[0 4 4]
9	$\Omega(C_3, C_3, C_3)$	$\min(\infty, \infty, \infty) = \infty$	-

TABLE 2.3(b)

3×8PSK SIGNAL SET PARTITION (II)

Partition Level (p)	Ω^p	Minimum squared subset distance (Δ_p^2)	Generator $(t^p)^T$
0	$\Omega(C_0,C_0,\underset{\downarrow}{C_0})$	min(4, 2, 0.586) = 0.586	[0 0 1]
1	$\Omega(C_0,C_0,\underset{\downarrow}{C_1^2})$	min(4, 2, 0.586) = 0.586	[0 1 1]
2	$\Omega(C_0,C_0,\underset{\downarrow}{C_2^2})$	min(4, 2, 1.757) = 1.757	[1 1 1]
3	$\Omega(C_0,\underset{\downarrow}{C_0},C_3)$	min(4, 2, ∞) = 2.0	[2 2 2]
4	$\Omega(C_0,\underset{\downarrow}{C_1^1},C_3)$	min(4, 4, ∞) = 4.0	[2 2 0]
5	$\Omega(C_0,\underset{\downarrow}{C_2^1},C_3)$	min(4, 4, ∞) = 4.0	[0 2 2]
6	$\Omega(\underset{\downarrow}{C_0},C_3,C_3)$	min(4, ∞, ∞) = 4.0	[4 4 4]
7	$\Omega(\underset{\downarrow}{C_1^1},C_3,C_3)$	min(8, ∞, ∞) = 8.0	[4 4 0]
8	$\Omega(\underset{\downarrow}{C_2^1},C_3,C_3)$	min(8, ∞, ∞) = 8.0	[0 4 4]
9	$\Omega(C_3,C_3,C_3)$	min(∞, ∞, ∞) = ∞	-

For L = 4 there is only one good way to partition length 4 block codes. Table 2.2(c) gives a summary of the basic parameters. Using Table 2.2(c), we can partition the 4×8PSK signal set as shown in Table 2.4.

For L×4PSK and L×16PSK we obtain from (1) that,

$$\Delta_p^2 \geq \min\ (4d_{m_1}, 2d_{m_0}), \tag{2.10a}$$

$$\Delta_p^2 \geq \min\ (4d_{m_3}, 2d_{m_2}, 0.586d_{m_1}, 0.152d_{m_0}), \tag{2.10b}$$

respectively, where $p = \Sigma_{i=0}^{I-1} m_i$ (I = 2 for (2.10a) and I = 4 for (2.10b)). In a similar fashion to L×8PSK, the signal set partitions can be obtained for L = 2 to 4. Tables 2.5, 2.6, and 2.7 give a summary of the partitions for L×4PSK, L×8PSK, and L×16PSK, respectively.

TABLE 2.3(c)

3×8PSK SIGNAL SET PARTITION (III)

Partition Level (p)	Ω^p	Minimum squared subset distance (Δ_p^2)	Generator $(t^p)^T$
0	$\Omega(C_0,C_0,\underset{\downarrow}{C_0})$	min(4, 2, 0.586) = 0.586	[0 0 1]
1	$\Omega(C_0,C_0,\underset{\downarrow}{C_1^2})$	min(4, 2, 0.586) = 0.586	[0 1 1]
2	$\Omega(C_0,C_0,\underset{\downarrow}{C_2^2})$	min(4, 2, 1.757) = 1.757	[1 1 1]
3	$\Omega(C_0,\underset{\downarrow}{C_0},C_3)$	min(4, 2, ∞) = 2.0	[0 0 2]
4	$\Omega(C_0,\underset{\downarrow}{C_1^2},C_3)$	min(4, 2, ∞) = 2.0	[0 2 2]
5	$\Omega(\underset{\downarrow}{C_0},C_2^2,C_3)$	min(4, 6, ∞) = 4.0	[4 4 4]
6	$\Omega(C_1^1,\underset{\downarrow}{C_2^2},C_3)$	min(8, 6, ∞) = 6.0	[2 2 2]
7	$\Omega(\underset{\downarrow}{C_1^1},C_3,C_3)$	min(8, ∞, ∞) = 8.0	[4 4 0]
8	$\Omega(\underset{\downarrow}{C_2^1},C_3,C_3)$	min(8, ∞, ∞) = 8.0	[0 4 4]
9	$\Omega(C_3,C_3,C_3)$	min(∞, ∞, ∞) = ∞	-

2.1.5 Larger Dimensional MPSK Signal Sets and the Squaring Construction

One way to obtain larger dimensional MPSK signal sets is to take an L×MPSK signal set partition (with its corresponding MSSD's relabeled as δ_i^2, for $0 \le i \le IL$) and form a 2LL′ dimensional MPSK signal set which we label as L′×L×MPSK. Thus if we have a 2×8PSK signal set, the MSSD's Δ_p^2, $0 \le p \le 6L'$, for L′×2×8PSK are given by

$$\Delta_p^2 \ge \min\,(8d_{m_5}, 4d_{m_4}, 4d_{m_3}, 2d_{m_2}, 1.172d_{m_1}, 0.586d_{m_0}), \tag{2.11}$$

where the d_{m_i}'s are the Hamming distances of $(L',L'-m_i)$ block codes. If L′ = 2 we can form the 2×2×8PSK signal set, which is equivalent to the

TABLE 2.4

4×8PSK SIGNAL SET PARTITION

Partition Level (p)	Ω^p	Minimum squared subset distance (Δ_p^2)	Generator $(t^p)^T$
0	$\Omega(C_0,C_0,C_0)$	min(4, 2, 0.586) = 0.586	[0 0 0 1]
1	$\Omega(C_0,C_0,\check{C}_1)$	min(4, 2, 1.172) = 1.172	[0 0 1 1]
2	$\Omega(C_0,C_0,\check{C}_2)$	min(4, 2, 1.172) = 1.172	[0 1 0 1]
3	$\Omega(C_0,C_0,\check{C}_3)$	min(4, 2, 2.343) = 2.0	[0 0 0 2]
4	$\Omega(C_0,\check{C}_1,C_3)$	min(4, 4, 2.343) = 2.343	[1 1 1 1]
5	$\Omega(C_0,C_1,\check{C}_4)$	min(4, 4, ∞) = 4.0	[0 0 2 2]
6	$\Omega(C_0,\check{C}_2,C_4)$	min(4, 4, ∞) = 4.0	[0 2 0 2]
7	$\Omega(C_0,\check{C}_3,C_4)$	min(4, 8, ∞) = 4.0	[0 0 0 4]
8	$\Omega(\check{C}_1,C_3,C_4)$	min(8, 8, ∞) = 8.0	[2 2 2 2]
9	$\Omega(C_1,\check{C}_4,C_4)$	min(8, ∞, ∞) = 8.0	[0 0 4 4]
10	$\Omega(\check{C}_2,C_4,C_4)$	min(8, ∞, ∞) = 8.0	[0 4 0 4]
11	$\Omega(\check{C}_3,C_4,C_4)$	min(16, ∞, ∞) =16.0	[4 4 4 4]
12	$\Omega(\check{C}_4,C_4,C_4)$	min(∞, ∞, ∞) = ∞	-

TABLE 2.5

SUMMARY OF L×4PSK PARTITIONS

	L = 2		L=3 (I)		L=3 (II)		L=3 (III)		L = 4	
Partition Level (p)	Δ_p^2	gen. $(t^p)^T$	Δ_p^2	gen. $(t^p)^T$	Δ_p^2	gen. $(t^p)^T$	Δ_p^2	gen. $(t^p)^T$	Δ_p^2	gen. $(t^p)^T$
0	2	01	2	111	2	001	2	001	2	0001
1	4	11	4	110	2	011	2	011	4	0011
2	4	02	4	011	4	222	4	002	4	0101
3	8	22	4	222	6	111	4	022	4	0002
4	-	-	8	220	8	220	6	111	8	1111
5	-	-	8	022	8	022	12	222	8	0022
6	-	-	-	-	-	-	-	-	8	0202
7	-	-	-	-	-	-	-	-	16	2222
p_0 p_1	1	3	0	3	3	2	4	5	4	7

TABLE 2.6

SUMMARY OF L×8PSK PARTITIONS

	L = 2		L=3 (I)		L=3 (II)		L=3 (III)		L = 4	
Partition Level (p)	Δ_p^2	gen. $(t^p)^T$	Δ_p^2	gen. $(t^p)^T$	Δ_p^2	gen. $(t^p)^T$	Δ_p^2	gen. $(t^p)^T$	Δ_p^2	gen. $(t^p)^T$
0	0.586	01	0.586	111	0.586	001	0.586	001	0.586	0001
1	1.172	11	1.172	110	0.586	011	0.586	011	1.172	0011
2	2	02	1.172	011	1.757	111	1.757	111	1.172	0101
3	4	22	2	222	2	222	2	002	2	0002
4	4	04	4	220	4	220	2	022	2.343	1111
5	8	44	4	022	4	022	4	444	4	0022
6	-	-	4	444	4	444	6	222	4	0202
7	-	-	8	440	8	440	8	440	4	0004
8	-	-	8	044	8	044	8	044	8	2222
9	-	-	-	-	-	-	-	-	8	0044
10	-	-	-	-	-	-	-	-	8	0404
11	-	-	-	-	-	-	-	-	16	4444
p_0 p_1 p_2	1 3 5		0 3 6		2 3 6		2 6 5		4 8 11	

4×8PSK signal set. Table 2.8 illustrates this partitioning. Note that the MSSD's obtained are exactly the same as those found with the 4×8PSK partitioning given in Table 2.4. Figure 2.5 shows a block diagram of a signal set mapper for the partition of 2×2×8PSK. The function T_1 corresponds to the mapping given by the generators in Table 2.8 and T_2 to the generators in Table 2.4.

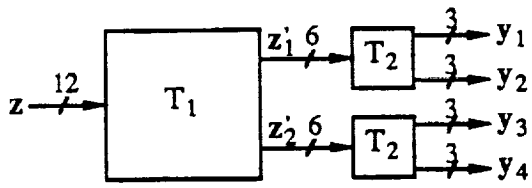

Figure 2.5: Block diagram of 2×2×8PSK signal set mapper.

For L′ = 2, the above method of obtaining larger dimensional MPSK is essentially equivalent to the squaring or two-construction described

TABLE 2.7

SUMMARY OF L×16PSK PARTITIONS

	L = 2		L=3 (I)		L=3 (II)		L=3 (III)		L = 4	
Partition Level (p)	Δ_p^2	gen. $(t^p)^T$	Δ_p^2	gen. $(t^p)^T$	Δ_p^2	gen. $(t^p)^T$	Δ_p^2	gen. $(t^p)^T$	Δ_p^2	gen. $(t^p)^T$
0	0.152	01	0.152	111	0.152	001	0.152	001	0.152	0001
1	0.304	11	0.304	110	0.152	011	0.152	011	0.304	0011
2	0.586	02	0.304	011	0.457	111	0.457	111	0.304	0101
3	1.172	22	0.586	222	0.586	222	0.586	002	0.586	0002
4	2	04	1.172	220	1.172	220	0.586	022	0.609	1111
5	4	44	1.172	022	1.172	022	1.757	222	1.172	0022
6	4	08	2	444	2	444	2	444	1.172	0202
7	8	88	4	440	4	440	4	440	2	0004
8	-	-	4	044	4	044	4	044	2.343	2222
9	-	-	4	888	4	888	4	888	4	0044
10	-	-	8	880	8	880	8	880	4	0404
11	-	-	8	088	8	088	8	088	4	0008
12	-	-	-	-	-	-	-	-	8	4444
13	-	-	-	-	-	-	-	-	8	0088
14	-	-	-	-	-	-	-	-	8	0808
15	-	-	-	-	-	-	-	-	16	8888
$p_0p_1p_2p_3$	1 3 5 7		0 3 6 9		2 3 6 9		2 5 6 9		4 8 12 15	

by Forney [23]. The cubing or three-construction corresponds to $L' = 3$. One can continue squaring or cubing various multi-D signal sets in an iterative fashion to obtain many larger dimensional signal sets. If we desire an L×MPSK signal set, all that is required is to factor L to determine which constructions are needed. For example, if L = 24, we could factor this into a 2×2×2×3×8PSK signal set. If L is a prime number, then the appropriate length L block codes and their corresponding generators must be found.

Table 2.9 gives the generators for L = 5 and 7. Also given are the Hamming distances and the number of nearest neighbors for each length L block code. Note that there are three different partitions for L = 5 and four different partitions for L = 7. This seems to suggest

TABLE 2.8

2×2×8PSK SIGNAL SET PARTITION

p	Ω^p	Δ_p^2	gen. $(t^p)^T$
0	$\Omega(C_0,C_0,C_0,C_0,C_0,C_0)$	min(8, 4, 4, 2, 1.172, 0.586) = 0.586	[0 1]
1	$\Omega(C_0,C_0,C_0,C_0,C_0,\check{C}_1)$	min(8, 4, 4, 2, 1.172, 1.172) = 1.172	[1 1]
2	$\Omega(C_0,C_0,C_0,C_0,C_0,\check{C}_2)$	min(8, 4, 4, 2, 1.172, ∞) = 1.172	[0 2]
3	$\Omega(C_0,C_0,C_0,C_0,\check{C}_1,C_2)$	min(8, 4, 4, 2, 2.343, ∞) = 2.0	[0 4]
4	$\Omega(C_0,C_0,C_0,\check{C}_1,C_1,C_2)$	min(8, 4, 4, 4, 2.343, ∞) = 2.343	[2 2]
5	$\Omega(C_0,C_0,C_0,C_1,\check{C}_2,C_2)$	min(8, 4, 4, 4, ∞, ∞) = 4.0	[4 4]
6	$\Omega(C_0,C_0,C_0,\check{C}_2,C_2,C_2)$	min(8, 4, 4, ∞, ∞, ∞) = 4.0	[0 8]
7	$\Omega(C_0,C_0,\check{C}_1,C_2,C_2,C_2)$	min(8, 4, 8, ∞, ∞, ∞) = 4.0	[0 16]
8	$\Omega(C_0,\check{C}_1,C_1,C_2,C_2,C_2)$	min(8, 8, 8. ∞, ∞, ∞) = 8.0	[8 8]
9	$\Omega(C_0,C_1,\check{C}_2,C_2,C_2,C_2)$	min(8, 8, ∞, ∞, ∞, ∞) = 8.0	[16 16]
0	$\Omega(C_0,\check{C}_2,C_2,C_2,C_2,C_2)$	min(8, ∞, ∞, ∞, ∞, ∞) = 8.0	[0 32]
1	$\Omega(\check{C}_1,C_2,C_2,C_2,C_2,C_2)$	min(16, ∞, ∞, ∞, ∞, ∞) =16.0	[32 32]
2	$\Omega(\check{C}_2,C_2,C_2,C_2,C_2,C_2)$	min(∞, ∞, ∞, ∞, ∞, ∞) = ∞	-

that the number of useful partitions increases by one for each successive prime number. Thus, L = 11 is expected to have five useful partitions, and so on. These partitions were constructed by hand and probably represent the practical limit of hand constructions. For L = 11 and above, an algorithmic or mathematical method is required. In forming each partition, we have tried to maximize the Hamming distance and minimize the number of nearest neighbors. For example, the type IV partition maximizes the Hamming distance and minimizes the number of nearest neighbors for the (7,4) block code while the type III partition maximizes the Hamming distance and minimizes the number of nearest neighbors for the (7,3) and (7,2) block codes.

For larger dimensions, these methods may produce block codes

TABLE 2.9

BINARY GENERATORS FOR L = 5 AND 7

m	L = 5 (I)			L = 5 (II)			L = 5 (III)		
	d_m	N_m	$(\tau_1^m)^T$	d_m	N_m	$(\tau_2^m)^T$	d_m	N_m	$(\tau_3^m)^T$
0	1	5	[11111]	1	5	[11111]	1	5	[00001]
1	2	10	[00011]	1	2	[00001]	1	1	[00010]
2	2	4	[00101]	2	2	[00110]	2	3	[00101]
3	2	1	[11000]	3	2	[10101]	2	1	[01001]
4	4	1	[01111]	4	1	[01111]	5	1	[11111]

m	L = 7 (I)			L = 7 (II)			L = 7 (III)			L = 7 (IV)		
	d_m	N_m	$(\tau_1^m)^T$	d_m	N_m	$(\tau_2^m)^T$	d_m	N_m	$(\tau_3^m)^T$	d_m	N_m	$(\tau_4^m)^T$
0	1	7	[1111111]	1	7	[1111111]	1	7	[0000001]	1	7	[0000001]
1	2	21	[0000011]	1	2	[0000001]	1	1	[0001000]	1	3	[0001000]
2	2	9	[0001001]	2	5	[0000101]	2	6	[1111111]	1	1	[1000000]
3	2	3	[0010010]	2	1	[0100010]	2	2	[0000101]	3	7	[0110100]
4	2	1	[0001100]	3	3	[0011100]	3	2	[0101010]	3	3	[0011010]
5	4	2	[1111000]	4	2	[0001111]	4	1	[1100011]	3	1	[0001101]
6	6	1	[0111111]	6	1	[1110111]	5	1	[0011111]	7	1	[1111111]

which do not have the largest possible minimum distance. For example, the largest Hamming distance that can be obtained for the (24,12) coset code is six. However, the (24,12) Golay code has a Hamming distance of eight. For L = 2, 3, and 4, the block codes are relatively simple. Thus, we are fairly certain that the best partitions for these L×MPSK signal sets have been found.

2.2 Trellis Coded Multi-D MPSK Design

This section describes how convolutional codes are constructed for the L×MPSK signal sets described previously. We first show how to construct signal sets which have good phase rotation properties.

Following this, a method used to find good convolutional codes based on parity check equations is presented.

2.2.1 Construction of Signal Sets

Equation (2.9) can be used to describe a signal point in an L×MPSK signal set. The number of bits z^j needed to describe each signal point is IL. If the lsb is used for coding, we can form a rate (IL-1)/IL code. A more convenient measure of rate is to use the average number of information bits transmitted per 2-D symbol. This is called the *bandwidth efficiency* of the code, K = (IL-1)/L (bit/sym). The unit bit/s/Hz can also be used, but this assumes that perfect Nyquist filtering is used in the receive and transmit filters. Since this is not the case in many practical systems, we make a distinction between the units bit/sym and bit/s/Hz.

Other rates can be achieved by setting the q lsb's of the mapping to 0. We do this to insure that the MSSD's are as large as possible, so that the best codes can be found. In this case (2.9) can be rewritten as

$$y^q(z) = \begin{bmatrix} y_1 \\ \vdots \\ y \end{bmatrix} = \sum_{j=q}^{IL-1} z^{j-q} t^j, \tag{2.12}$$

for $0 \le z \le 2^{IL-q-1}-1$, $0 \le q \le L-1$, and where $y^q(z)$ represents a point z in an L×MPSK signal set such that the first q bits of (2.9) are 0. As before, we do not restrict the type of addition that is used. We now let $z = [z^{IL-q-1},\ldots,z^1,z^0]$, where z is the binary representation of z, and the lsb of z is always the coding bit. This notation insures that the parity check equations of a convolutional code can always be

expressed in terms of the lsb's of z without depending on the type of signal set used or its partitioning. From (2.12), codes with $K = (IL-q-1)/L$ can be formed. An upper limit of $q = L-1$ is set because for $q \geq L$ the signal set is partitioned such that $d_{m_0} = \infty$, i.e., an $M/2^j$-PSK, for $j \geq 1$, signal set is being used (one exception is the 4×8PSK signal set (Table 2.4) where $d_{m_0} = 4$ for $q = L$). The MSSD's range from Δ_q^2 to Δ_{IL}^2 and the uncoded minimum squared Euclidean distance (MSED) is Δ_{q+1}^2, since uncoded transmission uses only half as many signals as coded transmission.

Example 2.1

We can form a rate 4/5 code with a K of 2.0 bit/sym from a 2×8PSK ($L = 2$, $I = 3$) signal set with $q = 1$. Then

$$y^1(z) = z^4\begin{bmatrix} 4 \\ 4 \end{bmatrix} + z^3\begin{bmatrix} 0 \\ 4 \end{bmatrix} + z^2\begin{bmatrix} 2 \\ 2 \end{bmatrix} + z^1\begin{bmatrix} 0 \\ 2 \end{bmatrix} + z^0\begin{bmatrix} 1 \\ 1 \end{bmatrix} \pmod 8.$$

The uncoded MSED is $\Delta_2^2 = 2.0$, which is the same as uncoded 4PSK.

2.2.2 Effect of a 360°/M Phase Rotation on a Multi-D MPSK Signal Set

Using modulo-M arithmetic in (2.12), multi-D signal sets can be constructed such that there are at most I bits in z affected by a signal set rotation of $\Psi \equiv 360°/M$. For 4PSK, 8PSK, and 16PSK, this corresponds to rotations of 90°, 45°, and 22.5°, respectively. Initially, we consider all possible mapped bits, i.e., $q = 0$.

Consider that a 1×MPSK signal set has been rotated by Ψ. Since we are using natural mapping, the integer representation of the rotated

signal point is $y_r = y + 1 \pmod{M}$, where y is the integer representation of the signal point before rotation. If y is in binary notation, then

$$y_r^0 = y^0 \oplus 1 = \overline{y^0}, \tag{2.13a}$$

$$y_r^1 = y^1 \oplus y^0, \tag{2.13b}$$

$$y_r^2 = y^2 \oplus y^0 \cdot y^1, \tag{2.13c}$$

$$\vdots$$

If there are $I = \log_2 M$ bits in a signal set, then we see from (2.13) that all I bits are affected by a phase rotation of Ψ.

Consider the 2×8PSK signal set, with the mapping given by (2.8). The phase rotation equations of this mapping can be determined as follows. From (2.8), the signal outputs can be written in terms of z as

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = (4z^5 + 2z^3 + z^1)\begin{bmatrix} 1 \\ 1 \end{bmatrix} + (4z^4 + 2z^2 + z^0)\begin{bmatrix} 0 \\ 1 \end{bmatrix} \pmod{8}. \tag{2.14}$$

After a 45° phase rotation we have $y_{j,r} = y_j + 1 \pmod{8}$, for $j = 1,2$. From (2.14), we can form the following phase rotation equations,

$$\begin{bmatrix} y_{1,r} \\ y_{2,r} \end{bmatrix} = (4z^5 + 2z^3 + z^1 + 1)\begin{bmatrix} 1 \\ 1 \end{bmatrix} + (4z^4 + 2z^2 + z^0)\begin{bmatrix} 0 \\ 1 \end{bmatrix} \pmod{8}.$$

Note that a 1 is added to the term whose coset is $[1\ 1]^T$. Hence this term "absorbs" the effect of the phase rotation, leaving the remaining term unaffected. As can be seen, bits z^5, z^3, and z^1 are affected in a manner similar to y^2, y^1, and y^0 in (2.13), and bits z^4, z^2, and z^0 are unaffected by the phase rotation. Thus, we can form the phase rotation equations

$$z_r^0 = z^0, \qquad z_r^2 = z^2, \qquad z_r^4 = z^4,$$
$$z_r^1 = z^1 \oplus 1, \quad z_r^3 = z^3 \oplus z^1, \quad z_r^5 = z^5 \oplus z^1 \cdot z^3. \tag{2.15}$$

If the signal set had been constructed using modulo-2 addition (instead of modulo-8), only z^0 would have remained unchanged by a 45° phase rotation.

Using general notation, we can express (2.14) as

$$\begin{bmatrix} y_1 \\ \vdots \\ y_L \end{bmatrix} = (2^{I-1} z^{p_{I-1}} + \cdots + 2z^{p_1} + z^{p_0}) \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} +$$
$$+ 2^{I-1}\left\{g_{I-1}\right\} + \cdots + 2\left\{g_1\right\} + \left\{g_0\right\} \pmod{M}, \tag{2.16}$$

where p_j, for $0 \le j \le I-1$, corresponds to those partition levels where t^p equals the vector $[2^j, 2^j, \ldots, 2^j]^T$. The term g_j, for $0 \le j \le I-1$, corresponds to those remaining terms that have at least one (but not all) component in t^p with value 2^j. For (2.14) we would have $p_0 = 1$, $p_1 = 3$, and $p_2 = 5$. These values of p_j are given for all the signal set partitions shown in Tables 2.5-2.7. We can now write the phase rotation equations as

$$z_r^{p_0} = z^{p_0} \oplus 1, \quad z_r^{p_1} = z^{p_1} \oplus z^{p_0}, \quad z_r^{p_2} = z^{p_2} \oplus z^{p_0} \cdot z^{p_1}, \ldots \tag{2.17}$$

and for all other partition levels, $z_r^p = z^p$.

For $L = 2$, there is only one term in each g_j. However, for $L \ge 3$, there are two or more terms in each g_j. Since the terms in g_j do not contribute to the phase rotational properties of the signal mapping, these terms can be added modulo-2 before being added modulo-M to the

other terms. This is best illustrated with an example. For the 3×8PSK (I) signal set in Table 2.3(a), we have the following mapping equation:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = z^8\begin{bmatrix} 0 \\ 4 \\ 4 \end{bmatrix} + z^7\begin{bmatrix} 4 \\ 4 \\ 0 \end{bmatrix} + z^6\begin{bmatrix} 4 \\ 4 \\ 4 \end{bmatrix} + z^5\begin{bmatrix} 0 \\ 2 \\ 2 \end{bmatrix} + z^4\begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix} +$$

$$+ z^3\begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix} + z^2\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} + z^1\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} + z^0\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$= (4z^6 + 2z^3 + z^0)\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + 4\left\{z^8\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \oplus z^7\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}\right\} +$$

$$+ 2\left\{z^5\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \oplus z^4\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}\right\} + \left\{z^2\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \oplus z^1\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}\right\} \pmod 8$$

$$= (4z^6 + 2z^3 + z^0)\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + 4\begin{bmatrix} z^7 \\ z^8 \oplus z^7 \\ z^8 \end{bmatrix} + 2\begin{bmatrix} z^4 \\ z^5 \oplus z^4 \\ z^5 \end{bmatrix} + \begin{bmatrix} z^1 \\ z^2 \oplus z^1 \\ z^2 \end{bmatrix} \pmod 8.$$

The reason for this combination of modulo-2 and modulo-M arithmetic is that it reduces the number of logic circuits required in a signal set mapper. For small IL, it may be simpler to use ROM's for signal set mapping, but for large IL this dual addition becomes preferable. Figure 2.6 gives a block diagram of the three 3×8PSK signal set mappers and Figure 2.7 illustrates the mapper for 4×8PSK. This combination of modulo-2 and modulo-M addition has no effect on the MSSD's (at least for $L \leq 4$). In a similar manner, we can also obtain the signal set mappers for L×4PSK and L×16PSK.

Due to the phase rotational properties and simplified hardware that the combined modulo-2 and modulo-M mapping allows, these are the signal sets that are used to find all the trellis codes in this chapter.

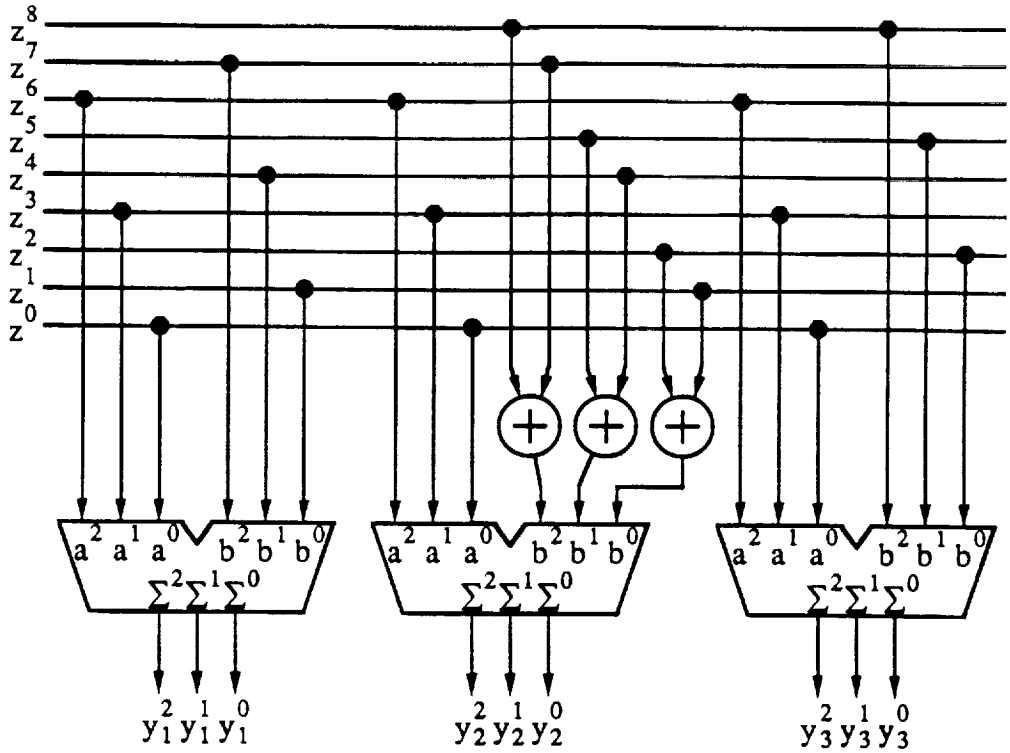

Figure 2.6(a): 3×8PSK signal set mapper (I).

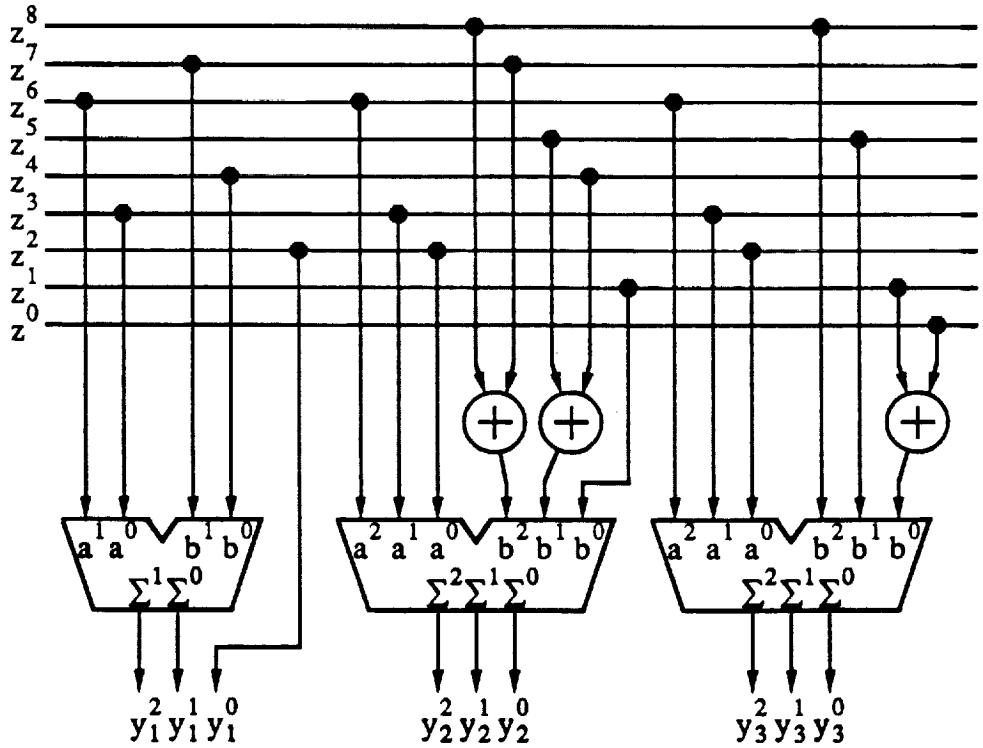

Figure 2.6(b): 3×8PSK signal set mapper (II).

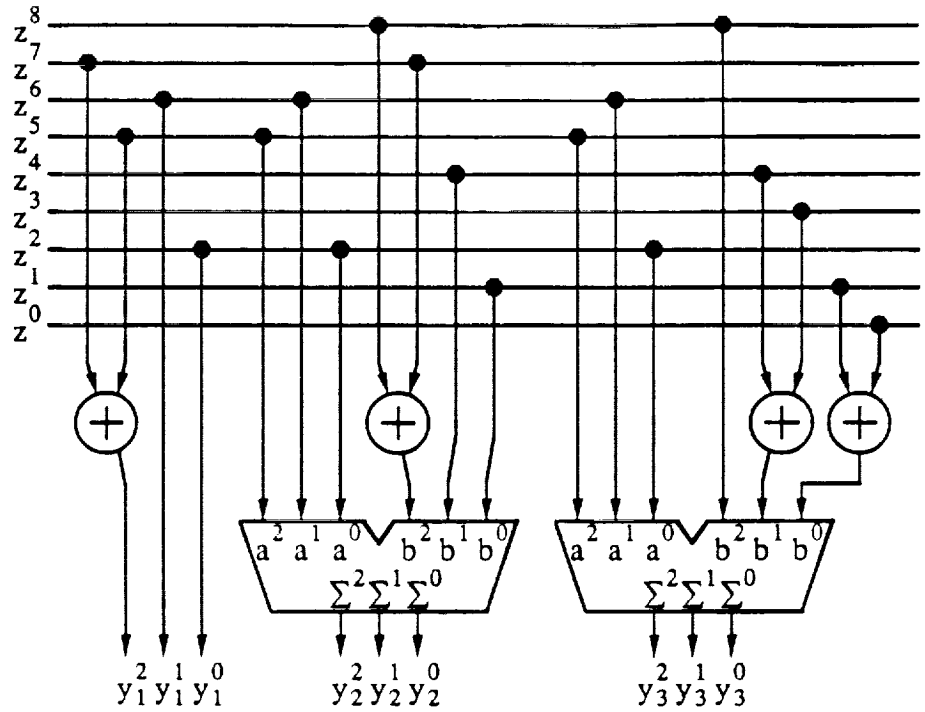

Figure 2.6(c): 3×8PSK signal set mapper (III).

We have shown that for $q = 0$, the bits that are affected by a phase rotation of Ψ are z^{p_j}, for $0 \le j \le I-1$. For $q > 0$ the bits that are affected are $z^{p_j - q}$, for $0 \le j \le I-1$. However, depending on the signal set, $p_j - q$ for some j may be less than zero. If this is true, the minimum phase transparency is $2^{d'}\Psi$, where d' is the number of terms $p_j - q$ that are less than zero, and the number of bits that are affected by a $2^{d'}\Psi$ phase rotation is $s' = I - d'$. For example, the 3×8PSK signal set in Table 2.3(a) has $p_0 = 0$, $p_1 = 3$, and $p_2 = 6$. Thus if $q = 1$, then $p_0 - q = -1$, which is less than zero, implying that $d' = 1$, and thus only $s' = I - d' = 2$ bits are affected by a $2\Psi = 90°$ phase rotation. (A phase rotation of $\Psi = 45°$ of this signal set produces its coset.)

Fortunately, for the codes and signal sets considered in this chapter, the above complication does not occur. This is partly due to the fact that for many signal sets with $q = 0$, the first L-1 lsb's are

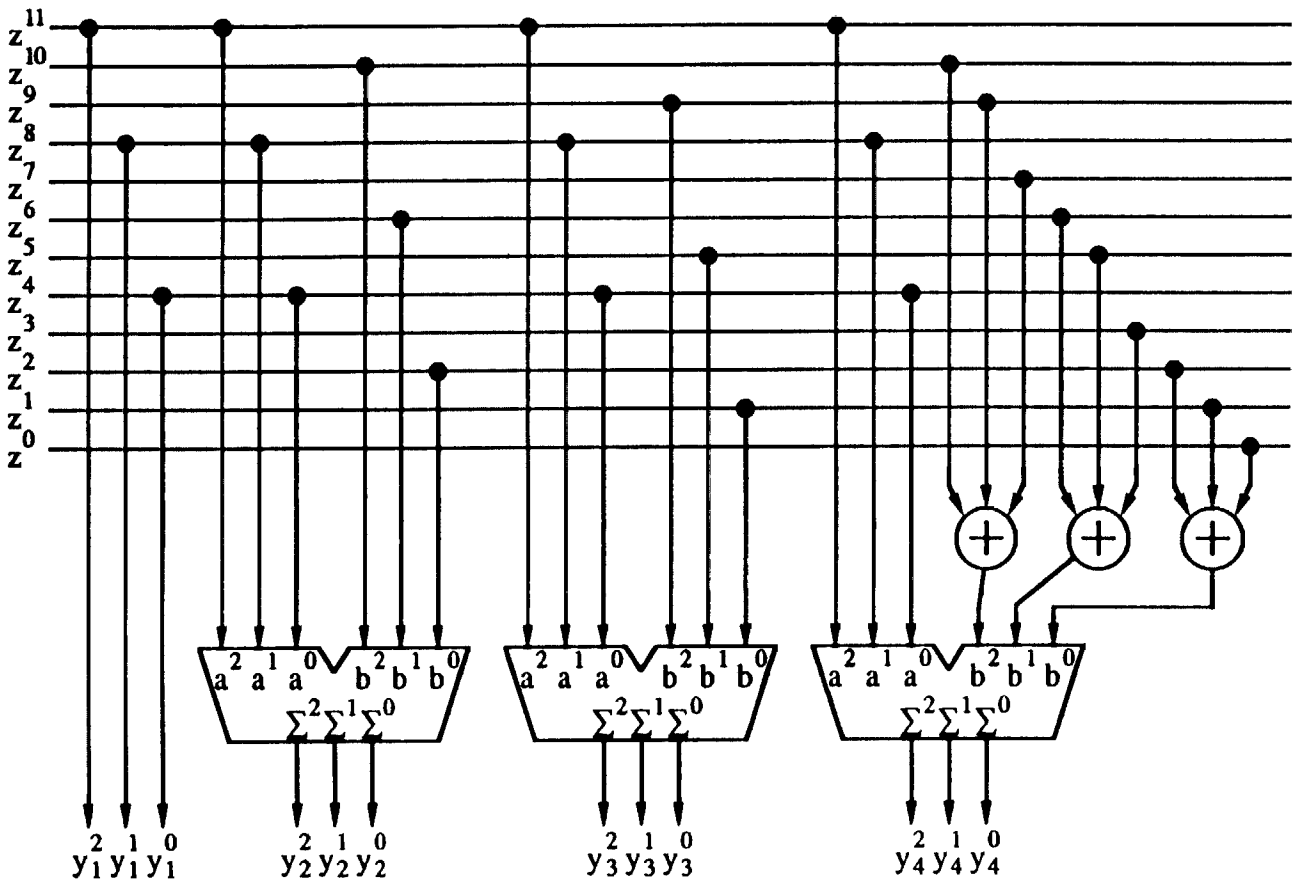

Figure 2.7: 4×8PSK signal set mapper.

not affected by a phase rotation of Ψ. Since we consider only signal sets with $0 \le q \le L-1$, $d' = 0$ in these cases. For those signal sets where this is not true (e.g., in some 3×MPSK signal sets), it has been found that the convolutional codes produced are inferior (in either d_{free} or number of nearest neighbors) to an alternative signal set with $d' = 0$.

When a signal set is combined with a convolutional encoder we must consider the effect of rotating coded sequences. A similar result to above is obtained so that, depending on the code and the signal set, the signal set can be rotated in multiples of $2^d\Psi$ and still produce valid code sequences (where d defines the degree of transparency). The actual determination of d is described in Section 2.2.4. The number of bits that are affected by a $2^d\Psi$ phase rotation is $s = I - d$.

For $0 \le q \le L-1$, the actual bits that are affected by a phase rotation of Ψ are z^{b_j}, where $b_j = p_j - q$, for $0 \le j \le I-1$. More generally, the bits that are affected by a phase rotation of $2^d\Psi$ are z^{c_j}, where $c_j = p_{j+d} - q$, for $0 \le j \le s-1$. These two separate notations (b_j and c_j) are used because the determination of d depends on b_j, as will be shown in Section 2.2.4.

2.2.3 The General Encoder System

From the above information we can now construct a suitable encoder system, as illustrated in Figure 2.8. The general encoder system consists of five sections. These sections are the differential encoder (or precoder), the binary convolutional encoder, the multi-D signal set mapper, the parallel to serial converter, and the 2-D signal set mapper.

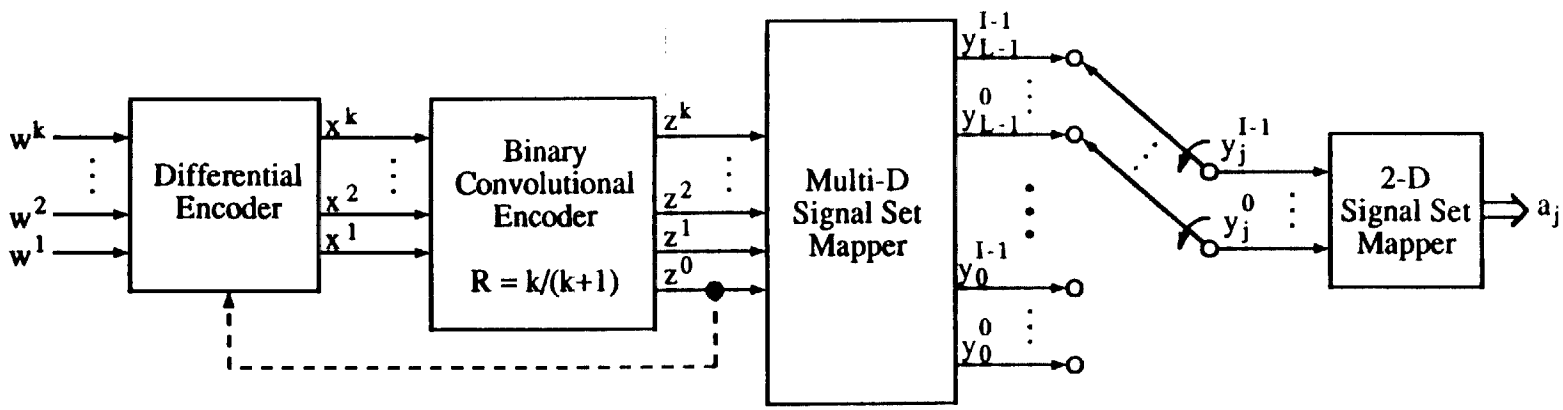

Figure 2.8: General encoder system.

The precoder codes only those bits which are affected by a phase rotation. The input bits into the encoder which are precoded are denoted w^{c_0}, w^{c_1},..., $w^{c_{s-1}}$. If $c_0 = 0$, we replace w^0 (which does not exist) by z^0, as shown in Figure 2.8 by the dashed line (a different precoder must then be used). For example, an encoder for a rate 8/9 code which uses the 3×8PSK (I) signal set given in Table 2.3(a) may (depending on the phase transparency) need this modification. This is because this signal set has $b_0 = 0$, and thus if the code has $d = 0$, then z^0 will need to be precoded.

The multi-D signal set mapper can be implemented as described in Section 2.2.2. We must insure that the correct labels are used to map the signal set if q is greater than zero. All the labels in Figures 2.4, 2.7, and 2.8 assume that $q = 0$.

The second to last section of the encoder is the parallel to serial converter, which takes the L groups of I bits and forms a stream with I bits in each group. That is, we assume that the channel is limited to transmitting one 2-D signal point at a time. Finally, the 2-D signal set mapper takes the I bits for each 2-D signal point and produces the required real and imaginary (or amplitude and phase) components for a modulator.

At this point, we summarize the notation and indicate the limits on the parameters used in the search for good codes. For a rate (IL-q-1)/(IL-q) code,

I = no. of bits in each 2-D signal ($2 \leq I \leq 4$),

$M = 2^I$ = no. of signal points in each 2-D signal set,

L = no. of 2-D signal sets ($1 \leq L \leq 4$),

p = partition level of signal set ($0 \le p \le IL$),

q = the partition level p where mapping begins ($0 \le q \le L-1$),

z = signal set mapping parameter ($0 \le z \le 2^{p-q}-1$),

$k = IL-q-1$ = no. of input bits to encoder,

$\tilde{k}$ = no. of bits checked by encoder ($1 \le \tilde{k} \le k$),

$\Psi = 360°/M$ = minimum phase transparency with $q = 0$,

p_j = the bits z^{p_j} affected by a Ψ phase rotation with $q = 0$,

d = degree of phase transparency ($2^d\Psi$, for $0 \le d \le I$),

$s = I-d$ = no. of bits in z affected by a $2^d\Psi$ phase rotation,

$c_j = p_{j+d}-q$ = the bits z^{c_j} affected by a $2^d\Psi$ phase rotation.

The following two sections describe the precoder and encoder design in more detail.

2.2.4 Differential Encoding and Decoding

Let the bit streams that are differentially encoded be $w^{c_0}(D)$, $w^{c_1}(D)$,..., $w^{c_{s-1}}(D)$, where D is the delay operator.. We first assume that $c_0 > 0$ (i.e., the convolutional encoder output $z^0(D)$ is not affected by a phase rotation of $2^d\Psi$, where $d = I-s$). Let

$$w(D) = \sum_{i=0}^{s-1} 2^i w^{c_i}(D). \tag{2.18}$$

The precoder outputs are the bit streams $x^{c_0}(D)$, $x^{c_1}(D)$,..., $x^{c_{s-1}}(D)$ which go into the convolutional encoder. Similar to (2.18), we let

$$x(D) = \sum_{i=0}^{s-1} 2^i x^{c_i}(D). \tag{2.19}$$

For the noiseless channel, we let the Viterbi decoder output which goes into the differential decoder (or postdecoder) be $x_r(D)$, and the output from the postdecoder be $w_r(D)$. After a $2^d\Psi$ phase rotation, we have from Section 2.2.2 that

$$x(D) = x(D) + 1(D) \pmod{S}, \tag{2.20}$$

where $S = 2^s$ and $1(D)$ is the all ones sequence. For the postdecoder, we desire that $w_r(D) = w(D)$ for all multiples of $2^d\Psi$ phase rotations. This is achieved by defining the postdecoder equation as

$$w_r(D) = ((S-1)D + 1)x_r(D) \pmod{S}. \tag{2.21}$$

Substituting (2.20) into (2.21) we obtain

$$\begin{aligned} w_r(D) &= ((S-1)D + 1)(x(D) + 1(D)) && \pmod{S} \\ &= ((S-1)D + 1)x(D) + ((S-1)D + 1)1(D) && \pmod{S} \\ &= w(D) + (S-1)1(D) + 1(D) && \pmod{S} \\ &= w(D) + (S)1(D) && \pmod{S} \\ &= w(D), \end{aligned}$$

as required. Notice that since $1(D)$ is defined to be 1 for all time, then $D^i1(D) = 1(D)$ for all i. In practical situations, the sequence added to $x(D)$ to form $x_r(D)$ is not constant, and will change with time (e.g., random phase slips within a demodulator). This will introduce short error bursts in $w_r(D)$ whenever a phase slip occurs due to the combined effect of decoding and postdecoding. The precoder equation can be derived from (2.21) as

$$x(D) = Dx(D) + w(D) \pmod{S}. \tag{2.22}$$

We shall now consider the case when $c_0 = 0$, i.e., $z^0(D)$ is affected by a $2^d\Psi$ phase rotation. In this case we redefine w(D) to be

$$w(D) = \sum_{i=1}^{s-1} 2^{i-1} w^{c_i}(D), \tag{2.23}$$

and x(D) to be

$$x(D) = \sum_{i=1}^{s-1} 2^{i-1} x^{c_i}(D). \tag{2.24}$$

For this case, we have $2x_r(D) + z_r^0(D) = 2x(D) + z^0(D) + 1(D)$, where $x_r(D)$ and $z_r^0(D)$ are the inputs to the postdecoder for a noiseless channel. Thus, similar to (2.21), the postdecoder equation is defined to be

$$2w_r(D) = ((S-1)D + 1)(2x_r(D) + z_r^0(D)) \pmod{S}. \tag{2.25}$$

Rearranging (2.25), we obtain the precoder equation

$$2x(D) = 2Dx(D) + 2w(D) + (D + S-1)z^0(D) \pmod{S}. \tag{2.26}$$

Figure 2.9 illustrates the two types of precoders. Note that the storage elements have a delay of LT. Figure 2.9(a) illustrates the precoder with $c_0 > 0$, where there are s inputs that are precoded. The basic component of the precoder is the modulo-2^s adder. For most codes this is the precoder to be used. For the bits that are not precoded, $x^i = w^i$, for $i \neq c_j$.

Figure 2.9(b) shows the other case, where $c_0 = 0$ and s-1 input bits are precoded (the other precoded bit is z^0). The adder circuit for this case is different from Figure 2.9(a).

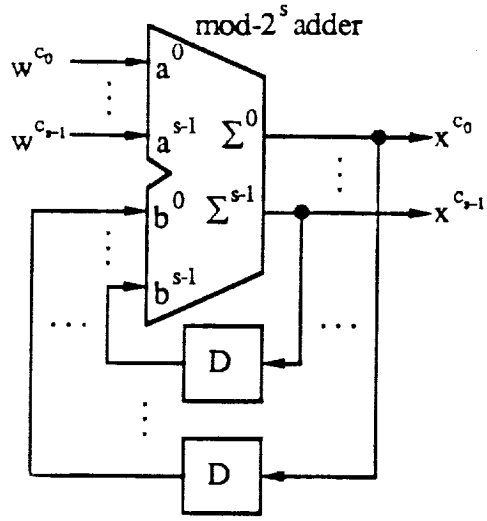

Figure 2.9(a): Differential encoder for $c_0 > 0$.

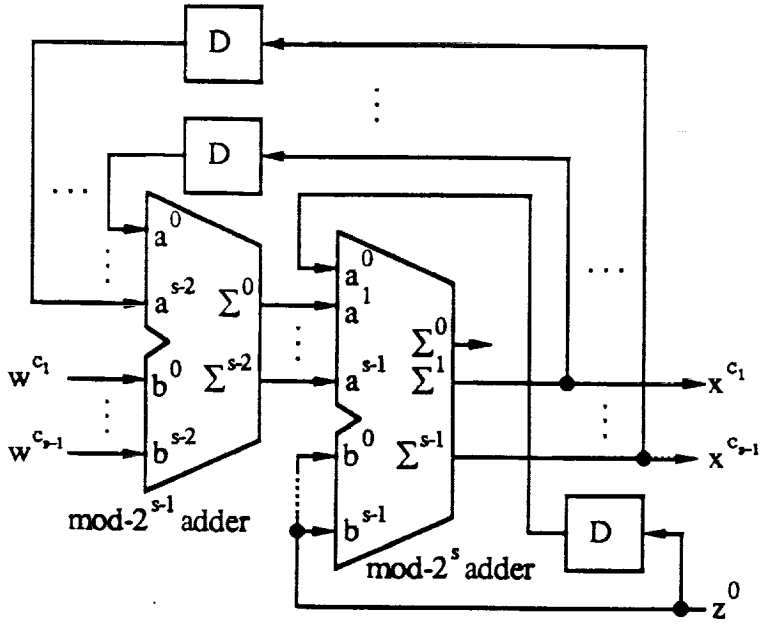

Figure 2.9(b): Differential encoder for $c_0 = 0$.

2.2.5 Convolutional Encoder

The convolutional encoder is assumed to be in feedback systematic form, as in [65]. That is, $z^j(D) = x^j(D)$ for $1 \le j \le k$, where D is the delay operator and polynomial notation is used. The parity sequence, $z^0(D)$, will be some function of itself and the $x^j(D)$, for $1 \le j \le k$. The Parity Check Equation (PCE) of an encoder describes the relationship in time of the encoded bit streams. It is a very useful and efficient means of describing high rate convolutional codes, since it represents the input/output encoder relationships in a single equation. For an $R = k/(k+1)$ code, the parity check equation is

$$H^{\tilde{k}}(D)z^{\tilde{k}}(D) \oplus \cdots \oplus H^1(D)z^1(D) \oplus H^0(D)z^0(D) = 0(D), \tag{2.27}$$

where $\tilde{k}$, $1 \le \tilde{k} \le k$, is the number of input sequences that are checked by the encoder, $H^j(D)$, for $0 \le j \le \tilde{k}$, is the parity check polynomial of $z^j(D)$, and $0(D)$ is the all zero sequence.

There are two types of systematic convolutional encoders that can be constructed. Before proceeding with the description of these encoders, we return to the parity check equation given in (2.27). As in [65], we define the *constraint length* ν to be the maximum degree of all the parity check polynomials $H^j(D)$, for $0 \le j \le \tilde{k}$. For $\tilde{k} < j \le k$, $H^j(D) = 0$, since the bits corresponding to these polynomials are not checked by the encoder. The parity check polynomials are of the form

$$H^j(D) = 0 \oplus h^j_{\nu-1}D^{\nu-1} \oplus \cdots \oplus h^j_1 D \oplus h^j_0, \quad 1 \le j \le \tilde{k}, \tag{2.28a}$$

$$H^0(D) = D^\nu \oplus h^0_{\nu-1}D^{\nu-1} \oplus \cdots \oplus h^0_1 D \oplus 1. \tag{2.28b}$$

If $\tilde{k} < \nu$, we let $h^j_0 = 0$, for $1 \le j \le \tilde{k}$. This insures that the squared

Euclidean distance (SED) between paths in a trellis leaving or entering a state is at least Δ^2_{q+1}. Thus all codes in this class have a MSED between all possible non-parallel coded sequences of at least $2\Delta^2_{q+1}$. The parallel transitions provide an upper bound on the d_{free} of a code. A theoretical justification for constructing codes in this manner can be found in [60] where it is shown, using random coding arguments, that these codes have a large free MSED on the average.

A minimal systematic encoder can be implemented from (2.28), since $h^0_0 = 1$ [65]. The encoding equations are

$$z^j(D) = x^j(D), \quad 1 \le j \le k, \tag{2.29a}$$

$$z^0(D) = H^{\tilde{k}}(D)x^{\tilde{k}}(D) \oplus \cdots \oplus H^1(D)x^1(D) \oplus (H^0(D) \oplus 1)z^0(D). \tag{2.29b}$$

An encoder implementation using (2.29) is shown in Figure 2.10.

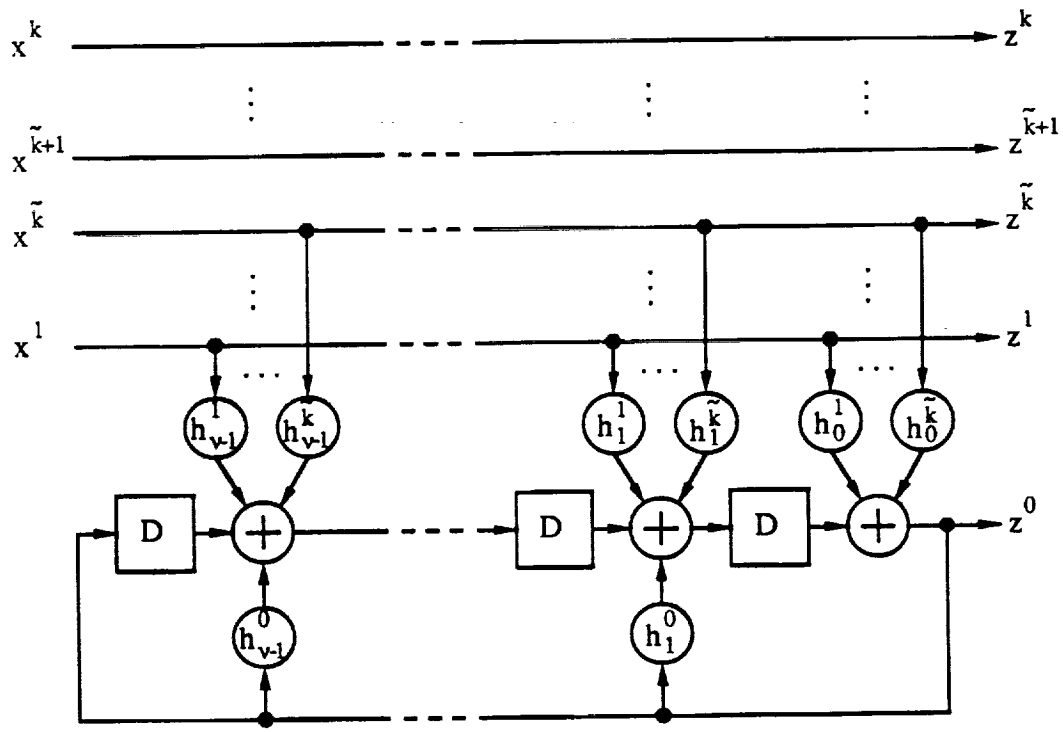

Figure 2.10: Systematic convolutional encoder with $\tilde{k}$ checked bits.

For all codes with $\nu = 1$ and for some codes with $\nu > 1$, $\tilde{k} = \nu$. For these codes we cannot restrict h_0^j, for $1 \le j \le \tilde{k}$. This is because $\tilde{k}$ checked bits require at least $\tilde{k}$ terms in $H^j(D)$, for $1 \le j \le \tilde{k}$, that are variable. If there are not enough variables, then there will be some non-zero $x^{\tilde{k}} = [x^{\tilde{k}},\ldots,x^2,x^1]$ such that $\sum_{j=1}^{\tilde{k}} H^j(D)x^j = 0 \pmod 2$. That is, there will be more than $2^{k-\tilde{k}}$ parallel transitions between states in the trellis. To avoid this problem, when $\tilde{k} = \nu$, we use (2.28) without any restrictons. In this case, the MSED between all possible non-parallel coded sequences is at least $\Delta_q^2 + \Delta_{q+1}^2$, since the MSED between paths leaving a state is Δ_q^2 (since $h_0^j \in \{0,1\}$, for $1 \le j \le \tilde{k}$) and between paths entering a state is Δ_{q+1}^2 (since $h_\nu^j = 0$, for $1 \le j \le \tilde{k}$).

An algorithm in [57] allows the conversion of the systematic form of the encoder to a non-systematic form. There are usually a number of non-systematic encoders to choose from which give the same PCE as the systematic encoder. The encoder selected should have the same effect on a phase rotation as the systematic encoder, in order to be compatible with the precoder.

Example 2.2

In this example, we describe how to implement a particular code. The code is used with a 3×8PSK signal set. Thus $L = 3$ and $I = 3$. We also choose $q = 1$, so that a 2.33 bit/sym (rate 7/8) code is formed. The partition that is used is given in Table 2.3(b), from which we obtain $p_0 = 2$, $p_1 = 3$, and $p_2 = 6$. The code is 90° transparent, so that $d = 1$ and $s = 2$. Therefore $c_0 = p_1 - q = 2$, and $c_1 = p_2 - q = 5$. Thus bits w^2 and w^5 are precoded using a modulo-4 adder. Since $c_0 > 0$, the

precoder given in Figure 2.9(a) is used. For this code, $\tilde{k} = 2$ and the parity check polynomials are $H^0(D) = D^4 \oplus D^2 \oplus D \oplus 1$, $H^1(D) = D$, and $H^2(D) = D^3 \oplus D^2$. Excluding the parallel to serial converter and the 2-D signal mapper, the encoder is shown in Figure 2.11. This code has 16 states ($\nu = 4$). Note that the multi-D signal set mapper does not correspond exactly to Figure 2.6(b), since $q = 1$.

2.2.6 Convolutional Encoder Effects on Transparency

The convolutional encoder can affect the total transparency of the system. The method used to determine transparency is to examine the parity check equation and the bits that are affected by a phase rotation. A code is transparent if its parity check equation, after substituting $z^j(D)$ with $z_r^j(D)$, for $0 \le j \le \tilde{k}$ (the rotated sequences), remains the same. There are normally at most I bits that are affected by a phase rotation, $z^{b_0},\ldots,z^{b_{I-1}}$, $b_j = p_j - q$, for $0 \le j \le I-1$. That is,

$$z_r^{b_0} = z^{b_0} \oplus 1, \tag{2.30a}$$

$$z_r^{b_1} = z^{b_1} \oplus z^{b_0}, \tag{2.30b}$$

$$z_r^{b_2} = z^{b_2} \oplus z^{b_0} \cdot z^{b_1}, \tag{2.30c}$$

$$\vdots$$

Assume that the largest value of $b_j \le \tilde{k}$ is b_0. This implies that only one term in the parity check equation is affected by a phase rotation. The other bits have no effect since they are not checked by the encoder, i.e., $b_j > \tilde{k}$ for $1 \le j \le I-1$. The parity check equation after a phase rotation of Ψ then becomes

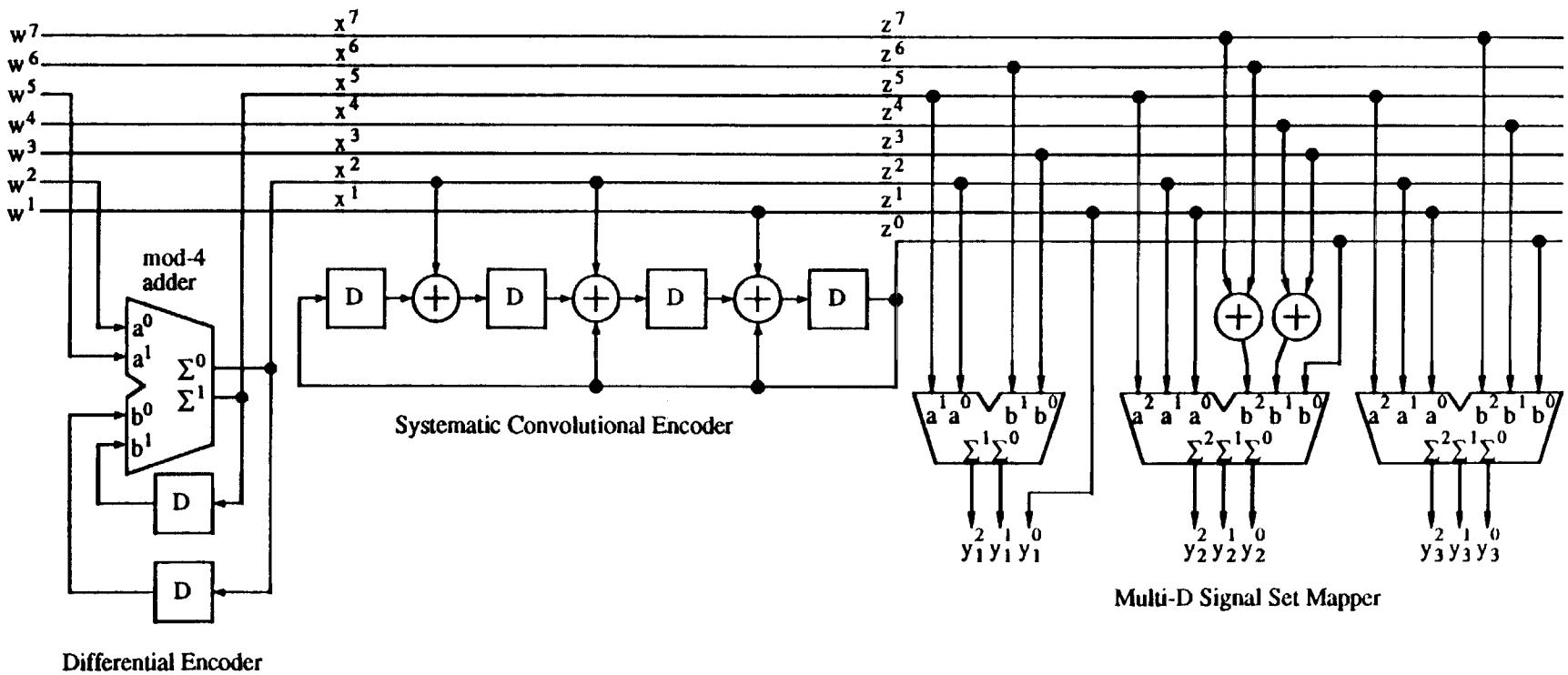

Figure 2.11: Encoder system for a rate 7/8 (2.33 bit/sym), 3×8PSK (II) signal set and 90° transparent code with 16 states and $\tilde{k} = 2$.

$$H^{\bar{k}}(D)z^{\bar{k}}(D) \oplus \cdots \oplus H^{b_0}(D)[z^{b_0}(D) \oplus 1(D)] \oplus \cdots \oplus H^0(D)z^0(D) = 0(D),$$

$$H^{\bar{k}}(D)z^{\bar{k}}(D) \oplus \cdots \oplus H^{b_0}(D)z^{b_0}(D) \oplus \cdots \oplus H^0(D)z^0(D) = E[H^{b_0}(D)](D), \qquad (2.31)$$

where $E[H^{b_0}(D)]$ is the modulo-2 number of non-zero terms in $H^{b_0}(D)$ and $1(D) = \sum_{j=-\infty}^{\infty} D^j$ is the all one sequence (i.e., $E[H^{b_0}(D)](D) = H^{b_0}(D)1(D)$). Thus if there is an even number of terms in $H^{b_0}(D)$, (2.31) is the same as (2.27). That is, the code is transparent to integer multiples of Ψ phase rotations of the signal set. However, if there is an odd number of terms in $H^{b_0}(D)$, then $E[H^{b_0}(D)] = 1$ and the coset of the convolutional code is produced. Even though the two equations are closely related, the codes are quite different and a decoder is not able to produce correctly decoded data from a Ψ phase rotation of the signal set.

Now assume that the first two terms are affected by a phase rotation, i.e., the largest value of $b_j \le \tilde{k}$ is b_1. The terms in the parity check polynomial $H^{b_0}(D)z^{b_0}(D) \oplus H^{b_1}(D)z^{b_1}(D)$ now become

$$[H^{b_0}(D) \oplus H^{b_1}(D)]z^{b_0}(D) \oplus H^{b_1}(D)z^{b_1}(D) \oplus E[H^{b_0}(D)](D).$$

In this case the parity check equation is different after a phase rotation (even if $E[H^{b_0}(D)] = 0$). This means that the code is not transparent to a Ψ phase rotation, but it could be transparent to 2Ψ or 4Ψ phase rotations. This is because the phase rotation equations reduce to

$$z_r^{b_0} = z^{b_0}, \ldots, z_r^{b_{d-1}} = z^{b_{d-1}}, z_r^{b_d} = z^{b_d} \oplus 1, z_r^{b_{d+1}} = z^{b_{d+1}} \oplus z^{b_d}, \ldots$$

for a $2^d\Psi$ phase rotation, where d = 1 or 2. If there is an even number of terms in $H^{b_1}(D)$, then d = 1. This is because an even number of terms in $H^{b_1}(D)$ cancels the effect on $z^{b_1}(D)$ when the signal set is rotated by 2Ψ. That is, the code is transparent to integer multiples of 2Ψ phase rotations, but not to multiples of Ψ. If there is an odd number of terms in $H^{b_1}(D)$, this cancellation effect does not occur, implying that d = 2 and the phase transparency is 4Ψ.

In general, if the largest value of $b_j \le \tilde{k}$ is b_f, then $d = f + E[H^{b_f}(D)]$. We can then determine those bits z^{c_j} which are affected by a $2^d\Psi$ phase rotation, i.e., $c_j = b_{j+d} = p_{j+d} - q$, for $0 \le j \le s-1$, where s = I-d.

Example 2.3

For the code given in Example 2.2, $\tilde{k} = 2$, I = 3, and q = 1. Thus $b_0 = 1$, $b_1 = 2$, and $b_2 = 5$. Since the largest value of $b_j \le 2$ is b_1, then f = 1. Therefore $d = 1 + E[H^{b_1}(D)] = 1 + E[D^3 \oplus D^2] = 1$. Thus the code is 90° transparent, and $c_0 = 2$ and $c_1 = 5$.

2.2.7 Systematic Search for Good Small Constraint Length Codes

An approximate lower bound for the symbol error probability [65] of a multi-D code is given by

$$P_s(e) \gtrsim \frac{N_{free}}{L} Q\left(\sqrt{\frac{d^2_{free} K}{2} \frac{E_b}{N_0}}\right), \tag{2.32}$$

where E_b/N_0 is the energy per information bit to single sided noise density ratio and $Q(\cdot)$ is the complementary error function. In (2.32),

the division by L normalizes the average number of errors per multi-D signal to that of a 2-D signal set.

For each multi-D signal set considered, there are a number of code rates which can be achieved. As ν is increased, a comprehensive code search becomes time consuming due to the greater complexity of each code. We have thus limited our search to $\nu + \tilde{k} \leq 10$. (The number of checked bits $\tilde{k}$ also affects the complexity of the code search.) As indicated by (2.32), the criteria used to find the best codes are the free MSED (d^2_{free}) and the number of nearest neighbors (N_{free}). We have also included the code transparency (d) as a criteria in the code serach. The code search algorithm that was implemented is similar to that in [65], but with a number of differences which include the extra criteria mentioned above.

The actual code search involves using a rate $\tilde{k}/(\tilde{k}+1)$ code. Thus two separate notations are used to distinguish the rate k/(k+1) encoder and the simplified rate $\tilde{k}/(\tilde{k}+1)$ encoder. For the rate k/(k+1) encoder, we have $\mathbf{x}_n = [x^k_n,\ldots,x^1_n]$ (the input to the encoder) and $\mathbf{z}_n = [z^k_n,\ldots,z^1_n,z^0_n]$ (the mapped bits or encoder output) at time n. Also, $\mathbf{e}_n = [e^k_n,\ldots,e^1_n,e^0_n]$ is the modulo-2 difference between two encoder outputs $\mathbf{z}_n$ and $\mathbf{z}'_n$ at time n, i.e., $\mathbf{e}_n = \mathbf{z}_n \oplus \mathbf{z}'_n$. Note that there are 2^{k+1} combinations of $\mathbf{z}_n$ and $\mathbf{z}'_n$ that give the same $\mathbf{e}_n$. For the rate $\tilde{k}/(\tilde{k}+1)$ code, we denote reduced versions of $\mathbf{x}_n$, $\mathbf{z}_n$, and $\mathbf{e}_n$ as $\tilde{\mathbf{x}}_n = [x^{\tilde{k}}_n,\ldots,x^1_n]$, $\tilde{\mathbf{z}}_n = [z^{\tilde{k}}_n,\ldots,z^1_n,z^0_n]$, and $\tilde{\mathbf{e}}_n = [e^{\tilde{k}}_n,\ldots,e^1_n,e^0_n]$, respectively.

In order to find d^2_{free} for a particular code, the squared Euclidean weights (SEW) $w^2(\mathbf{e}_n)$ are used. As defined in [65], $w^2(\mathbf{e}_n)$ is the MSED between all combinations of $a(\mathbf{z}_n)$ and $a(\mathbf{z}'_n)$ such that

$\mathbf{e}_n = \mathbf{z}_n \oplus \mathbf{z}'_n$ and $a(\mathbf{z}_n)$ is the actual L×MPSK signal point. This can be defined as

$$w^2(\mathbf{e}_n) = \min_{\text{all } \mathbf{z}_n} d^2[a(\mathbf{z}_n), a(\mathbf{z}_n \oplus \mathbf{e}_n)], \tag{2.33}$$

where $d^2[a(\mathbf{z}_n), a(\mathbf{z}'_n)]$ is the SED between $a(\mathbf{z}_n)$ and $a(\mathbf{z}'_n)$. One can then use the all zero path as a reference to find d^2_{free} in a code search, i.e.,

$$d^2_{free} = \min \sum_n w^2(\mathbf{e}_n), \tag{2.34}$$

where the minimization is over all allowable code sequences with the exception of the all-zero sequence. We can use (2.34) to find d^2_{free} provided that the minimization of (2.33) does not depend on z^0_n, as shown by Ungerboeck [65].

Although the minimization of (2.33) does not depend on z^0_n for 1×MPSK signal sets, it cannot be assumed that this also applies to L×MPSK for $L \geq 2$. By expressing $d^2[a(\mathbf{z}_n), a(\mathbf{z}_n \oplus \mathbf{e}_n)]$ directly in terms of $\mathbf{z}_n$ and $\mathbf{e}_n$, it can be shown that 3×4PSK (I), 3×8PSK (I and II), and 3×16PSK (I, II, and III) all depend on z^0_n. This implies that (2.34) becomes a lower bound in these cases. However, due to the large number of parallel transitions for these codes, we can still determine d^2_{free} (and N_{free}) using a slightly modified version of (2.34).

Since there are 2^{k+1} values of $\mathbf{e}_n$, there are a total of 2^{2k+2} computations required to find all the values of $w^2(\mathbf{e}_n)$. For example, a rate 11/12 code with 4×8PSK modulation requires nearly 17 million computations. This can be reduced by letting $z^0_n = 0$ (or 1) and minimizing (2.33) over all $\mathbf{z}_n = [z^k_n,\ldots,z^1_n,0]$. This reduces the number

of computations to 2^{2k+1}. In fact, it is possible to even further decrease the number of computations. Using some difficult algebraic manipulations, it can be shown that the L output bits z_n^p corresponding to cosets t^p with some components equal to 2^{I-1} can all be set to zero. For example, the 4×8PSK signal set with q = 0 can have bits z_n^7, z_n^9, z_n^{10}, and z_n^{11} all set to 0 when minimizing (2.33). This is due in part to the MPSK signals being antipodal for these values. Thus the total number of computations can be reduced to 2^{2k-L+1}.

In order to reduce the time needed to find d^2_{free}, we note that the trellis is equivalent to a rate $\tilde{k}/(\tilde{k}+1)$ code with $2^{k-\tilde{k}}$ parallel transitions. Also, there are $2^{\tilde{k}+1}$ different sets of parallel transitions. If the minimum SEW is found for each of these sets of parallel transitions, the code search is greatly simplified, since the search for a rate $\tilde{k}/(\tilde{k}+1)$ code is all that is needed and $\tilde{k}$ is usually small. Thus, the SEW's required for a rate $\tilde{k}/(\tilde{k}+1)$ code search are

$$w^2(\tilde{e}_n) = \min w^2(e_n), \tag{2.35}$$

where the minimization is over all $[e_n^k,\ldots,e_n^{\tilde{k}+1}]$. We define the free MSED of this rate $\tilde{k}/(\tilde{k}+1)$ code as

$$\tilde{d}^2_{free} = \min \sum_n w^2(\tilde{e}_n), \tag{2.36}$$

where the minimization is over all allowable code sequences $(\tilde{e}(D))$ defined by

$$\tilde{e}(D) = \tilde{e}_1 D \oplus \tilde{e}_2 D^2 \oplus \cdots \oplus \tilde{e}_N D^N,$$

for $\tilde{e}_1$, $\tilde{e}_N \neq \mathbf{0}$, and $N \geq 2$. The code sequences of length N = 1 are the

parallel transitions, where the MSED is the MSSD of the parallel transitions. A code might have $\tilde{d}^2_{free}$ larger than the MSSD of the parallel transitions, implying that d^2_{free} occurs along the parallel transitions. With $\tilde{k}$ checked bits and a rate $\tilde{k}/(\tilde{k}+1)$ code, the MSSD of the parallel transitions is $\Delta^2_{q+\tilde{k}+1}$. Thus we can express d^2_{free} as

$$d^2_{free} = \min\,(\tilde{d}^2_{free}, \Delta^2_{q+\tilde{k}+1}). \tag{2.37}$$

The best value of $\tilde{k}$ can be determined from the free MSED of the best code for the previous value of ν. The search starts with $\nu = 1$ and $\tilde{k} = 1$, and we find the code with the best d^2_{free} and N_{free}. We then increase ν by one and determine $\tilde{k}$ as follows. If d^2_{free} for the previous best code was $\tilde{d}^2_{free}$, then $\tilde{k}$ remains the same. This is because the limit of the parallel transitions $(\Delta^2_{q+\tilde{k}+1})$ has not yet been reached and the trellis connectivity needs to be reduced in order to increase d^2_{free} or reduce N_{free}. If d^2_{free} for the previous best code was $\Delta^2_{q+\tilde{k}+1}$, then $\tilde{k}$ is increased by one from the previous value; otherwise, d^2_{free} and N_{free} would remain the same. If $\tilde{d}^2_{free} = \Delta^2_{q+\tilde{k}+1}$ for the previous best code, then $\tilde{k}$ can remain the same or increase by one. Both values of $\tilde{k}$ should be tried in order to find the best code. The best code is then found for this value of ν and $\tilde{k}$, and the above process is repeated for each increasing value of ν.

As can be seen from (2.33), there may be some values of $\mathbf{e}_n$ and $\mathbf{z}_n$ for which $w^2(\mathbf{e}_n) < d^2[a(\mathbf{z}_n), a(\mathbf{z}_n \oplus \mathbf{e}_n)]$. The "number of nearest neighbors" for $\mathbf{e}_n$ (denoted $m(\mathbf{e}_n)$) is defined as the average number of times that $w^2(\mathbf{e}_n)$ equals $d^2[a(\mathbf{z}_n), a(\mathbf{z}_n \oplus \mathbf{e}_n)]$. If $w^2(\mathbf{e}_n)$ equals $d^2[a(\mathbf{z}_n), a(\mathbf{z}_n \oplus \mathbf{e}_n)]$ for all values of $\mathbf{z}_n$, then $m(\mathbf{e}_n) = 1$. For example, in naturally mapped 8PSK it is found that for $\mathbf{e}_n = [0\ 1\ 1]$ and

[1 1 1], $d^2[a(\mathbf{z}_n), a(\mathbf{z}_n \oplus \mathbf{e}_n)] = 0.586$ for four values of $\mathbf{z}_n$ and 3.414 for the other four values of $\mathbf{z}_n$. Thus $m(\mathbf{e}_n) = 0.5$ for $\mathbf{e}_n = [0\ 1\ 1]$ and [1 1 1]. For all other values of $\mathbf{e}_n$, it can be shown that $m(\mathbf{e}_n) = 1$. Zehavi and Wolf [79] give a general approach to determining the full code distance spectrum, whereas we are only interested in the number of nearest neighbors.

We can state this generally as follows. Let the number of bits in $\mathbf{z}_n$ that are varied to find $w^2(\mathbf{e}_n)$ be b. Then

$$m(\mathbf{e}_n) = \sum u\left(w^2(\mathbf{e}_n) - d^2[a(\mathbf{z}_n), a(\mathbf{z}_n \oplus \mathbf{e}_n)]\right)2^{-b}, \tag{2.38}$$

where $u(\cdot)$ is the unit step function and the summation is over all the bits in $\mathbf{z}_n$ that are varied to find $w^2(\mathbf{e}_n)$. Normally $b = k + 1$, but this can be reduced to $b = k - L$ for the reasons mentioned previously.

For the simplified rate $\tilde{k}/(\tilde{k}+1)$ code, $m(\tilde{\mathbf{e}}_n)$ is the sum of all the $m(\mathbf{e}_n)$'s for which $w^2(\tilde{\mathbf{e}}_n) = w^2(\mathbf{e}_n)$, i.e.,

$$m(\tilde{\mathbf{e}}_n) = \sum u\left(w^2(\tilde{\mathbf{e}}_n) - w^2(\mathbf{e}_n)\right)m(\mathbf{e}_n), \tag{2.39}$$

where the summation is over all $[e_n^k,\ldots,e_n^{\tilde{k}+1}]$. We can think of $m(\tilde{\mathbf{e}}_n)$ as the total average number of nearest neighbors along each set of parallel transitions.

The number of nearest neighbors for the MSSD $\Delta^2_{q+\tilde{k}+1}$ is

$$N_\Delta = \sum u\left(\Delta^2_{q+\tilde{k}+1} - w^2(\mathbf{e}_n)\right)m(\mathbf{e}_n), \tag{2.40}$$

where the summation is over all $\mathbf{e}_n = [e_n^k,\ldots,e_n^{\tilde{k}+1},0,\ldots,0]$. The number of nearest neighbors for paths with SED $\tilde{d}^2_{free}$ can be calculated using $m(\tilde{\mathbf{e}}_n)$ as follows:

$$\tilde{N}_{free} = \sum_{\alpha=1}^{A} \prod_{n=1}^{N_\alpha} m(\tilde{e}_n), \tag{2.41}$$

where N_α is the length of a path α that has a SED of $\tilde{d}^2_{free}$ and A is the number of paths that have a SED of $\tilde{d}^2_{free}$. If d^2_{free} occurs along the parallel transitions, $N_{free} = N_\Delta$, and we define the next nearest free SED and number of nearest neighbors as $d^2_{next} = \tilde{d}^2_{free}$ and $N_{next} = \tilde{N}_{free}$, respectively. (Note that d^2_{next} and N_{next} may not be the true next nearest paths, since there may be some closer paths occuring along the parallel transitions.) When there are several codes that have the same free MSED and number of nearest neighbors, the "next nearest" values are used in code selection. When d^2_{free} occurs along paths with SED $\tilde{d}^2_{free}$, $N_{free} = \tilde{N}_{free}$. The next nearest values in this case are not given in the code tables. If $\tilde{d}^2_{free} = \Delta^2_{q+\tilde{k}+1}$, then $N_{free} = N_\Delta + \tilde{N}_{free}$.

Example 2.4

In Example 2.2 we have a $\tilde{k} = 2$, $q = 1$, rate 7/8 (2.33 bit/sym) code with a 3×8PSK (II) signal set. After determining the mapping of the signal set, (2.33) was used to find the SEW's for each signal point. Equation (2.35) determines the $w^2(\tilde{e}_n)$'s that were used to find the best rate 2/3 codes. For these codes $d^2_{free} = \Delta^2_{q+\tilde{k}+1} = \Delta^2_4 = 4.0$. Using (2.40) we determined that N_{free} is 15 (after normalizing, there are only 5 paths per 2-D symbol). In the code search for the best rate 2/3 codes, there were many codes which had $d^2_{next} = \tilde{d}^2_{free} = 4.343$. Thus (2.41) was used to determine N_{next} for each best code. Table 2.10 gives the values of $w^2(\tilde{e}_n)$ and $m(\tilde{e}_n)$ for each $\tilde{e}_n$ that were used in the code search. The best code with a transparency of 90° was found to have $N_{next} = 24$.

TABLE 2.10

SQUARED EUCLIDEAN WEIGHTS USED IN THE CODE SEARCH FOR RATE 7/8 (2.33 bit/sym) CODES WITH 3×8PSK (II) AND $\tilde{k} = 2$

$\tilde{e}_n$	$w^2(\tilde{e}_n)$	$m(\tilde{e}_n)$
000	0.0	1
001	1.172	2
010	1.757	4
011	0.586	1
100	2.0	6
101	1.172	2
110	1.757	4
111	0.586	1

In order to reduce the number of codes that must be tested in our code search algorithm, rejection rules were used. As in Rule 1 of [65], time reversal of the parity check polynomials was used to reject codes. Even though $w^2(\tilde{e}_n)$ and $m(\tilde{e}_n)$ are used to find the best codes, Rule 2 in [65] can still be exploited, provided that $w^2(\tilde{e}_n) = \Delta^2_{r(\tilde{e}_n)+q}$, where $r(\tilde{e}_n)$ is the number of trailing zero's in $\tilde{e}_n$. When this is not true, it may still be possible to find some combinations of the parity check polynomials that can be rejected (this was also implemented in our code search). Rule 3 in [65] was also used to eliminate codes.

In the code search, a rate $\tilde{k}/(\tilde{k}+1)$ code is searched for a particular v. Before finding $\tilde{d}^2_{free}$, the code search program checks to make sure that the code only produces sequences with length $N \geq 2$. If for some input $\tilde{x}_n \neq \mathbf{0}$, the inputs to the systematic encoder are all zero, the state of the encoder goes from one state to the next as if a zero input had occurred. Thus parallel transitions will occur in the rate $\tilde{k}/(\tilde{k}+1)$ code, which should not have parallel transitions. Therefore, in the code search, codes at level i $(1 \leq i \leq \tilde{k})$ were

rejected if for some $[x^i,\ldots,x^l] \neq \mathbf{0}$, $\sum_{j=1}^{i} x^j H^j(D) \pmod 2 = 0(D)$.

Two programs written in pascal were used in the code search, one for codes with $\nu > \tilde{k}$ and the other for codes with $\nu = \tilde{k}$. For specific values of I, L, and q, $y^q(z)$, for $0 \le z \le 2^{IL-q}-1$, was generated using the coset representatives t^p, for $0 \le p \le IL-1$, that are given in Tables 2.5, 2.6, and 2.7. The squared Euclidean weights $w^2(\mathbf{e}_n)$ were then calculated using (2.33) for all $\mathbf{e}_n$. Since the value of $\tilde{k}$ can change with each ν, $w^2(\tilde{\mathbf{e}}_n)$ and $m(\tilde{\mathbf{e}}_n)$ were computed, if necessary, as the program went from the smallest to the largest ν.

The code search used the various rejection rules before the time consuming tasks of finding $\tilde{d}^2_{free}$ (using the bi-directional search algorithm [38]) and N_{free} (using a technique based on the Viterbi algorithm). The rejection rules were organized so that the best codes for each of the two possible phase transparancies were found. The code search found those codes which had the largest free distance (for a particular transparency). If a code was found to have its free MSED equal to or greater than the previous best code, $\tilde{N}_{free}$ was determined and this code was listed if either its $\tilde{d}^2_{free}$ or $\tilde{N}_{free}$ had improved over the previous best code.

The octal code generators were then listed along with their $\tilde{d}^2_{free}$, $\tilde{N}_{free}$, and phase transparency d. A small list of codes was produced (for each code search) from which the best codes could be chosen. Every time that $\tilde{k}$ is increased by one in the code search (which is done automatically), the program determines and lists $\Delta^2_{q+\tilde{k}+1}$ and N_Δ for use in the code tables.

The asymptotic coding gain γ of each code compared to the uncoded case, as shown in the code tables, is

$$\gamma = 10 \log_{10}(d^2_{free}/d^2_u) \text{ (dB)}, \tag{2.42}$$

where d^2_u is the smallest MSSD of an equivalent uncoded 2-D or multi-D scheme. In nearly all cases, $d^2_u = \Delta^2_{q+1}$. For codes with a non-integer K, no equivalent 1×MPSK scheme exists which has the same K, and so the equivalent uncoded multi-D signal set is used instead. For the 4×8PSK signal set with q = 3, K = 2 bit/sym. Thus, a natural comparison would be against uncoded 4PSK, which has $d^2_u = 2$. (In this case, $\Delta^2_{q+1} = 2.343$, which is inconsistent with other codes that also have K = 2 bit/sym.) The asymptotic coding gains compared to uncoded (M/2)-PSK are found by adding to γ the appropriate correction factor

$$\gamma_{M/2} = 10 \log_{10} \left(\frac{K\, d^2_u}{(I-1)\delta^2_1} \right) \text{ (dB)}, \tag{2.43}$$

as shown in the code tables. The transparency (in degrees) is also given for each code. The parity check polynomials are expressed in octal notation in the code tables, e.g., $H^0(D) = D^6 + D^4 + D^2 + D + 1 \equiv (001\ 010\ 111)_2 \equiv (127)_8$.

In Tables 2.11, 2.15, and 2.19, codes for TC-1×4PSK (rate 1/2 4PSK), TC-1×8PSK (rate 2/3 8PSK), and TC-1×16PSK (rate 3/4 16PSK), respectively, are presented. These tables give the best code for each phase transparency, which (to the best of our knowledge) have not been previously published. The best codes, without regard for phase transparency, were originally published by Odenwalder [43] for 4PSK (with the codes in non-systematic form), by Ungerboeck [65,68] for 8PSK, and by Wilson, et. al. [76] for 16PSK.

TABLE 2.11

TRELLIS CODED 1×4PSK.

$K = 1.0$ bit/sym, $d_u^2 = 4.0$, $N_u = 1$ (1×2PSK).

ν	$\tilde{k}$	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	1	3	360°	6	1	-	-	1.76
2	1	2	5	360°	10	1	-	-	3.98
3	1	06	13	180°	12	2	-	-	4.77
	1	04	13	360°	12	1	-	-	4.77
4	1	06	21	180°	12	1	-	-	4.77
	1	10	23	360°	14	2	-	-	5.44
5	1	36	45	180°	16	2	-	-	6.02
	1	26	53	360°	16	1	-	-	6.02
6	1	042	117	180°	20	11	-	-	6.99
7	1	126	235	180°	20	2	-	-	6.99
	1	144	223	360°	20	1	-	-	6.99
8	1	262	435	180°	24	11	-	-	7.78
	1	362	515	360°	24	9	-	-	7.78
9	1	0644	1123	180°	24	2	-	-	7.78
	1	0712	1047	360°	24	1	-	-	7.78

$\gamma_2 = 0$ dB

Tables 2.12, 2.16, and 2.20 list the TC-2×4PSK codes (rates of 1.5 and 1.0 bit/sym), the TC-2×8PSK codes (2.5 and 2.0 bit/sym), and the TC-2×16PSK codes (3.5 and 3.0 bit/sym), respectively. Tables 2.13, 2.17, and 2.21 list the TC-3×4PSK codes (1.67, 1.33, and 1.0 bit/sym), the TC-3×8PSK codes (2.67, 2.33, and 2.0 bit/sym), and the TC-3×16PSK codes (3.67, 3.33, and 3.0 bit/sym), respectively. Tables 2.14, 2.18, and 2.22 list the TC-4×4PSK codes (1.75, 1.5, 1.25, and 1.0 bit/sym), the TC-4×8PSK codes (2.75, 2.5, 2.25, and 2.0 bit/sym), and the TC-4×16PSK codes (3.75, 3.5, 3.25, and 3.0 bit/sym), respectively.

TABLE 2.12(a)

TRELLIS CODED 2×4PSK

K = 1.5 bit/sym, q=0, d_u^2 = 4, N_u = 6 (2×4PSK).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	1	3	180°	4	2	6	8	0.00
2	2	-	1	3	5	90°	6	6	-	-	1.76
3	2	-	04	06	11	90°	8	5	-	-	3.01
4	2	-	10	06	23	90°	8	1	10	16	3.01
5	3	14	30	02	41	180°	10	8	-	-	3.98
	3	16	24	06	53	360°	10	7	-	-	3.98
6	3	030	042	014	103	180°	12	40.25	-	-	4.77
	3	076	024	010	157	360°	12	30.75	-	-	4.77
7	3	044	022	114	211	180°	12	8	-	-	4.77

γ_2 = 1.76 dB

TABLE 2.12(b)

TRELLIS CODED 2×4PSK

K = 1.0 bit/sym, q=1, d_u^2 = 4.0, N_u = 1 (1×2PSK).

ν	$\tilde{k}$	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	1	3	90°	8	5	-	-	3.01
2	1	-	2	5	90°	8	1	12	8	3.01
3	2	04	02	11	360°	12	5	-	-	4.77
4	2	14	06	23	180°	12	1	-	-	4.77
5	2	30	16	41	180°	16	8	-	-	6.02
6	2	036	052	115	180°	16	1	-	-	6.02
7	2	044	136	203	180°	20	6	-	-	6.99
8	2	110	226	433	180°	24	33	-	-	7.78

γ_2 = 0 dB

TABLE 2.13(a)

TRELLIS CODED 3×4PSK

K = 1.67 bit/sym, q=0, d_u^2 = 4.0, N_u = 15 (3×4PSK I).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)	sig. set
1	1	-	-	1	3	90°	4	7	6	32	0.00	I
2	2	-	2	1	5	90°	4	3	6	24	0.00	I
	2	-	2	1	5	360°	4	2	-	-	0.00	II
3	2	-	04	02	11	90°	4	1	6	6	0.00	III
	2	-	04	02	11	360°	6	11	-	-	1.76	II
	3	05	04	02	11	90°	4	0.25	-	-	0.00	III
4	2	-	14	02	21	180°	6	6	-	-	1.76	II
3	3	01	02	06	11	360°	6	4	-	-	1.76	II
4	3	10	04	02	21	90°	6	5.5	-	-	1.76	III
	3	12	04	02	21	180°	8	19	-	-	3.01	I
5	3	24	14	02	41	180°	8	7	-	-	3.01	I
6	3	024	042	010	105	180°	8	3	10	16	3.01	I

γ_2 = 2.22 dB

Equivalent R = 5/6, TC-2×8PSK (2.5 bit/sym) codes with up to 16 states have been found independently by Lafanechère and Costello [37] and by Wilson [76], although with reduced phase transparency. The 2 state TC-L×8PSK and TC-L×16PSK codes were also found by Divsalar and Simon [15].

In the code tables it can be seen that for the same complexity, there are usually two codes (and in some cases three codes) that are given. Note that the code with the worst phase transparency has a better free distance or a fewer number of nearest or next nearest neighbors. Thus, if phase transparency is not required, one should choose the less phase transparent code in order to obtain the maximum performance for a given complexity.

TABLE 2.13(b)

TRELLIS CODED 3×4PSK

$K = 1.33$ bit/sym, $q=1$, $d_u^2 = 4.0$, $N_u = 3$ (3×4PSK II).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)	sig. set
1	1	-	-	1	3	90°	4	1	8	4	0.00	III
	1	-	-	1	3	360°	6	7	-	-	1.76	II
2	1	-	-	2	5	360°	6	4	10	9	1.76	II
	2	-	2	1	5	90°	6	2	8	4	1.76	III
	2	-	3	1	5	180°	8	21	-	-	3.01	I
	2	-	2	1	5	360°	8	16	-	-	3.01	II
3	2	-	04	02	11	90°	6	2	8	1	1.76	III
	2	-	02	06	11	180°	8	3	12	100	3.01	II
	3	06	04	03	11	90°	8	1	-	-	3.01	III
4	3	14	04	12	23	90°	10	5	-	-	3.98	III
5	3	30	04	22	43	90°	12	13	-	-	4.77	III
6	3	036	060	026	103	90°	12	2	-	-	4.77	III
7	3	140	160	062	213	90°	12	1	14	5	4.77	III
	3	004	154	056	207	180°	12	1	16	128	4.77	III

$\gamma_2 = 1.25$ dB

2.2.8 Decoder Implementation

When the Viterbi algorithm is used as the decoder, a measure of decoding complexity is given by $2^{\nu+\tilde{k}}/L$. This is the number of distinct transitions in the trellis diagram for any TCM scheme normalized to a 2-D signal set. The maximum bit rate of the decoder is kf_d, where f_d is the symbol speed of the decoder. Since k is quite large for multi-D signal sets (at least (I-1)L), high bit rates can be achieved. For example, a Viterbi decoder has been constructed for a rate 7/9 periodically time varying trellis code (PTVTC) with $\nu = 4$, $\tilde{k} = 2$, and 8PSK modulation [30]. This decoder has $f_d = 60$ MHz and a bit rate of

TABLE 2.13(c)

TRELLIS CODED 3×4PSK

K = 1.00 bit/sym, q=2, d_u^2 = 4.0, N_u = 1 (1×2PSK).

v	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)	sig. set
0	0	-	-	-	-	90°	6	4	-	-	1.76	II
1	1	-	-	1	3	90°	6	2	8	1	1.76	III
	1	-	-	1	3	180°	8	3	12	16	3.01	II
2	2	-	3	2	5	90°	10	4	-	-	3.98	III
3	2	-	06	02	11	90°	10	2	-	-	3.98	III
	2	-	02	06	13	180°	12	5	-	-	4.77	III
4	2	-	12	16	21	90°	12	1	14	2	4.77	III
	2	-	04	12	27	180°	12	1	16	22	4.77	III
	3	10	04	02	21	180°	14	3	-	-	5.44	II
5	3	22	16	04	53	180°	16	2	-	-	6.02	II
	3	24	14	02	43	360°	16	1	-	-	6.02	II
6	3	070	004	022	101	180°	18	3	-	-	6.53	II
7	3	156	024	046	213	180°	20	3	-	-	6.99	II
	3	044	014	102	217	360°	20	2	-	-	6.99	II

γ_2 = 0.0 dB

TABLE 2.14(a)

TRELLIS CODED 4×4PSK

K = 1.75 bit/sym, q=0, d_u^2 = 4.0, N_u = 28 (4×4PSK).

v	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	1	3	90°	4	12	6	64	0.00
2	2	-	2	1	5	90°	4	4	6	48	0.00
3	3	04	02	01	11	90°	6	28	-	-	1.76
4	3	10	04	02	21	90°	8	78	-	-	3.01
5	3	24	14	02	41	90°	8	30	-	-	3.01
6	3	050	032	004	103	90°	8	14	10	160	3.01

γ_2 = 2.43 dB

TABLE 2.14(b)

TRELLIS CODED 4×4PSK

K = 1.50 bit/sym, q=1, d_u^2 = 4.0, N_u = 6 (2×4PSK).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	-	1	3	90°	4	4	8	64	0.00
2	2	-	-	2	1	5	90°	8	78	-	-	3.01
3	2	-	-	04	02	11	90°	8	30	-	-	3.01
4	2	-	-	12	04	23	90°	8	16	12	320	3.01
5	3	-	14	34	06	41	90°	8	6	12	176	3.01
	3	-	04	14	22	43	180°	8	6	12	160	3.01
6	4	014	006	056	022	103	90°	8	2	12	62	3.01

γ_2 = 1.76 dB

TABLE 2.14(c)

TRELLIS CODED 4×4PSK

K = 1.25 bit/sym, q=2, d_u^2 = 4.0, N_u = 4 (4×4PSK).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	-	1	3	90°	8	30	-	-	3.01
2	1	-	-	-	2	5	90°	8	14	12	64	3.01
3	2	-	-	06	02	11	90°	8	6	12	64	3.01
	2	-	-	02	06	11	180°	8	6	12	32	3.01
	3	-	01	03	06	11	90°	8	2	12	56	3.01
4	3	-	10	14	06	21	90°	8	2	12	8	3.01
5	4	10	04	06	22	41	90°	12	8	-	-	4.77
6	4	024	014	006	042	103	90°	16	109	-	-	6.02

γ_2 = 0.97 dB

TABLE 2.14(d)

TRELLIS CODED 4×4PSK

K = 1.00 bit/sym, q=3, d_u^2 = 4.0, N_u = 1 (1×2PSK).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
0	0	-	-	-	-	90°	8	14	-	-	3.01
1	1	-	-	1	3	180°	8	6	16	64	3.01
2	2	-	2	3	5	90°	8	2	16	64	3.01
3	3	02	04	03	11	90°	16	45	-	-	6.02
4	3	02	10	06	21	90°	16	17	-	-	6.02
5	3	22	10	06	41	90°	16	5	-	-	6.02
6	3	010	060	036	105	90°	16	1	20	4	6.02

γ_2 = 0 dB

TABLE 2.15

TRELLIS CODED 1×8PSK

K = 2.0 bit/sym, d_u^2 = 2.0, N_u = 2 (1×4PSK).

ν	$\tilde{k}$	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	1	3	180°	2.586	2	-	-	1.12
2	1	-	2	5	180°	4.0	1	4.586	4	3.01
3	2	04	02	11	360°	4.586	2	-	-	3.60
4	2	14	06	23	180°	5.172	4	-	-	4.13
	2	16	04	23	360°	5.172	2.25	-	-	4.13
5	2	14	26	53	180°	5.172	0.25	-	-	4.13
	2	20	10	45	360°	5.757	2	-	-	4.59
6	2	074	012	147	180°	6.343	3.25	-	-	5.01
7	2	146	052	225	180°	6.343	0.125	-	-	5.01
	2	122	054	277	360°	6.586	0.5	-	-	5.18
8	2	146	210	573	180°	7.515	3.375	-	-	5.75
	2	130	072	435	360°	7.515	1.5	-	-	5.75

γ_4 = 0 dB

TABLE 2.16(a)

TRELLIS CODED 2×8PSK

K = 2.5 bit/sym, q=0, d_u^2 = 1.172, N_u = 4 (2×8PSK).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	1	3	90°	1.757	8	2.0	4	1.76
2	1	-	-	2	5	90°	2.0	4	2.929	32	2.32
3	2	-	04	06	11	45°	2.929	16	-	-	3.98
4	2	-	16	12	23	45°	3.515	56	-	-	4.77
5	2	-	10	06	41	45°	3.515	16	-	-	4.77
6	2	-	004	030	113	45°	4.0	6	4.101	80	5.33
	2	-	044	016	107	90°	4.0	6	4.101	48	5.33
7	3	110	044	016	317	90°	4.0	2	4.101	25	5.33

γ_4 = -1.35 dB

TABLE 2.16(b)

TRELLIS CODED 2×8PSK

K = 2.0 bit/sym, q=1, d_u^2 = 2.0, N_u = 2 (1×4PSK).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	1	3	45°	3.172	8	4.0	6	2.00
2	1	-	-	2	5	45°	4.0	6	5.172	32	3.01
3	2	-	04	02	11	180°	4.0	2	5.172	16	3.01
4	3	04	14	02	21	90°	5.172	8	-	-	4.13
5	3	24	14	06	43	90°	6.0	6	-	-	4.77
6	3	012	050	004	125	90°	6.343	5.5	-	-	5.01
7	3	110	044	016	317	90°	7.515	25	-	-	5.75

γ_4 = 0 dB

TABLE 2.17(a)

TRELLIS CODED 3×8PSK

K = 2.67 bit/sym, q=0, d_u^2 = 1.172, N_u = 12 (3×8PSK I).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)	sig. set
1	1	-	-	1	3	45°	1.172	4	-	-	0.00	II
2	1	-	-	2	5	45°	1.757	16	-	-	1.76	II
3	2	-	04	02	11	45°	2.0	6	2.343	16	2.32	I
4	3	14	04	02	21	90°	2.343	12	-	-	3.01	I
	3	10	04	02	21	180°	2.343	8	-	-	3.01	I
5	3	30	14	02	53	90°	2.929	48	-	-	3.98	I
6	3	050	022	006	103	90°	3.172	12	-	-	4.33	I
7	3	056	112	004	225	90°	3.515	84	-	-	4.77	I
	3	100	050	022	255	180°	3.515	76	-	-	4.77	I

γ_4 = -1.07 dB

TABLE 2.17(b)

TRELLIS CODED 3×8PSK

K = 2.33 bit/sym, q=1, d_u^2 = 1.757, N_u = 8 (3×8PSK II).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)	sig. set
1	1	-	-	-	1	3	90°	2.0	6	2.343	16	0.56	II
2	2	-	-	3	1	7	90°	2.586	6	-	-	1.68	II
3	2	-	-	06	02	11	90°	3.515	16	-	-	3.01	II
	2	-	-	04	02	11	180°	3.757	24	-	-	3.30	II
4	3	-	10	04	06	21	45°	3.757	12	-	-	3.30	III
	2	-	-	14	02	27	90°	4.0	15	4.343	24	3.57	II
5	3	-	22	16	06	41	45°	4.0	7	-	-	3.57	III
6	3	-	010	046	060	105	45°	4.0	3	4.686	8	3.57	III
	4	060	024	014	002	101	180°	4.0	2	-	-	3.57	III

γ_4 = 0.11 dB

TABLE 2.17(c)

TRELLIS CODED 3×8PSK

K = 2.00 bit/sym, q=2, d_u^2 = 2.0, N_u = 2 (1×4PSK).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)	sig. set
1	1	-	-	-	1	3	180°	3.757	24	-	-	2.74	II
2	1	-	-	-	2	5	180°	4.0	15	5.757	144	3.01	II
3	2	-	-	04	02	11	45°	4.0	7	-	-	3.01	III
4	2	-	-	12	04	27	45°	4.0	3	5.757	32	3.01	III
5	3	-	14	24	02	41	180°	5.757	17.5	-	-	4.59	III
	3	-	16	22	06	53	360°	5.757	17	-	-	4.59	III
6	3	-	030	042	014	103	180°	6.0	11	-	-	4.77	III
	4	014	044	024	006	103	180°	6.0	4	-	-	4.77	II

γ_4 = 0 dB

TABLE 2.18(a)

TRELLIS CODED 4×8PSK

K = 2.75 bit/sym, q=0, d_u^2 = 1.172, N_u = 24 (4×8PSK).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	-	1	3	45°	1.172	8	1.757	64	0.00
2	2	-	-	2	1	5	45°	1.757	48	-	-	1.76
3	2	-	-	04	02	11	45°	2.0	8	2.343	64	2.32
4	3	-	10	04	02	21	45°	2.343	40	-	-	3.01
5	3	-	30	14	02	41	45°	2.343	8	2.929	288	3.01
6	4	030	020	052	014	101	45°	2.929	136	-	-	3.98

γ_4 = -0.94 dB

TABLE 2.18(b)

TRELLIS CODED 4×8PSK

K = 2.50 bit/sym, q=1, d_u^2 = 1.172, N_u = 4 (2×8PSK).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	1	3	45°	2.0	8	2.343	64	2.32
2	2	-	2	1	5	45°	2.343	40	-	-	3.01
3	2	-	04	02	11	45°	2.343	8	3.172	32	3.01
4	3	14	04	02	21	45°	3.172	16	-	-	4.33
5	3	24	14	02	41	45°	3.515	64	-	-	4.77
6	3	014	024	042	103	45°	4.0	28	4.686	1088	5.33

γ_4 = -1.35 dB

TABLE 2.18(c)

TRELLIS CODED 4×8PSK

K = 2.25 bit/sym, q=2, d_u^2 = 2.0, N_u = 8 (4×8PSK).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	-	1	3	45°	2.343	8	3.172	32	0.69
2	2	-	-	3	1	5	45°	3.172	16	-	-	2.00
3	2	-	-	06	02	11	45°	4.0	28	4.343	64	3.01
	2	-	-	02	06	11	90°	4.0	28	4.686	64	3.01
4	3	-	04	06	12	21	45°	4.0	12	4.686	32	3.01
5	4	10	04	06	22	41	45°	4.0	4	4.686	16	3.01

γ_4 = 0.51 dB

C-2

TABLE 2.18(d)

TRELLIS CODED 4×8PSK

K = 2.00 bit/sym, q=3, d_u^2 = 2.0, N_u = 2 (1×4PSK).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	-	1	3	90°	4.0	28	4.686	64	3.01
2	2	-	-	2	3	5	45°	4.0	12	4.686	32	3.01
3	3	-	02	04	03	11	45°	4.0	4	4.686	16	3.01
4	4	10	04	02	03	21	45°	4.686	8	-	-	3.70
5	4	02	10	04	22	41	45°	6.343	16	-	-	5.01
6	4	034	044	016	036	107	45°	6.686	6	-	-	5.24
	4	044	024	014	016	103	90°	7.029	24	-	-	5.46

γ_4 = 0 dB

TABLE 2.19

TRELLIS CODED 1×16PSK

K = 3.0 bit/sym, d_u^2 = 0.586, N_u = 2 (1×8PSK).

ν	$\tilde{k}$	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	1	3	90°	0.738	2	-	-	1.00
2	1	-	2	5	90°	1.324	4	-	-	3.54
3	1	-	06	13	45°	1.476	8	-	-	4.01
	1	-	04	13	90°	1.476	4	-	-	4.01
4	1	-	06	21	45°	1.476	4	-	-	4.01
	1	-	10	23	90°	1.628	4	-	-	4.44
5	1	-	24	43	45°	1.781	8	-	-	4.83
	1	-	10	45	90°	1.910	8	-	-	5.13
6	1	-	056	135	45°	2.0	2	2.085	16	5.33
	1	-	032	107	90°	2.0	2	2.085	8	5.33
7	1	-	126	235	45°	2.0	2	2.366	16	5.33
8	2	344	162	717	90°	2.085	2.938	-	-	5.51
	2	224	112	527	180°	2.085	1.219	-	-	5.51

γ_8 = 0 dB

TABLE 2.20(a)

TRELLIS CODED 2×16PSK

K = 3.5 bit/sym, q=0, d_u^2 = 0.304, N_u = 4 (2×16PSK).

ν	$\tilde{k}$	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	1	3	45°	0.457	8	-	-	1.76
2	1	-	2	5	45°	0.586	4	0.761	32	2.84
3	2	04	06	11	22.5°	0.761	16	-	-	3.98
4	2	16	12	23	22.5°	0.913	56	-	-	4.77
5	2	10	06	41	22.5°	0.913	16	-	-	4.77
6	2	004	030	113	22.5°	1.066	80	-	-	5.44
	2	044	016	107	45°	1.066	48	-	-	5.44
7	2	074	132	217	22.5°	1.172	4	1.218	228	5.85

γ = -2.17 dB

TABLE 2.20(b)

TRELLIS CODED 2×16PSK

K = 3.0 bit/sym, q=1, d_u^2 = 0.586, N_u = 2 (1×8PSK)

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	1	3	22.5°	0.890	8	-	-	1.82
2	1	-	-	2	5	22.5°	1.172	4	1.476	32	3.01
3	2	-	04	02	11	90°	1.476	16	-	-	4.01
4	2	-	14	06	23	45°	1.757	8	-	-	4.77
5	2	-	30	16	41	45°	1.781	16	-	-	4.83
6	2	-	044	016	107	45°	2.0	4	2.085	48	5.33
7	3	110	044	016	317	45°	2.085	25	-	-	5.51

γ_8 = 0 dB

TABLE 2.21(a)

TRELLIS CODED 3×16PSK

K = 3.67 bit/sym, q=0, d_u^2 = 0.304, N_u = 12 (3×16PSK I).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)	sig. set
1	1	-	-	1	3	22.5°	0.304	4	-	-	0.00	II
2	1	-	-	2	5	22.5°	0.457	16	-	-	1.76	II
3	2	-	04	02	11	22.5°	0.586	6	0.609	16	2.84	I
4	3	14	04	02	21	45°	0.609	12	-	-	3.01	I
	3	10	04	02	21	90°	0.609	8	-	-	3.01	I
5	3	30	14	02	53	45°	0.761	48	-	-	3.98	I
6	3	050	022	006	103	45°	0.890	12	-	-	4.66	I
7	3	056	112	004	225	45°	0.913	84	-	-	4.77	I
	3	100	050	022	255	90°	0.913	76	-	-	4.77	I

γ_8 = 0 dB

TABLE 2.21(b)

TRELLIS CODED 3×16PSK

K = 3.33 bit/sym, q=1, d_u^2 = 0.457, N_u = 8 (3×16PSK II).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)	sig. set
1	1	-	-	1	3	45°	0.586	6	0.609	16	1.08	II
2	2	-	3	1	7	45°	0.738	6	-	-	2.08	II
3	2	-	06	02	11	45°	0.913	16	-	-	3.01	II
	2	-	04	02	11	90°	1.043	24	-	-	3.58	II
4	3	10	04	06	21	22.5°	1.043	12	-	-	3.58	III
	2	-	14	02	27	45°	1.172	12	1.195	24	4.09	II
5	3	34	16	06	41	22.5°	1.172	4	-	-	4.09	III
6	3	032	046	006	103	22.5°	1.218	8	-	-	4.26	III
7	3	014	102	044	203	22.5°	1.370	32	-	-	4.77	III
	3	006	072	062	223	45°	1.476	8	-	-	5.09	III

γ_8 = -1.97 dB

TABLE 2.21(c)

TRELLIS CODED 3×16PSK

K = 3.00 bit/sym, q=2, d_u^2 = 0.586, N_u = 2 (1×8PSK).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)	sig. set
1	1	-	-	1	3	90°	1.043	24	-	-	2.50	II
2	1	-	-	2	5	90°	1.172	12	1.628	144	3.01	II
3	2	-	04	02	11	22.5°	1.172	4	-	-	3.01	III
4	2	-	12	04	27	22.5°	1.628	32	-	-	4.44	III
5	2	-	14	02	41	22.5°	1.628	16	-	-	4.44	III
	2	-	22	14	43	45°	1.757	16	-	-	4.77	III
6	2	-	054	020	115	22.5°	1.757	8	2.085	48	4.77	III
	3	020	004	012	101	45°	2.0	6	2.085	72	5.33	II
	3	050	030	026	101	90°	2.0	6	2.085	60	5.33	II
7	3	060	106	050	213	45°	2.0	6	2.214	56	5.33	III
	3	016	110	052	203	90°	2.0	6	2.343	64	5.33	III

γ_8 = 0 dB

TABLE 2.22(a)

TRELLIS CODED 4×16PSK

K = 3.75 bit/sym, q=0, d_u^2 = 0.304, N_u = 24 (4×16PSK).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	-	1	3	22.5°	0.304	8	0.457	64	0.00
2	2	-	-	2	1	5	22.5°	0.457	48	-	-	1.76
3	2	-	-	04	02	11	22.5°	0.586	8	0.609	64	2.84
4	3	-	10	04	02	21	22.5°	0.609	40	-	-	3.01
5	3	-	30	14	02	41	22.5°	0.609	8	0.761	288	3.01
6	4	030	020	052	014	101	22.5°	0.761	136	-	-	3.98

γ_8 = -1.87 dB

TABLE 2.22(b)

TRELLIS CODED 4×16PSK

K = 3.50 bit/sym, q=1, d_u^2 = 0.304, N_u = 4 (2×16PSK).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	1	3	22.5°	0.586	8	0.609	64	2.84
2	2	-	2	1	5	22.5°	0.609	40	-	-	3.01
3	2	-	04	02	11	22.5°	0.609	8	0.890	32	3.01
4	3	14	04	02	21	22.5°	0.890	16	-	-	4.66
5	3	24	14	02	41	22.5°	0.913	64	-	-	4.77
6	3	014	024	042	103	22.5°	1.172	24	1.218	1088	5.85

γ_8 = -2.17 dB

TABLE 2.22(c)

TRELLIS CODED 4×16PSK

K = 3.25 bit/sym, q=2, d_u^2 = 0.586, N_u = 8 (4×16PSK).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	-	1	3	22.5°	0.609	8	0.890	32	0.17
2	2	-	-	3	1	5	22.5°	0.890	16	-	-	1.82
3	2	-	-	06	02	11	22.5°	1.172	24	1.195	64	3.01
	2	-	-	02	06	11	45°	1.172	24	1.218	64	3.01
4	3	-	04	06	12	21	22.5°	1.172	8	1.218	32	3.01
5	4	10	04	06	22	41	22.5°	1.218	16	-	-	3.18
6	4	050	030	024	016	101	22.5°	1.499	72	-	-	4.08

γ_8 = 0.35 dB

TABLE 2.22(d)

TRELLIS CODED 4×16PSK

$K = 3.00$ bit/sym, $q=3$, $d_u^2 = 0.586$, $N_u = 2$ (1×8PSK).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	1	3	45°	1.172	24	1.218	64	3.01
2	2	-	2	3	5	22.5°	1.172	8	1.218	32	3.01
3	3	02	04	03	11	22.5°	1.218	16	-	-	3.18
4	3	04	10	06	21	22.5°	1.781	48	-	-	4.83
5	3	22	16	06	41	22.5°	1.804	24	-	-	4.88
	3	24	14	02	43	45°	1.827	64	-	-	4.94
6	3	050	024	006	103	22.5°	2.0	8	2.343	64	5.33

$\gamma_8 = 0$ dB

140 Mbit/s. However, with the equivalent rate 7/8 code with 3×8PSK modulation, the bit rate will be $L = 3$ times as fast, i.e., 420 Mbit/s. The branch metric calculator, though, will be more complicated due to the larger number of parallel transitions between states. Alternatively, one could build a decoder operating at a 20 MHz speed and achieve the same bit rate of 140 Mbit/s. In addition to providing decreased decoder complexity, this multi-D code has an asymptotic coding gain which is 0.56 dB greater and is 90° transparent, compared with a 180° transparency for the PTVTC [48].

Although the decoding complexity of the Viterbi algorithm is measured in terms of $2^{\nu+\tilde{k}}/L$, for multi-D schemes the complexity of subset (parallel transition) decoding must also be taken into account due to the large number of parallel transitions.

The Viterbi decoder must find which of the $2^{k-\tilde{k}}$ parallel transitions is closest, in a maximum likelihood sense, to the received

signal. A brute force method would be to determine the metric for each of the $2^{k-\tilde{k}}$ paths and then find the minimum. This would involve at least $2^{k-\tilde{k}}-1$ comparisons. Since there are $2^{\tilde{k}+1}$ sets of parallel transitions, a total of $2^{k+1}-2^{\tilde{k}+1}$ comparisons would be required. For large k and small $\tilde{k}$, this is an unacceptably large number of computations.

Fortunately, as shown in [23] for binary lattices, it is possible to greatly reduce the number of computations required. In fact, the decoding scheme becomes very similar to Viterbi decoding, except that finite length sequences are used.

To illustrate this we will present the decoding scheme for TC-2×8PSK parallel transitions with $\tilde{k} = 2$ and an efficiency of 2.5 bit/sym (a rate 5/6 code). There are eight sets of parallel transitions, with eight paths in each set. Figure 2.12 shows the parallel transition decoding trellis for $\tilde{z} = [0\ 0\ 0]$ (i.e., the three lsb's are set to zero). In Figure 1.2, we use the notation A0 to indicate the whole 8PSK signal set, which divides into B0 and B1 (4PSK signal sets rotated 45° from each other). B0 divides into C0 and C2 (2PSK signal sets rotated 90° from each other), and B1 divides into C1 and C3. This notation is also used in [65] for partitioning an 8PSK signal set. Each segment in Figure 2.12 thus represents two parallel lines. The length of this trellis equals the dimensionality L = 2 of the signal set.

The path C0×C0 corresponds to those four paths that have $z^3 = 0$ and C2×C2 corresponds to those four paths that have $z^3 = 1$, giving a total of eight paths. To decode, hard decisions can be made for C0 and C2 for each time period, from which the values of z^4 and z^5 can be

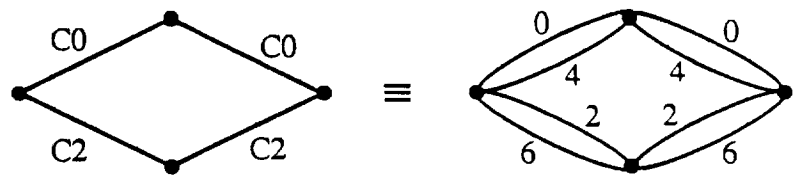

Figure 2.12: The parallel transition decoding trellis for $\tilde{\mathbf{z}} = [0\ 0\ 0]$ and the 2×8PSK signal set.

determined. For example, say that C0×C0 decodes into the points 04, with a metric of m_0, and C2×C2 decodes into the points 66, with a metric of m_1, where the metrics are the sum of the Euclidean distances (or log-likelihood metrics for a quantized channel) from the first and second received points. After comparing the two metrics, if $m_0 < m_1$, then $z^3 = 0$ and the point 04 would give $z^4 = 1$ and $z^5 = 0$ (see Table 2.1). If $m_0 > m_1$, then $z^3 = 1$, and the point 66 would give $z^4 = 0$ and $z^5 = 1$. This is equivalent to the add-compare-select (ACS) operation within a Viterbi decoder.

To decode the other sets of parallel transitions, the cosets formed by z^0, z^1, and z^2 can be added to the trellis paths C0×C0 and C2×C2 to form the required trellis. This is illustrated in Figure 2.13, where the ending state in the trellis indicates which set of parallel transitions is being decoded. In this example, there are a total of eight hard comparisons and eight ACS type comparisons. These 16 comparisons compare with the 56 comparisons required in a brute force approach, a 3.5 times reduction.

The above maximum likelihood method can be applied to other codes where a Viterbi like decoder can be used to decode the parallel transitions. With this method, the complexity of decoding the parallel transitions can approach the complexity of the rate $\tilde{k}/(\tilde{k}+1)$ Viterbi

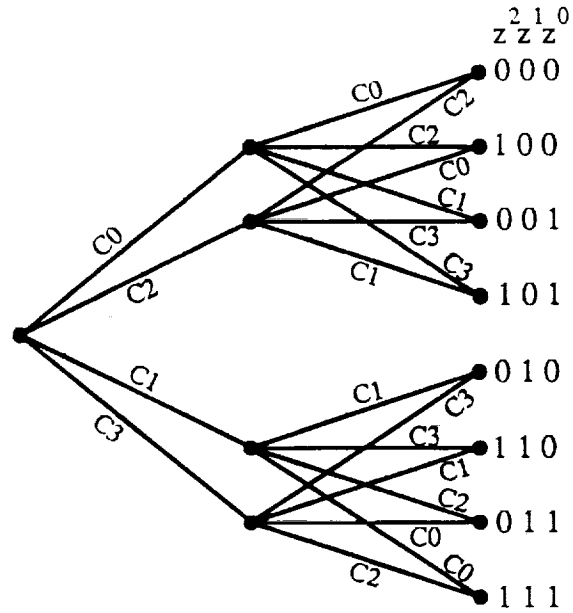

Figure 2.13: The full parallel transition decoding trellis for the 2×8PSK signal set.

decoder. A simpler approach may be with large look-up tables using ROM's. The ROM itself would output the $k - \tilde{k}$ bits of the chosen path, along with the branch metric for that path. For the TC-2×8PSK example given previously, we could use one ROM for each set of parallel transitions. If the ROM's had eight bit words, then three bits could be used for the decision, and the remaining five bits for the branch metric. A total of eight ROM's would then be required, one for each set of the parallel transitions.

When using ROM's, it is desirable to reduce the number of bits (b) required to represent each received 2-D signal point, since there are a total of bL bits required to address the ROM. One way to reduce b is to convert the "checkerboard" (rectangular) type decision boundaries that result from separate quantization of the inphase (I) and quadrature (Q) components to "pie chart" (radial) type decision boundaries. For example, if four bits are used in I and Q for an 8PSK

signal with checkerboard decision boundaries, a pie chart pattern as shown in Figure 2.14 may be used instead with a total of five bits to represent each point (a reduction of three bits). A ROM may be used to do the conversion, or the pie chart pattern may be already available as polar coordinates from a digital demodulator.

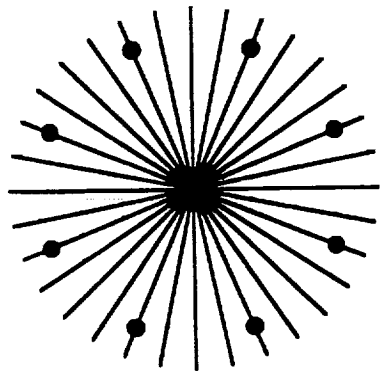

Figure 2.14: Pie chart decision boundries for 8PSK (32 regions).

A problem with TC-L×MPSK is the need to synchronize the decoder with the L 2-D symbols on each trellis branch. For $q = 0$, most codes are fully transparent. The decoder performance can then be used to find the correct synchronization with the received sequence. For $q > 0$, many codes are not fully transparent, and the decoder will need to synchronize to one of the $2^d L$ possibilities (which can be quite large for some codes). However, one can take advantage of the fact that not all signal points are used for $q > 0$. For example, the 2×8PSK signal set with $q = 1$ consists of the signal sets B0×B0 or B1×B1. The synchronizer would find the smallest distance between a received pair of points and the expected signal set. These distances would then be accumulated over a sufficient length of time to make a reliable decision on the symbol timing.

If we let each signal point be represented by its phase (since the amplitude is constant for 8PSK), we can write B0 = {0°,90°,180°,270°}, and B1 = {45°,135°,225°,315°}. Let ϕ_n^1 and ϕ_n^2 represent the phase of the first and second received symbols, respectively. The synchronizer distance metric is then given by

$$\Phi_n = \min_{i \in \{0,1\}} \left(\min_{\alpha \in Bi} |\phi_n^1 - \alpha| + \min_{\beta \in Bi} |\phi_n^2 - \beta| \right).$$

In the synchronized noiseless case, Φ_n will equal zero. In the non-synchronized noiseless case, there are two possible outcomes for Φ_n, i.e., complete matchup ($\Phi_n = 0°$) and only one signal is matched ($\Phi_n = 45°$). If each possiblity is equally likely, then the average value of Φ_n is 22.5°. With noise, Φ_n can be accumulated over a sufficient length of symbols to take advantage of this average phase distance between the non-synchronized and synchronized cases to reliably determine symbol synchronization. This symbol synchronization is independent of the Viterbi decoder, so the decoder must only determine phase synchronization.

2.2.9 Discussion

In order to make a comparison of all the codes listed, a plot of *nominal coding gain* $\gamma^* = 10\log_{10} d^2_{free}$ verses complexity ($\beta = \log_2(2^{\nu+\tilde{k}}/L) = \nu + \tilde{k} - \log_2 L$) for each code found is made. These plots are given in Figure 2.15 for effective rates of 1.0 (with 4PSK modulation), 2.0 (8PSK), and 3.0 bit/sym (16PSK), Figure 2.16 for effective rates of 1.5 (4PSK), 2.5 (8PSK), and 3.5 bit/sym (16PSK), and Figure 2.17 (for the remaining rates). (Note that these graphs do not

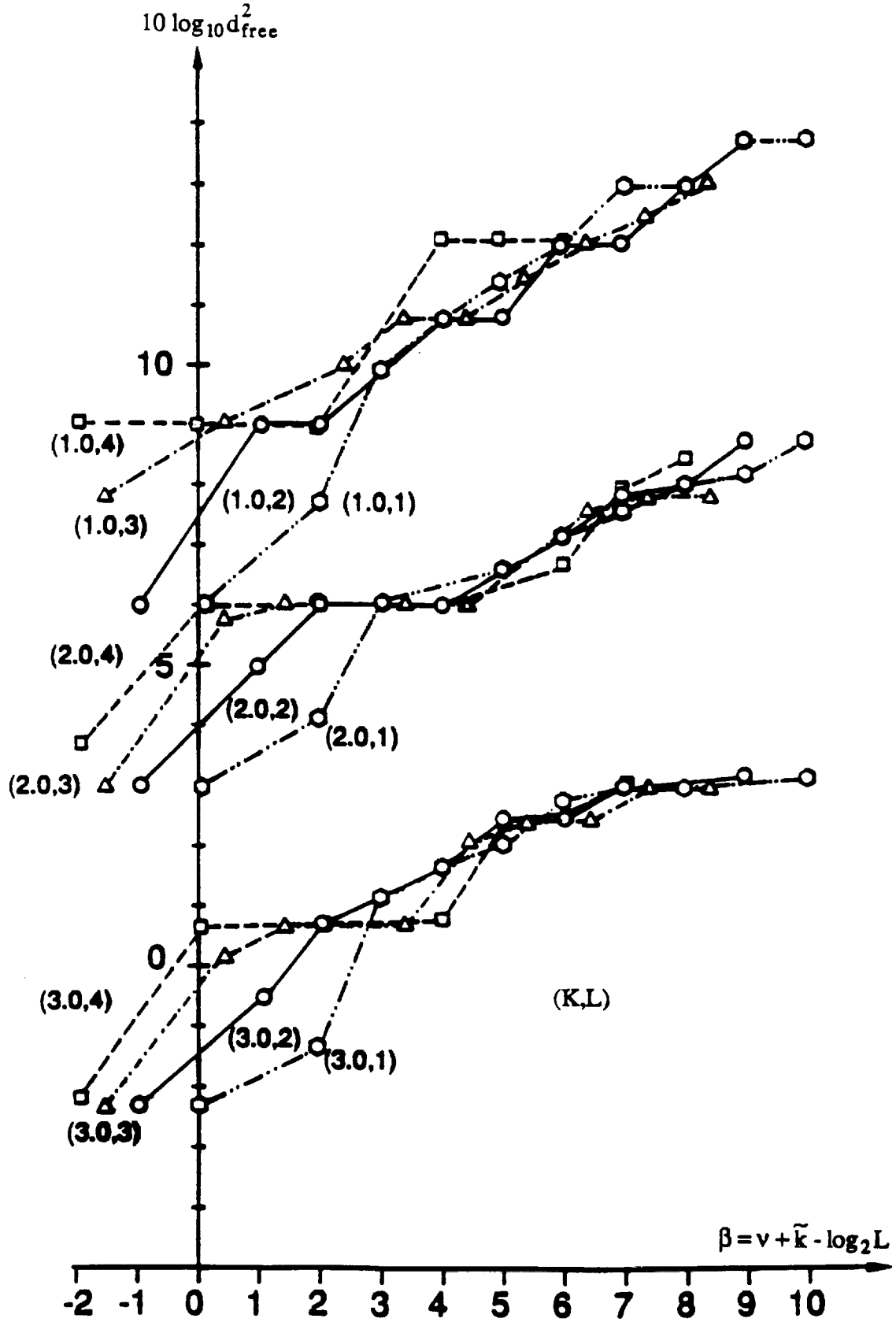

Figure 2.15: Plot of $10 \log_{10} d^2_{free}$ verses complexity β for K = 1.0, 2.0, and 3.0 bit/sym.

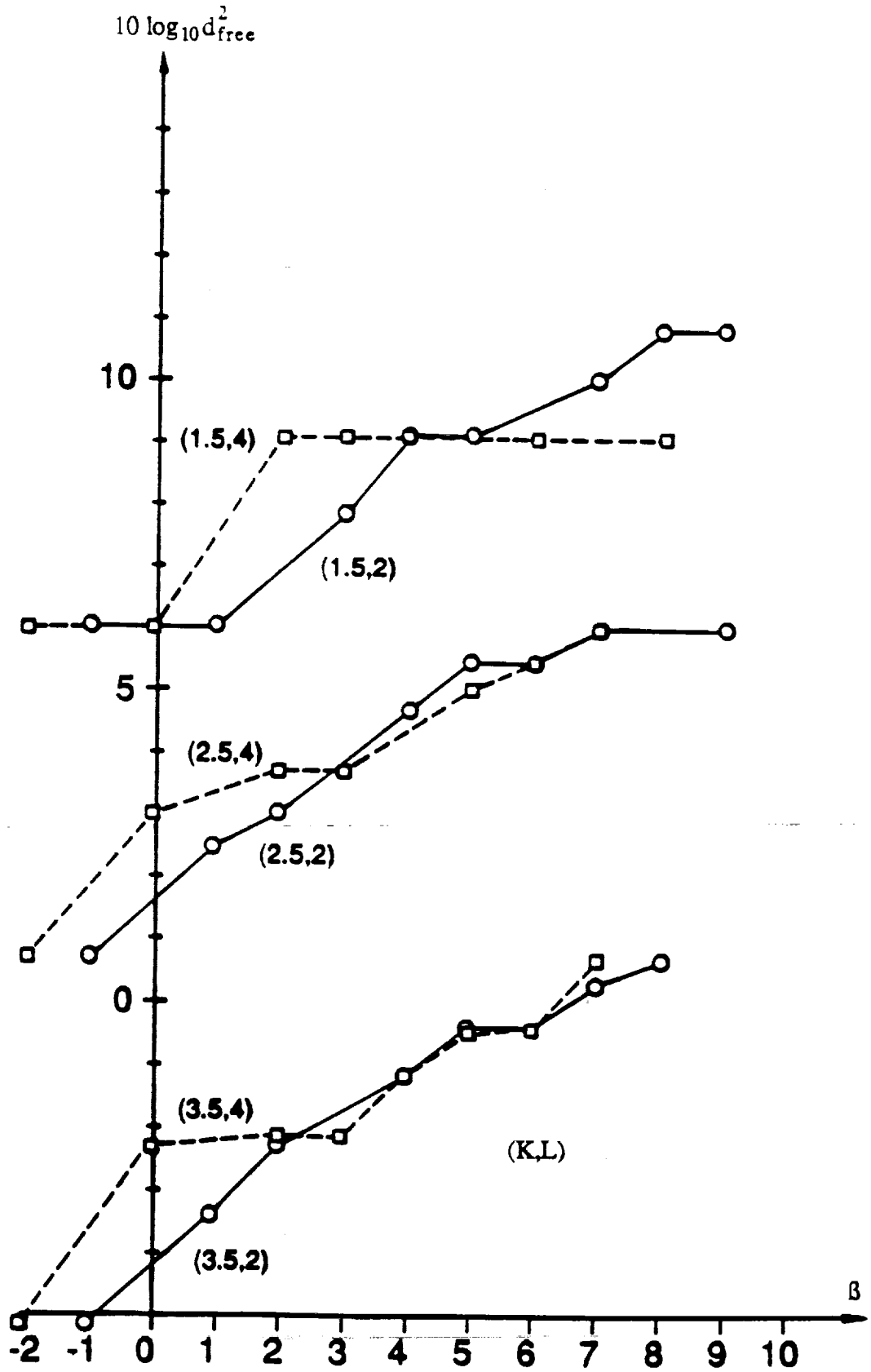

Figure 2.16: Plot of $10 \log_{10} d^2_{free}$ verses complexity ß for K = 1.5, 2.5, and 3.5 bit/sym.

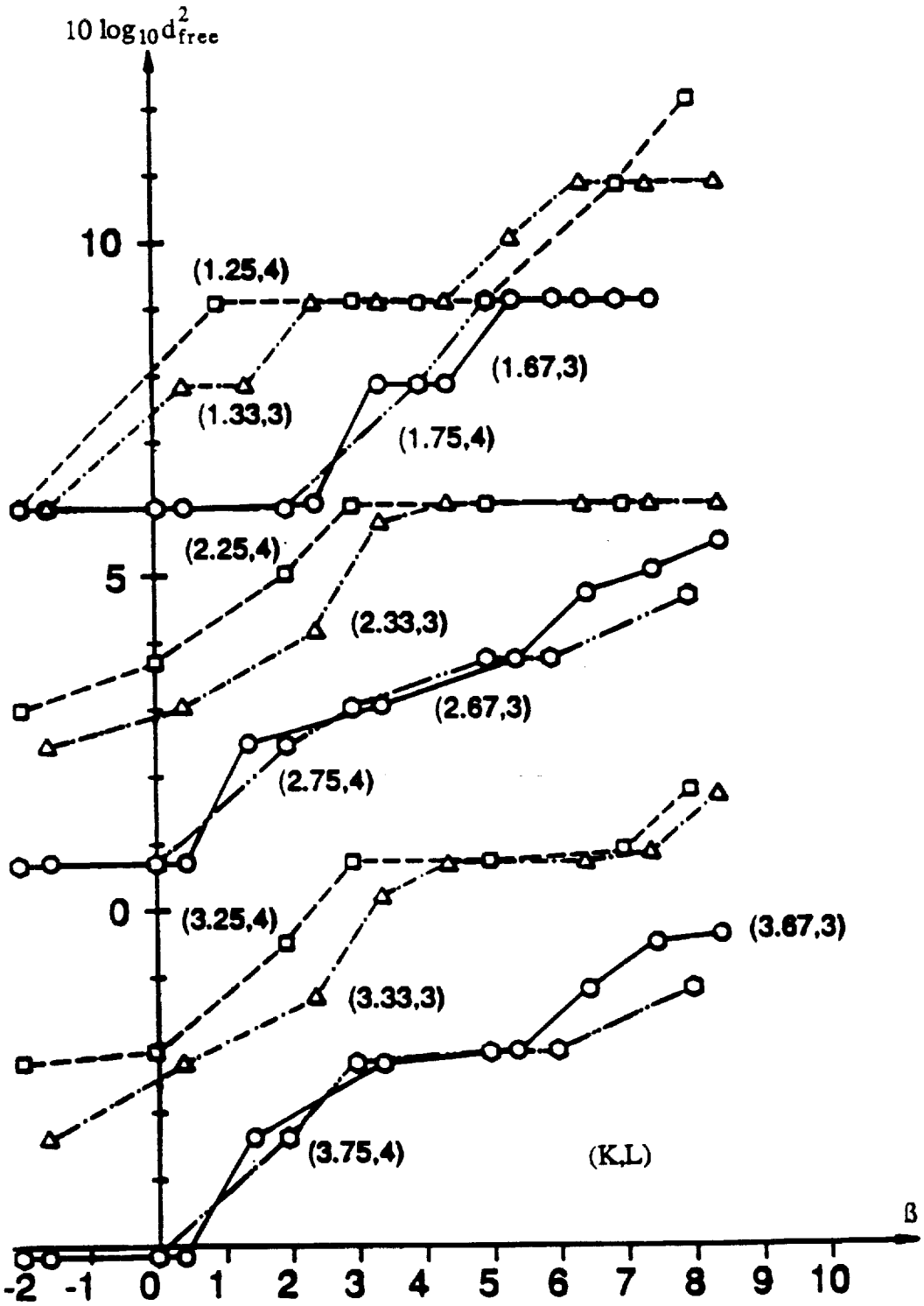

Figure 2.17: Plot of $10 \log_{10} d^2_{free}$ verses complexity ß for K = 1.25, 1.33, 1.67, 1.75, 2.25, 2.33, 2.67, 2.75, 3.25, 3.33, 3.67, and 3.75 bit/sym.

take into account the additional complexity due to parallel transitions.) Some one state ("uncoded") codes are included as well. These one state codes correspond to block coded (or multilevel) schemes that have recently become an active research area [5,11,13,27,31,34-36,41,59,61,63,80]. Although the multi-D one state codes have negative complexity (compared to trellis codes), they can achieve coding gains above 0 dB. There has also been research in multi-level schemes which use convolutional codes [78].

Note from Figure 2.15 for TC-L×8PSK, K = 2.0 bit/sym, and $\nu = 1$, that as L increases the complexity decreases and γ^* increases, eventually reaching 6.0 dB for L = 4. Thus, for the 8-D signal set, the complexity factor can be reduced by a factor of four, while maintaining γ^*, compared to the TC-1×8PSK code with $\nu = 2$. Beyond $\beta = 4$ (and $\gamma^* = 6.0$ dB), increases in asymptotic coding gain are achieved with the new codes that have been found. With L = 4, a ceiling of $\gamma^* = 9.0$ dB will be reached due to the nature of the set partitioning. It would seem that very complex codes are required ($\beta \geq 15$) if this 9.0 dB limit is to be exceeded.

Figure 2.15 also shows the L×16PSK codes with effective rates of 3.0 bit/sym. For small β, the same effect observed for TC-L×8PSK and 2.0 bit/sym occurs. That is, β decreases and γ^* increases as L increases. Between $\beta = 3$ and $\beta = 9$, the L = 1 and L = 2 codes are very close.

Figure 2.17 illustrates the wide range of performance that can be achieved with the codes found. One can choose from a high rate code with 3.75 bit/sym (but requiring a large amount of power) to a low rate code with 1.25 bit/sym. In choosing a code, a designer may start with a

required K in order to obtain a certain bit rate through a bandwidth constrained channel. A trade-off can then be made between decoder complexity and the reduction in SNR that can be achieved with the codes found. Simulations or theoretical calculations of a few selected codes may also be made in order to obtain a more realistic assesment of the performance available.

Note that many codes have the same asymptotic coding gain for increasing complexity. In reality, these codes do increase in performance with increasing complexity due to a decrease in number of nearest neighbors. This is especially noticeable for low SNR where the effect of nearest neighbors becomes more important.

2.3 Conclusions

An efficient method of partitioning multi-dimensional MPSK signal sets has been presented that leads to easily implemeted multi-D signal set mappers. When these signal sets are combined with trellis codes to form a rate k/(k+1) code, significant asymptotic coding gains in comparison to an uncoded system are achieved. These codes provide a number of advantages compared to trellis codes with 2-D signal sets. Most importantly, K can vary from I-1 to I-(1/L) bit/sym, allowing the coding system designer a greater choice of data rates without sacrificing data quality. As K approaches I, though, increased coding effort (in terms of decoder complexity) or higher SNR is required to achieve the same data performance.

The analytical description of multi-D signal sets in terms of block code cosets, and the use of systematic convolutional encoding,

has resulted in an encoder design (from the differential encoder to the 2-D signal set mapper) that allows many good codes to be found. This approach has also led to the construction of signal sets that allow codes to be transparent to multiples of 360°/M phase rotations. In general, increasing phase transparency usually results in lower code performance, due to more nearest or next nearest neighbors or smaller free distance.

Another advantage is decoder complexity. As a Viterbi decoder decodes k bits in each recursion of the algorithm, the large values of k of codes using multi-D signal sets allows very high bit rates to be achieved (compared to convolutional codes that map only into a 2-D signal set). The large number of branch metric computations can be reduced either through the use of a modified Viterbi algorithm or large look up tables. A method has been presented that uses the redundancy in some signal sets to achieve symbol synchronization at the decoder for codes that are not fully transparent.

CHAPTER THREE

TRELLIS CODING WITH MULTIDIMENSIONAL QAM SIGNAL SETS

The first work on trellis coding with Quadrature Amplitude Modulation (QAM) signal sets was presented by Ungerboeck in [65]. This work was extended in [58,72] for two-dimensional (2-D) signal sets and in [8-10,12,22-25,55,64,68,74] for multidimensional (multi-D) signal sets. The reason for the large activity in this area is due to the advantages that coding with an expanded signal set allows, viz., high bandwidth efficiency at a reduced Signal-to-Noise-Ratio (SNR) compared to an uncoded system.

Many of the codes presented in the literature have been found in an ad hoc manner. There has been no attempt, as far as the author is aware, to find the best codes with the fewest number of nearest neighbors. Also, many authors use infinite size signal sets in the design of their codes. Although this simplifies the search for good codes and leads to an easier understanding of multi-D signal sets, the codes produced may not be optimum for practical finite size signal sets.

This chapter addresses this problem and presents the results of a systematic code search for trellis codes with multi-D QAM signal sets. The 2-D signal sets used in the construction of the multi-D signal sets range from 16 to 512 points and were designed to have minimum energy, be 90° rotationally symmetric, and be suitable for partitioning. Where

possible, the signal set selected is the same as that commonly used in the literature and practical implementations of TCM schemes.

Only rate k/(k+1) codes are used in the code search. Rotationally invariant nonlinear codes for the 2-D signal sets presented in this chapter are given in Chapter 5. Also, the codes presented all have signal sets with 2^I signal points, where I is an integer greater than zero. Some of the advantages of multi-D signal sets are: possible 90° phase invariance, lower coding complexity, and non-integer bandwidth efficiency (K bits per 2-D symbol, or bit/sym). The multi-D signal sets were constructed by a method similar to that in Chapter 2, i.e., through the use of cosets.

In our code search we sought to find the codes which have the largest minimum free Euclidean distance (d_{free}) and of those codes, the code with the smallest number of nearest neighbors (N_{free}). In this way we maximize the asymptotic coding gain (γ) and minimize the probability of event error (P_e) at high SNR. There are usually two different possible phase transparencies for a linear code, and the best codes for each phase transparency are presented. The values of K range from 3 to 8 bit/sym, with signal sets up to eight dimensions for 16QAM and 32CROSS, six dimensions for 64CIRC, four dimensions for 128CROSS and 256CIRC, and two dimensions for 512STAR. (The 64CIRC, 256CIRC, and 512CROSS are new 2-D signal sets.) The codes have γ ranging up to 6 dB.

3.1 Construction of 2-D QAM Signal Sets

In designing the 2-D signal sets used in this chapter we considered three criteria. The most important is that the signal set

can be partitioned such that the minimum squared subset distance (MSSD) at partition level i, δ_i^2, is as large as possible. The values of i range from 0 for the full signal set to i = I where there are $M = 2^I$ subsets, with each subset having a single point (and $\delta_I^2 = \infty$). We will be using a binary partitioning where each subset is divided into two equal size subsets (in terms of the number of points) at the next partition level.

This can be achieved with rectangular or QAM signal sets. These signal sets are finite subsets of the $\mathbb{Z}^2 + (0.5,0.5)$ infinite size signal set ($\mathbb{Z}$ is the set of all integers). Letting δ_0^2 be normalized to one, we then have $\delta_{i+1}^2 = 2\delta_i^2$, for $0 \le i \le I-1$ (for finite signal sets there is usually some i for which this is not true). As shown later, we can obtain good codes as long as $\delta_1^2 = 2$, $\delta_2^2 = 4$, and $\delta_3^2 = 8$.

The second criterion is to choose the shape of the signal set to maximize the shaping gain, i.e., determining where the signal points are placed so as to minimize the energy of the constellation. From [65], we note that the lower bound on d_{free}^2 for many trellis codes is $2\delta_1^2$. This indicates that we should first design a 2^{I-1} point signal set that has minimum energy (and thereby maximize d_{free}^2).

The third criterion is that the signal set should have 90° rotational symmetry. This allows a demodulator to only have to lock within a 90° range, resulting in faster lock times [24]. This criterion can be incorporated into the above criteria by designing a 2^{I-1} point constellation that is 180° rotationally symmetric. By rotating this signal set 90° to form the second subset, a 90° symmetric 2^I point signal set is obtained. The 2^{I-1} point subset should not have any points that map onto points in the subset produced by a 90° rotation.

This prevents the 2^I point signal set from having $\delta_0^2 = 0$.

The method followed was to design the 2^{I-1} point subset from a finite subset of the $2^{0.5}(\mathbb{Z}^2 + (0,0.5))$ infinite lattice. This signal set has points along the vertical axis, but none along the horizontal axis. In practice, a subset of the $2\mathbb{Z}^2 + (1,0)$ infinite lattice is used in order to avoid working with fractions.

After obtaining the 2^I point signal set, we rotate the entire signal set by 45° to obtain our final rectangular signal set. The reason for this 45° rotation is to reduce the number of points in the inphase and quadrature components of the signal set. This results in a simpler modulator design.

3.1.1 Construction of 4, 8, 16, and 32 Point Signal Sets

Although codes are not given for the four and eight point signal sets, we will present their construction since they provide insight into the problem of constructing good signal sets. To construct a four point signal set, we start with the BPSK signal set and rotate it by 90° to form the second subset. Rotating these two subsets by 45° gives the QPSK signal set.

Figure 3.1 shows the three steps needed to obtain the eight point signal set. This signal set does not produce good codes because of its high energy (the energy $E_8 = 1.5$ with $\delta_0^2 = 1$). This results in d_{free}^2 having a lower bound of $2\delta_1^2/E_8 = 2.667$ (signal set energy normalized to one). Compare this with naturally mapped 8PSK where d_{free}^2 has a lower bound of 4. The 8PSK signal set can be constructed in a similar fashion to the rectangular signal sets. That is, we start with the QPSK signal set (where there are no points on the vertical or horizontal axis),

rotate the set by 45° (instead of 90°) to obtain the second subset, and then rotate these two subsets by 22.5° to obtain the final signal set.

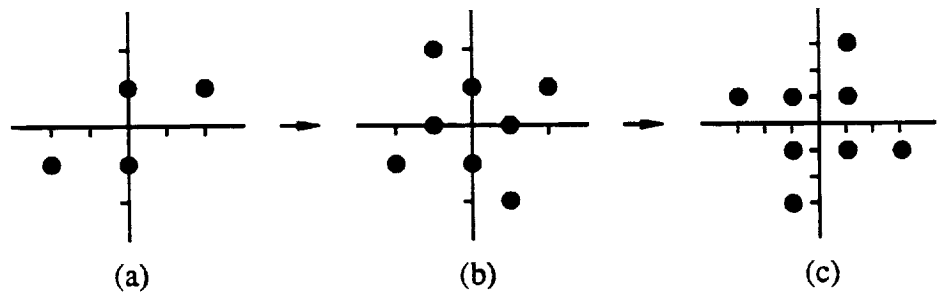

Figure 3.1: Construction of an eight point signal set.

Another eight point signal set is 8AMPM (which is 180° rotationally symmetric). This signal set has an energy of $E'_8 = 1.25$ (for $\delta_0^2 = 1$). Thus, the lower bound for codes using this signal set is $2\delta_1^2/E'_8 = 3.2$. This is better than the 8QAM signal set constructed previously, but still worse than 8PSK. The 8AMPM signal set has $\delta_0^2/E'_8 = 0.8$, compared with 0.586 for 8PSK. Thus, it appears that for a suffuciently large memory encoder, the 8AMPM codes may be able to "catch up" to the 8PSK codes due to larger "in-between" distances (i.e., 2-D symbols that are seperated by δ_0^2 between trellis path pairs). A total of (4-3.2)/(0.8-0.586) = 3.7 in-between distances would be required to make up the difference.

The eight point signal set seems to be the only case where the best signal set is not rectangular. For $I \geq 4$, our construction method works very well. Figure 3.2 illustrates the construction sequence of the 16 point signal set. In this case we start with the 8AMPM signal set to form the standard 16QAM signal set. This example illustrates the reason for rotating the signal set by 45°. In the unrotated signal set

there are seven different amplitude levels along each axis (requiring 3 bits to represent each axis), whereas the rotated signal set has four amplitude levels (requiring only 2 bits per axis).

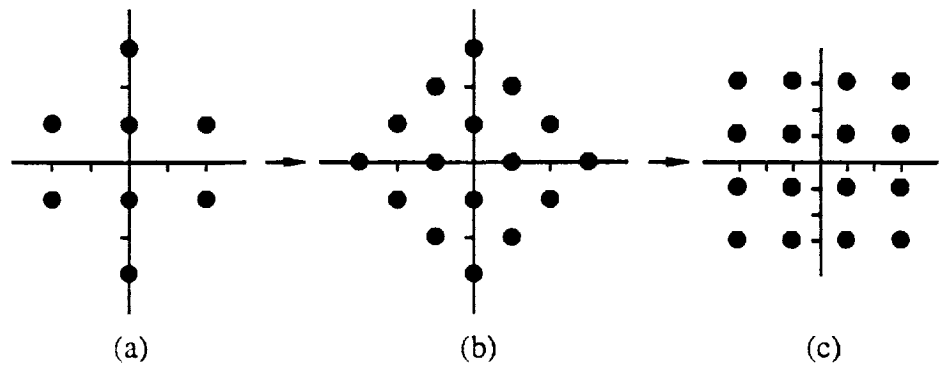

Figure 3.2: Construction of a 16 point signal set (16QAM).

For the 32 point signal set, the starting signal set is 16CROSS (which has the same energy as 16QAM). This gives the standard 32CROSS signal set. The 16QAM and 32CROSS signal sets are illustrated in Figures 3.3 and 3.4, respectively.

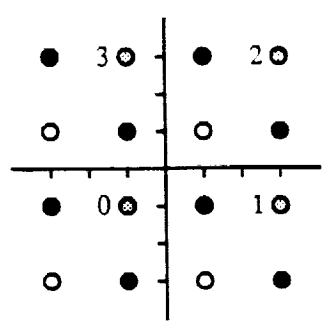

Figure 3.3: Partitioning of 16QAM signal set.

The bits $y^0, y^1, \ldots, y^{I-1}$ are used to represent each point in the signal set and correspond to the 1st, 2nd,..., I^{th} levels of partitioning. We use three shadings of points to illustrate the first

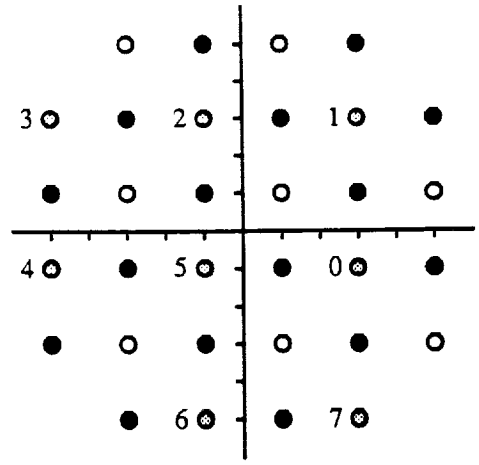

Figure 3.4: Partitioning of 32CROSS signal set.

three levels of partitioning. The black points represent the first level subset with $y^0 = 1$ while the white and gray points represent the first level subset with $y^0 = 0$. The gray points represent the second level subset with $y^0 = 0$ and $y^1 = 0$.

Bits y^0 and y^1 are used to label the first two levels of partitioning. The remaining bits, $y^2,\ldots,y^{I-1}$ are given in integer notation ($y' = \Sigma_{i=2}^{I-1} 2^{i-2}y^i$) next to the points in the grey second level subset. Let $\omega^2(2y^1 + y^0)$ represent a subset at the second partition level. Thus, $\omega^2(0)$ is the first subset (which is given by the gray points). By rotating $\omega^2(0)$ 90° we obtain $\omega^2(1)$. Rotating $\omega^2(1)$ by 90° gives $\omega^2(2)$, and so on. In this way, the whole signal set can be constructed from $\omega^2(0)$. We call this a "natural" mapping since the bits y^0 and y^1 are changed in exactly the same manner as in naturally mapped QPSK.

Let $\omega^2(2y_r^1 + y_r^0)$ represent the subset that results from rotating $\omega^2(2y^1 + y^0)$ by 90°, where y_r^1 and y_r^0 are the bits correponding to the rotated subset. Then

$$2y_r^1 + y_r^0 = 2y^1 + y^0 + 1 \pmod 4. \tag{3.1}$$

We can also express y_r^1 and y_r^0 in binary notation. That is,

$$y_r^0 = y^0 \oplus 1 = \overline{y^0}, \tag{3.2}$$

$$y_r^1 = y^1 \oplus y^0. \tag{3.3}$$

Since the y' bits stay with their signal sets after a rotation, the bits $y^2,\ldots,y^{I-1}$ are unaffected by a phase rotation (i.e., $y_r^i = y^i$, for $2 \le i \le I-1$).

The y' labels an Figures 3.3 and 3.4 are such that we try to double δ_i^2 with each partition. The 16QAM signal set has $\delta_i^2 = 2^i$ for all i. The 32CROSS signal set has $\delta_i^2 = 2^i$ for $i = 0$ to 3 and $\delta_4^2 = 8$ (instead of 16). All the signal sets in this section (QPSK, 8PSK, 16QAM, and 32CROSS) are well known and have been used in practical communication systems. The following section describes the construction of four larger signal sets, three of which are new.

3.1.2 Construction of 64, 128, 256, and 512 Point Signal Sets

Using the method described at the beginning of this chapter, we obtain the 64 point signal set as shown in Figure 3.5. This is basically a 64QAM signal set, except that we have taken a point from each corner and placed it at a less energetic point. The energy of this new signal set (which we have named 64CIRC) is 10.25. This is less than the energy of 64QAM which is 10.5 (a 0.105 dB difference). This signal set has $\delta_i^2 = 2^i$ for $i = 0$ to 4 and $\delta_5^2 = 16$ (instead of 32).

For the 128 point signal set, the construction was more difficult. The minimum energy 64 point signal set was found to have

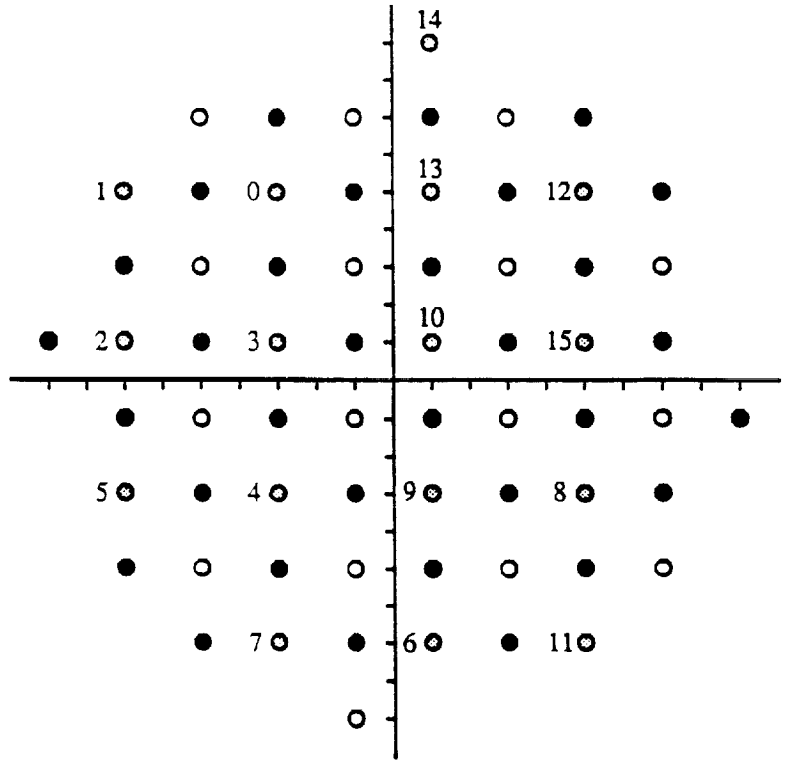

Figure 3.5: Partitioning of 64CIRC signal set.

$\delta_3^2 = 4$. Some of the outer points had the same energy, and so there were several ways to construct different signal sets with the same minimal energy. All of these were found to have $\delta_3^2 = 4$ (instead of 8). A suboptimum signal set was then found (with the next smallest energy). There were two constellations, one of which is the 128CROSS signal set. This is the signal set given in Figure 3.6. The 128CROSS signal set has an energy of 20.5 (compared with 20.4375 for the minimal energy signal set). The difference between the two signal sets is only 0.013 dB, a negligible amount. The 128CROSS signal set has $\delta_i^2 = 2^i$ for $i = 0$ to 5 and $\delta_6^2 = 32$ (instead of 64).

The 256 point signal set has only one solution. Figure 3.7 illustrates the constellation which we have named 256CIRC. The 256CIRC signal set has an energy of 40.6875 compared to the 42.5 for 256QAM.

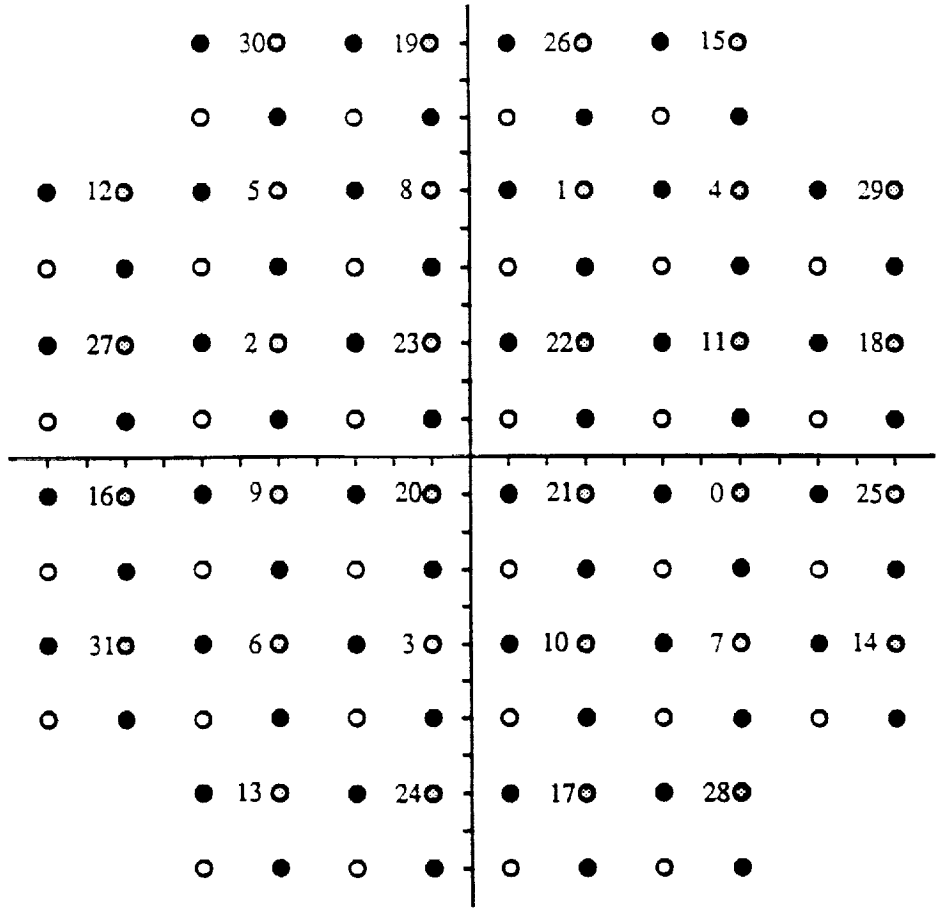

Figure 3.6: Partitioning of 128CROSS signal set.

The difference in energy is 0.189 dB. We have $\delta_i^2 = 2^i$ for $i = 0$ to 3, with $\delta_3^2 = 8$ for $i = 4$ to 6, and $\delta_7^2 = 64$.

As for other signal sets with an odd number of points, the "optimum" signal set for 512 points (in terms of energy) could not be found with $\delta_3^2 = 8$ (the best was $\delta_3^2 = 4$). A "suboptimum" signal set was found with an energy of 81.6875. The optimum signal set has an energy of 81.546875, a difference of only 0.007 dB. Figure 3.8 gives the signal set and its partial two-way partition. We have called this signal set 512STAR. This signal set has $\delta_i^2 = 2^i$ for $i = 0$ to 4, $\delta_5^2 = 16$, $\delta_6^2 = 16$, $\delta_7^2 = 80$, and $\delta_8^2 = 128$.

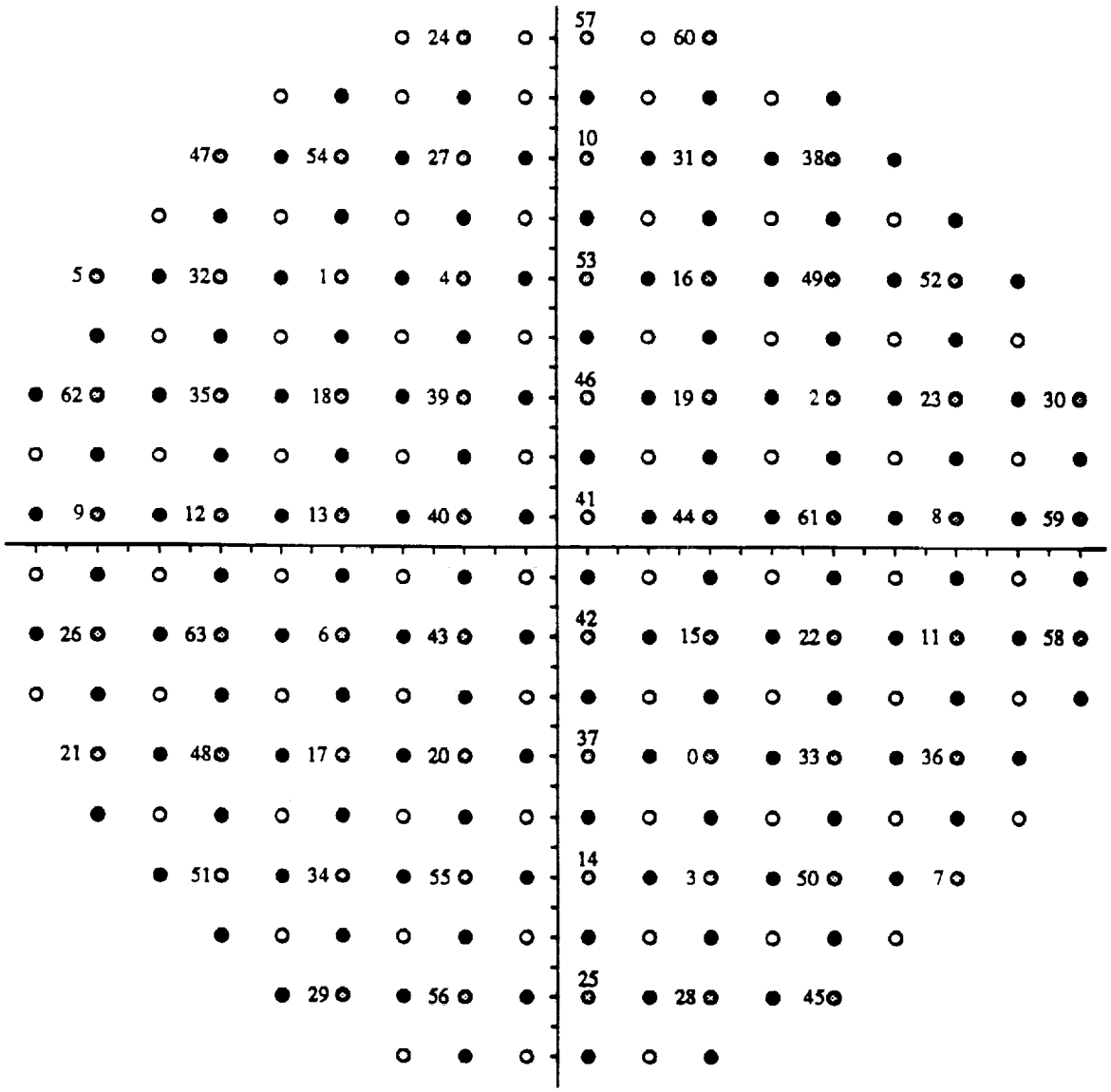

Figure 3.7: Partitioning of 256CIRC signal set.

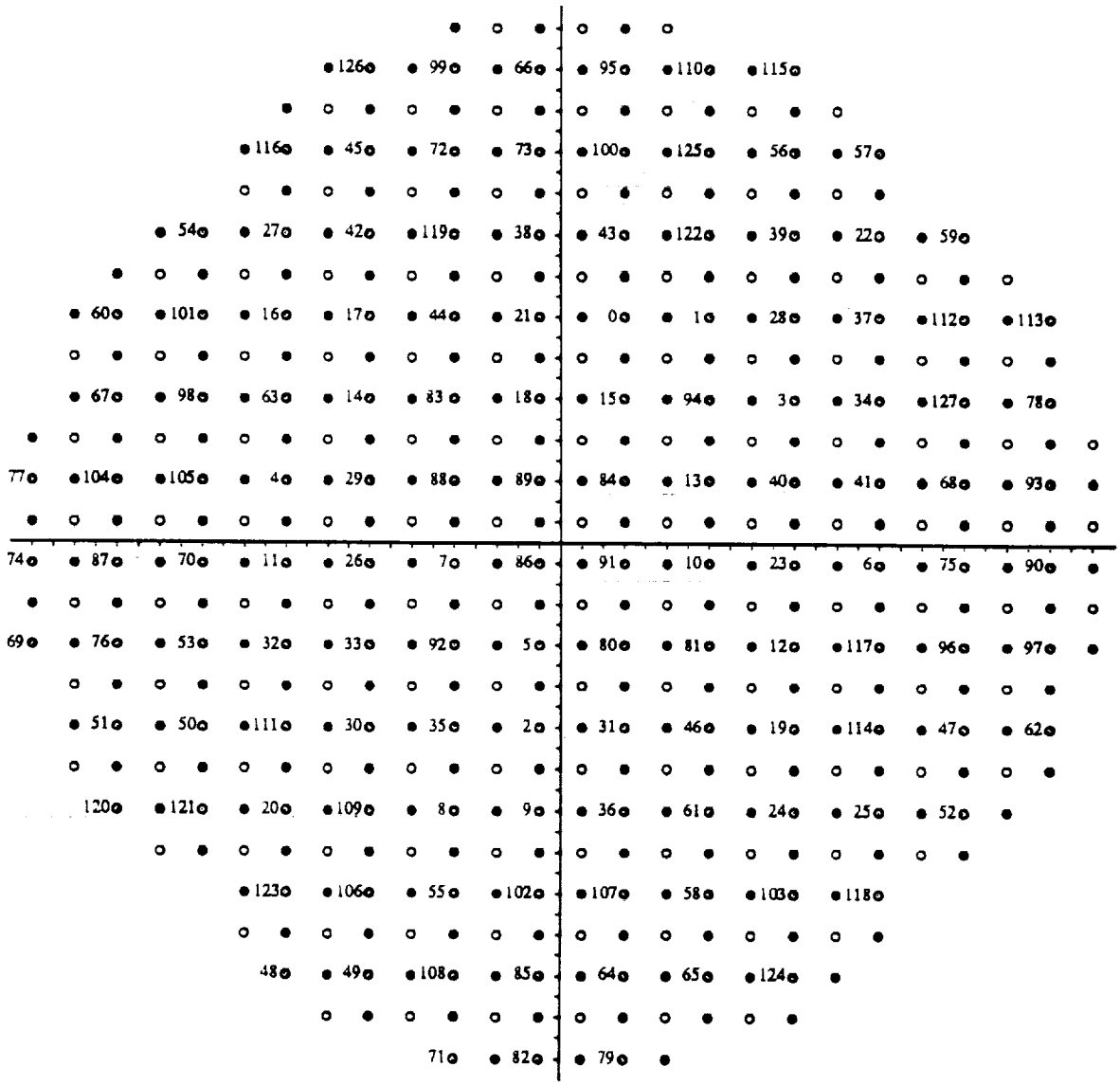

Figure 3.8: Partitioning of 512STAR signal set.

3.2 Trellis Coded Multi-D QAM Design

In this section we will describe how multi-D QAM signal sets are constructed. Also, the general encoder system and the results of our code search are presented.

3.2.1 Construction of Multidimensional QAM Signal Sets

The method used to construct the 4-D, 6-D, and 8-D signal sets is very similar to that described in Chapter 2 for multidimensional MPSK signal sets. We will assume that the reader is familiar with this construction method.

The length L block codes used in our construction are the same as those given in Chapter 2. Table 3.1 gives the minimum squared subset distance (Δ_p^2) and the generator (t^p) at partition level p for the multi-D 16QAM signal sets. To obtain the cosets for the larger size signal sets, the generators in Table 3.1 are used along with an extra (I-4)L cosets. For example, the extra cosets for 4-D 32CROSS would be $(t^8)^T = (0\ 16)^T$ and $(t^9)^T = (16\ 16)^T$. For the 8-D signal sets, the order of the cosets changes slightly. For 8-D 32CROSS we have $(t^{15})^T = (0\ 0\ 0\ 16)^T$, $(t^{16})^T = (8\ 8\ 8\ 8)^T$, $(t^{17})^T = (0\ 0\ 16\ 16)^T$, $(t^{18})^T = (0\ 16\ 0\ 16)^T$, and $(t^{19})^T = (16\ 16\ 16\ 16)^T$.

The total number of bits used to map a multi-D signal set is equal to IL (where L is the number of 2-D signal sets). This may be reduced by q bits (q < L) for lower code rates. In the search for good trellis codes, the computation of the Euclidean weights is proportional to $2^{2(IL-q)}$. We have limited IL-q to at most 17 (2^{34} computations); otherwise the code searches would have taken too long.

TABLE 3.1

SUMMARY OF L×16QAM PARTITIONS

	L = 2		L=3 (I)		L=3 (II)		L=3 (III)		L = 4	
Partition Level (p)	MSSD (Δ_p^2)	gen. ($t^{p\}}$)	MSSD (Δ_p^2)	gen. ($t^{p\}}$)	MSSD (Δ_p^2)	gen. ($t^{p\}}$)	MSSD (Δ_p^2)	gen. ($t^{p\}}$)	MSSD (Δ_p^2)	gen. ($t^{p\}}$)
0	1	01	1	111	1	001	1	001	1	0001
1	2	11	2	110	1	011	1	011	2	0011
2	2	02	2	011	2	222	2	002	2	0101
3	4	22	2	222	3	111	2	022	2	0002
4	4	04	4	220	4	220	3	111	4	1111
5	8	44	4	022	4	022	4	444	4	0022
6	8	08	4	444	4	444	6	222	4	0202
7	16	88	8	440	8	440	8	440	4	0004
8	-	-	8	044	8	044	8	044	8	2222
9	-	-	8	888	8	888	8	888	8	0044
10	-	-	16	880	16	880	16	880	8	0404
11	-	-	16	088	16	088	16	088	8	0008
12	-	-	-	-	-	-	-	-	16	4444
13	-	-	-	-	-	-	-	-	16	0088
14	-	-	-	-	-	-	-	-	16	0808
15	-	-	-	-	-	-	-	-	32	8888
p_0 p_1	1	3	0	3	3	2	4	6	4	8

The main difference in terms of signal mapping between MPSK and QAM is the effect of phase rotations on the mapping. For MPSK, all bits $y^0, y^1, \ldots, y^{I-1}$ are affected. For QAM signal sets, as shown in Section 3.1.1, only the two least significant bits are affected.

To describe how the generators are added to form the multi-D signal set, we first introduce some notation. Let z^0 to z^{IL-q-1} be the IL-q bits that map into the multi-D signal set. The integers $y_1, \ldots, y_L$ are the representations of each 2-D point in the 2L-D signal set. That is,

$$y_j = \sum_{i=0}^{I-1} 2^i y_j^i, \tag{3.4}$$

where $y_j^i \in \{0,1\}$. In general, a multi-D point $y = [y_1,\ldots,y_L]^T$ is expressed as follows,

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_L \end{bmatrix} = \sum_{j=q}^{IL-1} z^{j-q} t^j, \tag{3.5}$$

where $0 \le q < L$. The summation in (3.5) can be taken in a number of ways. Since the signal sets in this chapter are 90° symmetric (that is, there are four rotational symmetries), some of the cosets in (3.5) are added modulo-4, while the remaining cosets are added bit-wise modulo-2.

Using the preferred method of addition, and assuming $q = 0$, we can express (3.5) as

$$\begin{bmatrix} y_1 \\ \vdots \\ y_L \end{bmatrix} = \left[(2z^{p_1} + z^{p_0}) \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} + 2\{g_1\} + \{g_0\} \pmod 4 \right] + 2^{I-1}\{h_{I-1}\} + \cdots + 8\{h_3\} + 4\{h_2\}, \tag{3.6}$$

where p_0 and p_1 correspond to those partition levels where t^{p_0} and t^{p_1} equal the vectors $[1,1,\ldots,1]^T$ and $[2,2,\ldots,2]^T$, respectively. The terms g_0 and g_1 correspond to the modulo-2 sum of cosets with some (but not all) one's or two's in the coset. All the other cosets are added modulo-2 and are indicated by the terms h_2 to h_{I-1}.

After a 90° phase rotation and assuming $q = 0$ we have

$$\begin{bmatrix} y_{1,r} \\ \vdots \\ y_{L,r} \end{bmatrix} = \left[(2z^{p_1} + z^{p_0} + 1) \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} + 2\{g_1\} + \{g_0\} \pmod 4 \right] + 2^{I-1}\{h_{I-1}\} + \cdots + 8\{h_3\} + 4\{h_2\}, \tag{3.7}$$

where $y_{1,r}$ to $y_{L,r}$ are the rotated 2-D symbols. That is, $z_r^{p_0} = z^{p_0} \oplus 1$ and $z_r^{p_1} = z^{p_1} \oplus z^{p_0}$. All the other mapping bits are unaffected by the

phase rotation.

Equation (3.6) can best be illustrated by an example. This will show how we can use the mod-4 and mod-2 additions to form our multi-D signal set. We will use the 3×16QAM (I) signal set from Table 3.1. Thus, the mapping equation is (with q = 0),

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = z^{11}\begin{bmatrix} 0 \\ 8 \\ 8 \end{bmatrix} + z^{10}\begin{bmatrix} 8 \\ 8 \\ 0 \end{bmatrix} + z^{9}\begin{bmatrix} 8 \\ 8 \\ 8 \end{bmatrix} + z^{8}\begin{bmatrix} 0 \\ 4 \\ 4 \end{bmatrix} + z^{7}\begin{bmatrix} 4 \\ 4 \\ 0 \end{bmatrix} + z^{6}\begin{bmatrix} 4 \\ 4 \\ 4 \end{bmatrix} +$$

$$z^{5}\begin{bmatrix} 0 \\ 2 \\ 2 \end{bmatrix} + z^{4}\begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix} + z^{3}\begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix} + z^{2}\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} + z^{1}\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} + z^{0}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$= \left[(2z^3 + z^0)\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + 2\left\{z^5\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \oplus z^4\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}\right\} + \left\{z^2\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \oplus z^1\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}\right\} \pmod 4\right] +$$

$$8\left\{z^{11}\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \oplus z^{10}\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \oplus z^{9}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\right\} + 4\left\{z^{8}\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \oplus z^{7}\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \oplus z^{6}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\right\}$$

$$= \left[(2z^3 + z^0)\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + 2\begin{bmatrix} z^4 \\ z^4 \oplus z^5 \\ z^5 \end{bmatrix} + \begin{bmatrix} z^1 \\ z^1 \oplus z^2 \\ z^2 \end{bmatrix} \pmod 4\right] +$$

$$8\begin{bmatrix} z^9 \oplus z^{10} \\ z^9 \oplus z^{10} \oplus z^{11} \\ z^9 \oplus z^{11} \end{bmatrix} + 4\begin{bmatrix} z^6 \oplus z^7 \\ z^6 \oplus z^7 \oplus z^8 \\ z^6 \oplus z^8 \end{bmatrix}.$$

Figure 3.9 shows how this mapper may be implemented. The values of p_0 and p_1 for each type of multi-D signal set are also given in Table 3.1.

3.2.2 Encoder System

The encoder system is essentially the same as that given in Chapter 2. The only differences are in the construction of the multi-D signal set mapper (described in section 3.2.1) and the differential encoder (or precoder). Since only two bits are affected by a rotation,

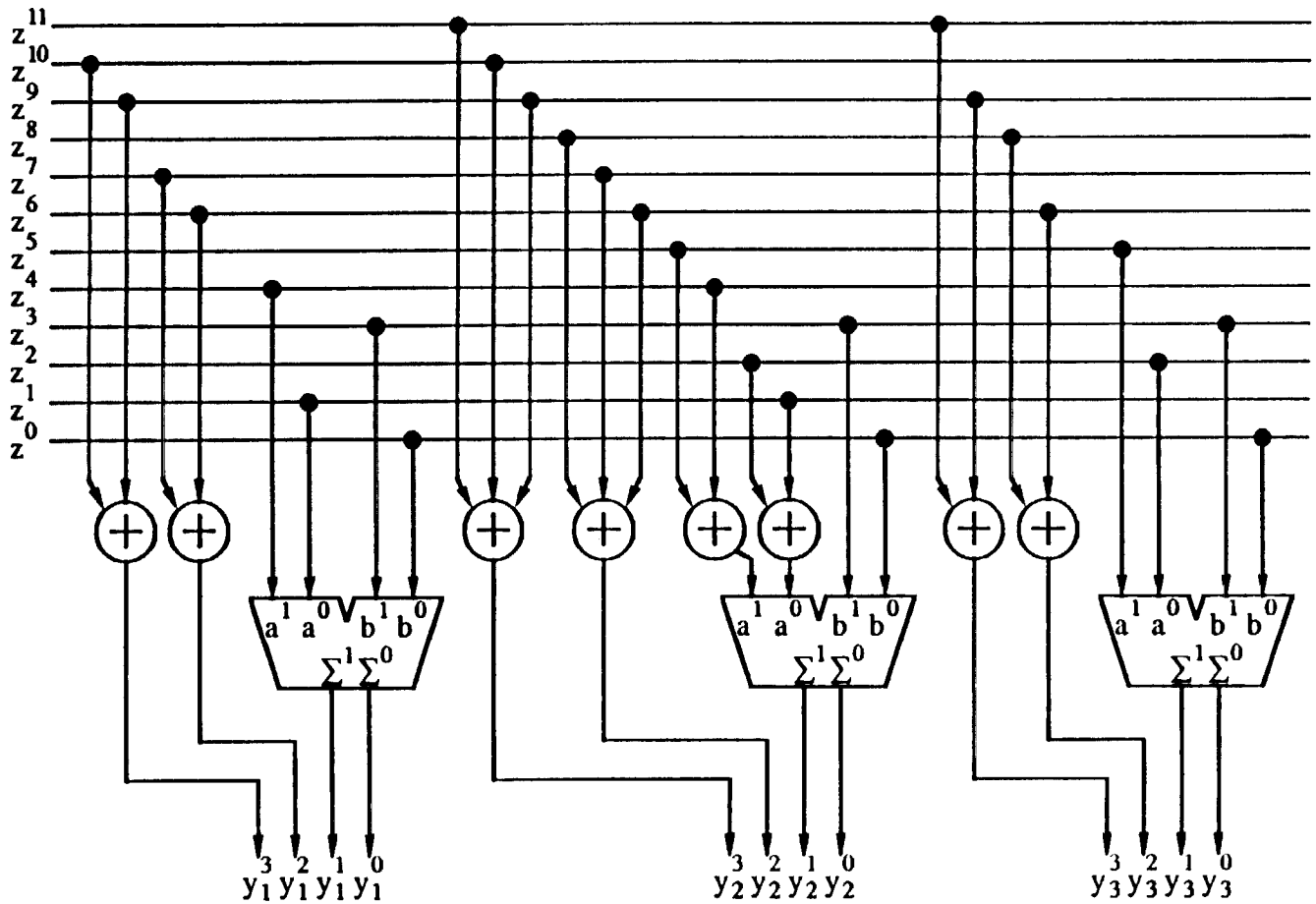

Figure 3.9: Signal set mapper for 3×16QAM (I) signal set.

the precoder and differential decoder (or postdecoder) equations are relatively easy to describe.

The encoders have rate (IL-q-1)/(IL-q) and can transmit I-(q+1)/L bit/sym. The value q indicates how many least significant bits (lsb) in the multi-D signal mapping are set to zero, as indicated in (3.5). A block diagram of the encoder system that is used in this chapter is given in Figure 2.8. We use the same notation, with $w^i(D)$ as the binary input sequences to the precoder, $x^i(D)$ as the binary input sequences to the systematic encoder, $z^i(D)$ as the binary output sequences of the encoder, and $y_j(D)$ as the integer output sequences of the signal set mapper.

In order to determine which precoder to use for a code, we need to determine the code transparency. Again, the technique used has been previously described in Chapter 2. We will summarize the important points here. Letting $b_0 = p_0 - q$ and $b_1 = p_1 - q$, the transparency $d = f + E[H^{b_f}(D)]$, where the largest value of $b_j \le \tilde{k}$ is b_f, $\tilde{k}$ is the number of checked encoder bits, $H^i(D)$, for $0 \le i \le \tilde{k}$, are the parity check polynomials of the code, and $E[H^i(D)]$ is the modulo-2 number of non-zero delay terms in $H^i(D)$. A code is then transparent to $2^d \times 90°$ phase rotations.

The mapping bits that are affected by a $2^d \times 90°$ phase rotation are z^{b_0} and z^{b_1} (with $z_r^{b_0} = z^{b_0} \oplus 1$ and $z_r^{b_1} = z^{b_1} \oplus z^{b_0}$) for $d = 0$, z^{b_1} (with $z_r^{b_1} = z^{b_1} \oplus 1$) for $d = 1$, and no bits for $d = 2$. For codes with $d = 2$, no precoder is required. When $d = 1$, no precoder is required if $b_1 = 0$. This is because $z^{b_1} = z^0$ is the encoded bit for the encoder. When $b_1 > 0$, the precoder equation is (from Chapter 2)

$$x^{b_l}(D) = Dx^{b_l}(D) \oplus w^{b_l}(D), \tag{3.8}$$

where $w^i(D)$, for $1 \le i \le IL\text{-}q$, are the inputs to the precoder and $x^i(D) = w^i(D)$ for all i not equal to b_l. The postdecoder equation is

$$\hat{w}^{b_l}(D) = (D \oplus 1)\hat{x}^{b_l}(D), \tag{3.9}$$

where $\hat{x}^{b_l}(D)$ is the estimated sequence for $x^{b_l}(D)$ from the decoder and $\hat{w}^{b_l}(D)$ is the resulting postdecoded sequence.

For $d = 0$ (full transparency), there are two types of precoders. From Chapter 2, we have for $b_0 > 0$ the precoding and postdecoding equations (letting $x(D) = 2x^{b_l}(D) + x^{b_0}(D)$ and $w(D) = 2w^{b_l}(D) + w^{b_0}(D)$)

$$x(D) = Dx(D) + w(D) \pmod 4, \tag{3.10}$$

and

$$\hat{w}(D) = (3D + 1)\hat{x}(D) \pmod 4. \tag{3.11}$$

The precoding and postdecoding equations for $b_0 = 0$ are (letting $x(D) = x^{b_l}(D)$ and $w(D) = w^{b_l}(D)$),

$$2x(D) = 2Dx(D) + 2w(D) + (D + 3)z^0(D) \pmod 4, \tag{3.12}$$

and

$$2\hat{w}(D) = (3D + 1)(2\hat{x}(D) + \hat{z}^0(D)) \pmod 4. \tag{3.13}$$

3.2.3 Code Search Results

The code search was basically the same as in Chapter 2. The main difference was in how the 2-D signal set is affected by a phase rotation. The code search program was modified so as to read in a subset and then rotate it by the required number of rotations to form

the 2-D signal set. For the QAM codes, these subsets are given in Figures 3.3 to 3.8, and rotations of 0°, 90°, 180°, and 270° were made to form the signal set. (An MPSK signal set is generated by rotating the subset point [0 1] M times in increments of 360/M degrees.) The construction of the multi-D signal sets and the different code transparencies were also incorporated into the program.

As in Chapter 2, we limited our code search to a code complexity of $\nu + \tilde{k} \leq 10$ (ν is the number of binary memory elements in the encoder). The systematic encoder is the same as is Chapter 2. The results of our code search are listed in Tables 3.2 to 3.17. Each table provides the following information:

The spectral efficiency of the codes K (in bit/sym).

The minimum squared distance of the uncoded signal set (d_u^2) used for comparison. This signal set is taken to be the smallest 2-D or multi-D signal set which has the same K as the codes. In most cases this signal set is the first subset of the first partition level.

The number of nearest neighbors for the uncoded signal set (N_u). When comparing N_u for signal sets with different dimensionalities, one should divide each N_u by the L of its signal set. This normalizes N_u to two dimensions.

The memory (ν), number of checked bits ($\tilde{k}$), parity check polynomials in octal notation (h^0 to $h^{\tilde{k}}$), phase invariance, minimum free squared Euclidean distance (d_{free}^2), number of nearest neighbors (N_{free}), and asympototic coding gain ($\gamma = \log_{10}(d_{free}^2/d_u^2)$ dB) of each code is given. For comparison purposes, N_{free} should be divided by L to normalize N_{free} to two dimensions.

When d_{free}^2 occurs along parallel transitions, the next nearest

squared Euclidean distance (d^2_{next}) and number of next nearest neighbors (N_{next}) are also given.

For comparison to 2-D signal sets with an integer value of K, the terms at the bottom of each table can be added to γ. This will give the adjusted asymptotic coding gain.

In Tables 3.2, 3.3(a), and 3.5(a) we have also given the number of nearest neighbors for infinite size constellations (N^{∞}_{free}), taken from [68].

TABLE 3.2

TRELLIS CODED 1×16QAM

K = 3.0 bit/sym, d^2_u = 2, N_u = 2.25 (1×8AMPM).

ν	$\tilde{k}$	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{free}	N^{∞}_{free}	γ (dB)
1	1	-	1	3	360°	3	3.375	-	-	-	1.76
2	1	-	2	5	360°	4	2.0	5	7.594	4	3.01
3	2	02	06	11	180°	5	3.781	-	-	-	3.98
	2	04	02	11	360°	5	3.656	-	-	16	3.98
4	2	16	12	27	180°	6	9.594	-	-	-	4.77
	2	16	04	23	360°	6	9.156	-	-	56	4.77
5	2	02	14	41	180°	6	1.891	-	-	16*	4.77
	2	36	02	55	360°	6	1.812	-	-	-	4.77
6	2	026	042	117	180°	7	6.172	-	-	-	5.44
	2	060	004	123	360°	7	4.828	-	-	56*	5.44
7	2	050	132	255	180°	8	19.238	-	-	344*	6.02
	2	164	026	253	360°	8	15.574	-	-	-	6.02
8	2	070	322	411	180°	8	2.387	-	-	44*	6.02

*Different code. γ_{8PSK} = 1.35 dB, γ_{8AMPM} = 0.0 dB.

As the number of points in each 2-D signal set increases, the maximum number of dimensions for which we give codes decreases, until

TABLE 3.3(a)

TRELLIS CODED 2×16QAM

K = 3.5 bit/sym, q=0, d_u^2 = 2, N_u = 13.5 (2×16QAM).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	N^{∞}_{free}	γ (dB)
1	1	-	-	-	1	3	180°	2	4.5	3	27.0	-	0.00
2	2	-	-	1	3	5	90°	3	20.25	-	-	-	1.76
3	2	-	-	04	02	11	180°	4	29.312	-	-	88	3.01
4	2	-	-	10	06	23	90°	4	9.062	5	121.5	24*	3.01
5	3	-	14	30	02	41	180°	4	4.0	5	60.75	-	3.01
	3	-	16	24	06	53	360°	4	4.0	5	53.156	8*	3.01
6	4	020	030	046	014	113	180°	5	31.641	-	-	-	3.98
	4	004	010	024	042	111	360°	5	31.328	-	-	144*	3.98

*Different code. γ_{8PSK} = 2.02 dB, γ_{8AMPM} = 0.67 dB.

TABLE 3.3(b)

TRELLIS CODED 2×16QAM

K = 3.0 bit/sym, q=1, d_u^2 = 2, N_u = 2.25 (1×8AMPM).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	1	3	90°	4	29.312	-	-	3.01
2	1	-	-	2	5	90°	4	9.062	6	91.125	3.01
3	2	-	04	02	11	360°	4	4.0	6	56.953	3.01
4	3	04	14	06	23	180°	6	32.453	-	-	4.77
	3	02	04	12	21	360°	6	31.844	-	-	4.77
5	3	06	14	22	43	180°	6	9.062	-	-	4.77
6	3	024	030	056	103	180°	8	114.605	-	-	6.02
7	3	034	044	106	203	180°	8	28.5	-	-	6.02
	3	044	070	106	203	360°	8	23.797	-	-	6.02

γ_{8PSK} = 1.35 dB, γ_{8AMPM} = 0 dB

TABLE 3.4(a)

TRELLIS CODED 3×16QAM

$K = 3.67$ bit/sym, $q=0$, $d_u^2 = 2$, $N_u = 33.75$ (3×16QAM I).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)	sig. set
1	1	-	-	1	3	90°	2	15.75	3	108.0	0.00	I
2	1	-	-	2	5	90°	2	6.75	3	27.0	0.00	II
	2	-	2	1	5	360°	2	4.5	-	-	0.00	II
3	2	-	04	02	11	90°	2	2.25	3	20.25	0.00	III
	2	-	04	02	11	360°	3	37.125	-	-	1.76	II
4	2	-	14	02	21	180°	3	20.25	-	-	1.76	II
3	3	01	02	06	11	360°	3	13.5	-	-	1.76	II
4	3	12	04	02	21	180°	4	102.188	-	-	3.01	I
5	3	24	14	02	41	180°	4	41.438	-	-	3.01	I
6	3	024	042	010	105	180°	4	21.188	5	121.5	3.01	I

$\gamma_{8PSK} = 2.23$ dB, $\gamma_{8AMPM} = 0.87$ dB

we only give codes for the 2-D signal set for 512STAR. As mentioned in Section 3.2.1, the reason for this is due to the large number of computations associated with determining the Euclidean weights of each multi-D signal point.

When N_{free} or N_{next} is taken into consideration in doing a code search, different codes can result from different signal sets. For example, the best $\nu = 8$ codes for 16QAM, 32CROSS, 64CIRC, 256CIRC, and 512STAR are all different. Therefore separate tables for each type of signal set are given. With multi-D signal sets, codes with full transparency appear due to the "spreading out" of the two bits that are affected by a phase rotation.

Figure 3.10 is a plot of the asymptotic coding gains of various codes with multi-D 16QAM signal sets compared to uncoded 8AMPM versus

TABLE 3.4(b)

TRELLIS CODED 3×16QAM

K = 3.33 bit/sym, q=1, d_u^2 = 2, N_u = 6.75 (3×16QAM II).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)	sig. set
1	1	-	-	-	1	3	90°	2	2.25	4	20.25	0.00	III
	1	-	-	-	1	3	360°	3	23.625	-	-	1.76	II
2	1	-	-	-	2	5	360°	3	13.5	5	68.344	1.76	II
	2	-	-	2	1	5	90°	3	6.75	4	20.25	1.76	III
	2	-	-	3	1	5	180°	4	112.313	-	-	3.01	I
	2	-	-	2	1	5	360°	4	87.0	-	-	3.01	II
3	2	-	-	02	06	11	180°	4	21.188	6	1139.062	3.01	II
	3	-	06	04	03	11	90°	4	11.062	-	-	3.01	III
4	3	-	14	04	12	23	90°	4	6.0	5	37.969	3.01	III
5	4	04	30	10	22	47	90°	5	21.875	-	-	3.98	III
6	4	022	006	066	010	133	90°	6	177.547	-	-	4.77	III
	4	014	052	024	056	115	180°	6	149.797	-	-	4.77	III

γ_{8PSK} = 1.81 dB, γ_{8AMPM} = 0.46 dB

decoder complexity ($\nu + \tilde{k} - \log_2 L$). Only the highest rate codes for each L are plotted, otherwise the figure would become too complicated. The highest gains are achieved for the 2-D codes (which also has the lowest K). Increasing L results in a decrease in coding gain since K increases. However, the 3.75 bit/sym codes appear to perform slightly better than the 3.67 bit/sym codes.

In Tables 3.2, 3.3(a), and 3.5(a), note that the N_{free}'s obtained for the finite signal sets are much less than for the infinite size signal sets. For the more complex codes, where the codes in [68] are different than the ones we found, very large differences between the N_{free}'s can again be seen. This illustrates the advantages obtained

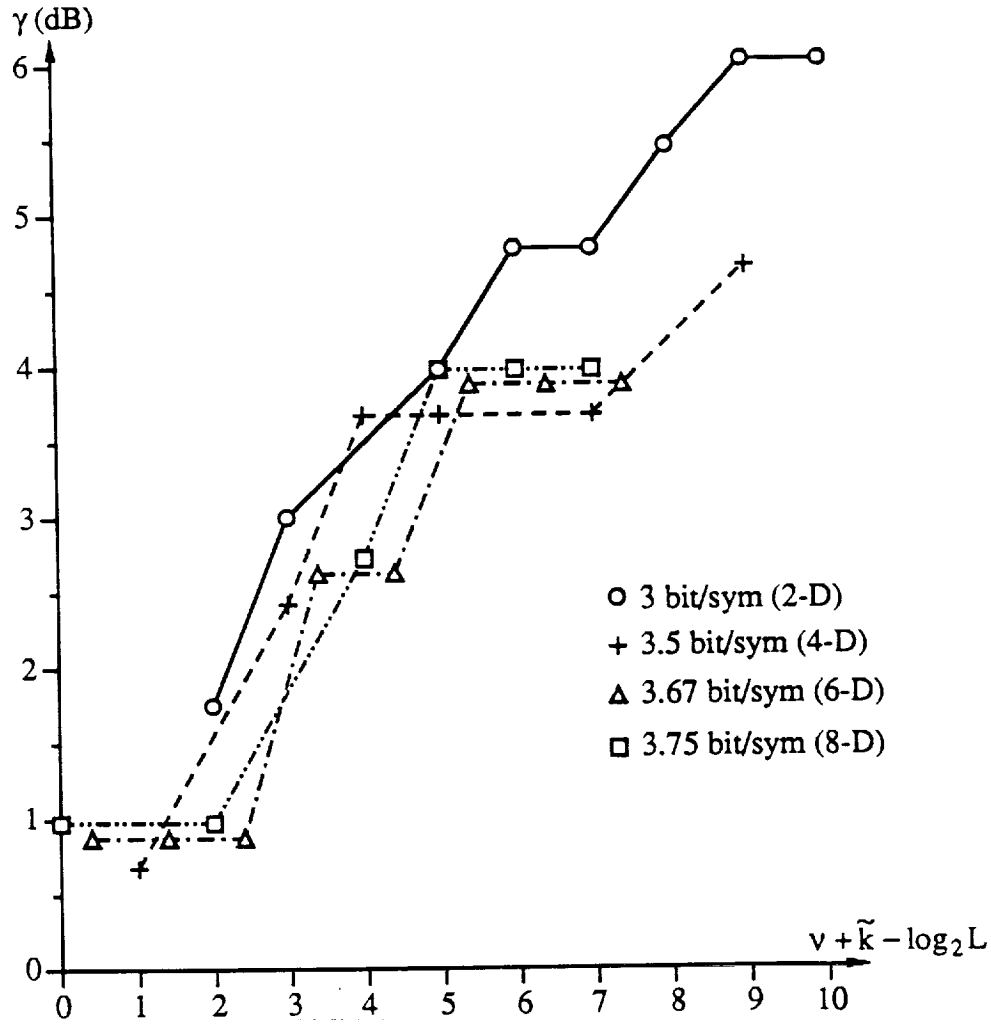

Figure 3.10: Plot of asymptotic coding gain (against uncoded 8AMPM) versus complexity for some trellis codes with multi-D 16QAM signal sets.

when a code search finds the codes with minimum N_{free}. As we increase the size of the signal set, the N_{free}'s asymptotically approach the values for the infinite size signal sets. A good example are the $\nu = 2$ codes for 2-D signal sets. We start with $N_{free} = 2$ for 16QAM and reach $N_{free} = 3.594$ for 512STAR, where $N^{\infty}_{free} = 4$.

An interesting code is the $\nu = 4$ code in Table 5(d). This code uses the 4×16QAM signal set and has K = 3 bit/sym. This sixteen state code has γ equal to 6.02 dB and is 90° invariant. Although the code has

TABLE 3.4(c)

TRELLIS CODED 3×16QAM

K = 3.00 bit/sym, q=2, d_u^2 = 2, N_u = 2.25 (1×8AMPM).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)	sig. set
0	0	-	-	-	-	-	90°	3	13.5	-	-	1.76	II
1	1	-	-	-	1	3	180°	4	21.188	6	182.25	3.01	II
2	2	-	-	3	2	5	90°	4	6.0	5	30.375	3.01	III
3	3	-	04	03	02	11	90°	5	14.875	-	-	3.98	III
4	3	-	04	06	12	27	90°	6	31.015	-	-	4.77	III
5	3	-	30	14	16	43	90°	6	5.703	7	44.0	4.77	III
	4	20	10	04	02	41	180°	7	34.43	-	-	5.44	II
6	4	024	010	004	042	101	180°	8	77.805	-	-	6.02	II
	4	044	050	024	002	103	360°	8	40.316	-	-	6.02	II

γ_{8PSK} = 1.35 dB, γ_{8AMPM} = 0 dB

TABLE 3.5(a)

TRELLIS CODED 4×16QAM

K = 3.75 bit/sym, q=0, d_u^2 = 2, N_u = 63 (4×16QAM).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	N^{∞}_{free}	γ (dB)
1	1	-	-	1	3	90°	2	27.0	3	216.0	-	0.00
2	2	-	2	1	5	90°	2	9.0	3	162.0	-	0.00
3	3	04	02	01	11	90°	3	94.5	-	-	-	1.76
4	3	10	04	02	21	90°	4	402.875	-	-	-	3.01
5	3	24	14	02	41	90°	4	159.875	-	-	496*	3.01
6	3	050	032	004	103	90°	4	78.875	5	1215.0	240*	3.01

*Different code, γ_{8PSK} = 2.32 dB, γ_{8AMPM} = 0.97 dB

TABLE 3.5(b)

TRELLIS CODED 4×16QAM

K = 3.50 bit/sym, q=1, d_u^2 = 2, N_u = 27 (4×16QAM).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	-	1	3	90°	2	9.0	4	324.0	0.00
2	2	-	-	2	1	5	90°	4	402.875	-	-	3.01
3	2	-	-	04	02	11	90°	4	159.875	-	-	3.01
4	2	-	-	12	04	23	90°	4	78.875	6	3645.0	3.01
5	3	-	14	34	06	41	90°	4	38.375	6	2004.75	3.01
	3	-	04	14	22	43	180°	4	38.375	6	1822.5	3.01
6	4	014	006	056	022	103	90°	4	18.125	6	637.875	3.01

γ_{8PSK} = 2.02 dB, γ_{8AMPM} = 0.67 dB

TABLE 3.5(c)

TRELLIS CODED 4×16QAM

K = 3.25 bit/sym, q=2, d_u^2 = 2, N_u = 9.0 (4×16QAM).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	-	1	3	90°	4	159.875	-	-	3.01
2	1	-	-	-	2	5	90°	4	78.875	6	729.0	3.01
3	2	-	-	06	02	11	90°	4	38.375	6	729.0	3.01
	2	-	-	02	06	11	180°	4	38.375	6	364.5	3.01
4	3	-	10	14	06	21	90°	4	18.125	6	91.125	3.01
5	4	10	04	06	22	41	90°	4	8.0	6	91.125	3.01

γ_{8PSK} = 1.70 dB, γ_{8AMPM} = 0.35 dB

TABLE 3.5(d)

TRELLIS CODED 4×16QAM

K = 3.00 bit/sym, q=3, d_u^2 = 2, N_u = 2.25 (1×8AMPM).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
0	0	-	-	-	-	-	90°	4	78.875	-	-	3.01
1	1	-	-	-	1	3	180°	4	38.375	8	1640.25	3.01
2	2	-	-	2	3	5	90°	4	18.125	8	1640.25	3.01
3	3	-	02	04	03	11	90°	4	8.0	8	1127.672	3.01
4	4	12	10	04	03	21	90°	8	734.628	-	-	6.02
5	4	26	22	10	06	41	90°	8	320.832	-	-	6.02
6	4	056	042	020	006	101	90°	8	142.066	-	-	6.02

γ_{8PSK} = 1.35 dB, γ_{8AMPM} = 0 dB

TABLE 3.6

TRELLIS CODED 1×32CROSS

K = 4.0 bit/sym, d_u^2 = 2, N_u = 2.875 (1×16CROSS).

ν	$\tilde{k}$	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{free}	γ (dB)
1	1	-	1	3	360°	3	4.672	-	-	1.76
2	1	-	2	5	360°	4	2.5	5	13.432	3.01
3	2	02	06	13	180°	5	6.715	-	-	3.98
	2	04	02	11	360°	5	6.693	-	-	3.98
4	2	16	12	27	180°	6	19.427	-	-	4.77
	2	16	04	23	360°	6	19.109	-	-	4.77
5	2	16	22	45	180°	6	3.75	-	-	4.77
	2	34	16	45	360°	6	3.438	-	-	4.77
6	2	026	042	117	180°	7	13.003	-	-	5.44
	2	036	064	115	360°	7	11.118	-	-	5.44
7	2	050	162	211	180°	8	45.124	-	-	6.02
	2	056	150	223	360°	8	38.338	-	-	6.02
8	2	070	226	431	180°	8	5.154	-	-	6.02
	2	272	304	455	360°	8	4.900	-	-	6.02

$\gamma_{16CROSS}$ = 0.0 dB.

TABLE 3.7(a)

TRELLIS CODED 2×32CROSS

K = 4.5 bit/sym, q=0, d_u^2 = 2, N_u = 16.312 (2×32CROSS).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	-	1	3	180°	2	5.75	3	34.328	0.00
2	2	-	-	1	3	5	90°	3	26.508	-	-	1.76
3	2	-	-	06	02	11	180°	4	41.266	-	-	3.01
4	2	-	-	10	06	23	90°	4	13.266	5	189.341	3.01
5	3	-	14	30	02	41	180°	4	5.0	5	94.671	3.01
	3	-	16	24	06	57	360°	4	5.0	5	83.34	3.01
6	4	004	014	020	046	113	180°	5	50.659	-	-	3.98
	4	004	010	024	042	111	360°	5	50.620	-	-	3.98

$\gamma_{16CROSS}$ = 0.51 dB.

TABLE 3.7(b)

TRELLIS CODED 2×32CROSS

K = 4.0 bit/sym, q=1, d_u^2 = 2.0, N_u = 2.875 (1×16CROSS).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	1	3	90°	4	43.633	-	-	3.01
2	1	-	-	2	5	90°	4	13.266	6	174.611	3.01
3	2	-	04	02	11	360°	4	5.0	6	111.069	3.01
4	3	04	14	02	21	180°	6	62.411	-	-	4.77
	3	10	04	02	21	360°	6	61.893	-	-	4.77
5	3	24	14	06	43	180°	6	18.240	-	-	4.77
6	3	024	014	042	103	180°	8	257.357	-	-	6.02
7	3	034	044	106	233	180°	8	59.178	-	-	6.02
	3	044	070	106	203	360°	8	55.477	-	-	6.02

$\gamma_{16CROSS}$ = 0 dB

TABLE 3.8(a)

TRELLIS CODED 3×32CROSS

$K = 4.67$ bit/sym, $q=0$, $d_u^2 = 2$, $N_u = 40.312$ (3×32CROSS I).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)	sig. set
1	1	-	-	1	3	90°	2	19.188	3	137.312	0.00	I
2	1	-	-	2	5	90°	2	8.625	3	34.328	0.00	II
	2	-	3	1	5	180°	2	7.922	-	-	0.00	II
	2	-	2	1	5	360°	2	4.5	-	-	0.00	II
3	2	-	04	02	11	90°	2	2.875	3	26.508	0.00	III
	2	-	04	02	11	360°	3	48.344	-	-	1.76	II
4	2	-	14	02	21	180°	3	25.746	-	-	1.76	II
3	3	01	02	06	11	360°	3	17.164	-	-	1.76	II
4	3	12	04	02	21	180°	4	143.566	-	-	3.01	I
5	3	24	14	02	41	180°	4	60.189	-	-	3.01	I
6	3	024	042	010	105	180°	4	32.297	5	181.295	3.01	I

$\gamma_{16CROSS} = 0.67$ dB

$\tilde{k} = 4$ (sixteen paths into every state) and a large N_{free} (183.657 when normalized to two dimensions) this code may be useful at high SNR where its large free distance will be very important. Its decoder complexity is six, the same as the 16 state code with a 2-D signal set, which has 1.25 dB less asymptotic coding gain and is 180° invariant.

For the larger size signal sets not given, the same codes found for the corresponding smaller size signal sets can be used. For small complexity, the codes are very likely to be optimum (for the criteria used). However, the larger the complexity, the more likely it is that the existing codes are not optimum. However, these codes are still likely to be better than a code with maximum d^2_{free} chosen at random without regard to N_{free}.

TABLE 3.8(b)

TRELLIS CODED 3×32CROSS

K = 4.33 bit/sym, q=1, d_u^2 = 2, N_u = 8.625 (3×32CROSS II).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)	sig. set
1	1	-	-	-	1	3	90°	2	2.875	4	30.367	0.00	III
	1	-	-	-	1	3	360°	3	31.180	-	-	1.76	II
2	1	-	-	-	2	5	360°	3	17.164	5	120.885	1.76	II
	2	-	-	2	1	5	90°	3	8.582	4	31.041	1.76	III
	2	-	-	3	1	5	180°	4	161.523	-	-	3.01	I
	2	-	-	2	1	5	360°	4	128.515	-	-	3.01	II
3	2	-	-	02	06	11	180°	4	32.297	6	2042.616	3.01	II
	3	-	06	04	03	11	90°	4	15.092	-	-	3.01	III
4	3	-	14	04	12	23	90°	4	7.5	5	62.778	3.01	III
5	4	10	12	32	04	41	90°	5	37.935	-	-	3.98	III
6	4	022	006	044	010	133	90°	6	321.508	-	-	4.77	III
	4	014	052	030	056	115	180°	6	273.836	-	-	4.77	III

$\gamma_{16CROSS}$ = 0.35 dB

Changing the signal set mapping may also result in better codes. For example, there is another 128 point signal set with the same energy as 128CROSS. A code search with this signal set resulted in some codes with a lower N_{free} than with 128CROSS. There are other multi-D signal sets that have 2^I points in each 2-D signal set where I is not an integer [10,12,24,74]. The codes presented here can be used with these signal sets as well. These signal sets are designed to map 2^{IL+1} points into 2L dimensions with minimum energy. The codes usually have d^2_{free} = 4, which gives a fundamental coding gain of 3 dB. Extra gain is achieved through the smaller size (and smaller energy) of the signal sets. Cosets can be partially used in the construction of these signal

TABLE 3.8(c)

TRELLIS CODED 3×32CROSS

K = 4.00 bit/sym, q=2, d_u^2 = 2, N_u = 2.875 (1×16CROSS).

v	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)	sig. set
0	0	-	-	-	-	-	90°	3	17.164	-	-	1.76	II
1	1	-	-	-	1	3	180°	4	32.297	6	294.605	3.01	II
2	2	-	-	3	2	5	90°	4	7.5	5	49.347	3.01	III
3	3	-	04	05	02	11	90°	5	24.668	-	-	3.98	III
4	3	-	04	06	12	23	90°	6	54.379	-	-	4.77	III
5	3	-	30	24	16	41	90°	6	11.881	7	84.531	4.77	III
	4	02	10	04	22	41	180°	7	69.262	-	-	5.44	II
6	4	034	024	014	042	101	180°	8	171.597	-	-	6.02	II
	4	044	050	024	002	103	360°	8	92.192	-	-	6.02	II

$\gamma_{16CROSS}$ = 0 dB

TABLE 3.9

TRELLIS CODED 4×32CROSS

K = 4.00 bit/sym, q=3, d_u^2 = 2, N_u = 2.875 (1×16CROSS).

v	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
0	0	-	-	-	-	-	90°	4	115.377	-	-	3.01
1	1	-	-	-	1	3	180°	4	59.594	8	3111.766	3.01
2	2	-	-	2	3	5	90°	4	26.531	8	3400.215	3.01
3	3	-	02	04	03	11	90°	4	10.0	8	2434.473	3.01
4	4	16	12	06	03	21	90°	8	1571.475	-	-	6.02
5	4	26	22	14	06	41	90°	8	686.798	-	-	6.02
6	4	056	042	020	006	101	90°	8	290.952	-	-	6.02

$\gamma_{16CROSS}$ = 0 dB

TABLE 3.10

TRELLIS CODED 1×64CIRC

$K = 5.0$ bit/sym, $d_u^2 = 2$, $N_u = 3.188$ (1×32CIRC).

ν	$\tilde{k}$	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{free}	γ (dB)
1	1	-	1	3	360°	3	5.379	-	-	1.76
2	1	-	2	5	360°	4	2.875	5	17.145	3.01
3	2	02	06	11	180°	5	8.573	-	-	3.98
	2	02	04	13	360°	5	8.566	-	-	3.98
4	2	16	12	23	180°	6	25.815	-	-	4.77
	2	16	04	23	360°	6	25.637	-	-	4.77
5	2	16	22	45	180°	6	4.672	-	-	4.77
	2	34	16	45	360°	6	4.492	-	-	4.77
6	2	064	042	115	180°	7	17.668	-	-	5.44
	2	036	064	123	360°	7	15.634	-	-	5.44
7	2	024	116	205	180°	8	67.858	-	-	6.02
	2	164	026	253	360°	8	59.375	-	-	6.02
8	2	124	204	413	180°	8	10.147	-	-	6.02
	2	272	304	523	360°	8	9.760	-	-	6.02

$\gamma_{32CIRC} = 0.0$ dB.

sets also.

In implementing the codes given in the tables one may wish to change various aspects of a code in order to simplify the encoder and Viterbi decoder. One desirable aspect is the conversion of the systematic (feedback) convolutional encoder to non-systematic (feedforward) form. An algorithm for this conversion is given in [57].

The cosets may also be changed. Since all the codes presented have parallel transitions, some of the cosets for the parallel transitions can be simplified since they do not play any part in

TABLE 3.11(a)

TRELLIS CODED 2×64CIRC

K = 5.5 bit/sym, q=0, d_u^2 = 2, N_u = 17.766 (2×64CIRC).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	-	1	3	180°	2	6.375	3	38.443	0.00
2	2	-	-	1	3	5	90°	3	29.979	-	-	1.76
3	2	-	-	06	02	11	180°	4	48.347	-	-	3.01
4	2	-	-	10	06	23	90°	4	15.910	5	232.012	3.01
5	3	-	14	30	02	41	180°	4	5.75	5	116.006	3.01
	3	-	16	24	06	57	360°	4	5.75	5	102.322	3.01
6	4	004	014	020	046	111	180°	5	62.280	-	-	3.98
	4	004	010	024	042	111	360°	5	62.268	-	-	3.98

γ_{32CIRC} = 0.41 dB.

TABLE 3.11(b)

TRELLIS CODED 2×64CIRC

K = 5.0 bit/sym, q=1, d_u^2 = 2.0, N_u = 3.188 (1×32CIRC).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	1	3	90°	4	52.218	-	-	3.01
2	1	-	-	2	5	90°	4	15.910	6	231.461	3.01
3	2	-	04	02	11	360°	4	5.75	6	148.116	3.01
4	3	04	14	02	21	180°	6	83.043	-	-	4.77
	3	10	04	02	21	360°	6	82.725	-	-	4.77
5	3	24	14	06	43	180°	6	24.860	-	-	4.77
6	3	024	014	042	103	180°	8	368.382	-	-	6.02
7	3	034	044	106	233	180°	8	83.408	-	-	6.02
	3	044	070	106	203	360°	8	79.962	-	-	6.02

γ_{32CIRC} = 0 dB

TABLE 3.12

TRELLIS CODED 3×64CIRC

K = 5.00 bit/sym, q=2, d_u^2 = 2, N_u = 3.188 (1×32CIRC).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)	sig. set
0	0	-	-	-	-	-	90°	3	19.222	-	-	1.76	II
1	1	-	-	-	1	3	180°	4	39.105	6	369.473	3.01	II
2	2	-	-	3	2	5	90°	4	8.625	5	61.269	3.01	III
3	3	-	04	03	02	11	90°	5	30.634	-	-	3.98	III
4	3	-	04	06	12	23	90°	6	70.312	-	-	4.77	III
5	3	-	30	14	16	41	90°	6	16.193	7	114.183	4.77	III
	4	20	10	04	02	41	180°	7	93.954	-	-	5.44	II
6	4	034	024	014	042	101	180°	8	245.520	-	-	6.02	II
	4	044	050	024	002	103	360°	8	197.143	-	-	6.02	II

γ_{32CIRC} = 0 dB

TABLE 3.13

TRELLIS CODED 1×128CROSS

K = 6.0 bit/sym, d_u^2 = 2, N_u = 3.344 (1×64CROSS).

ν	$\tilde{k}$	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{free}	γ (dB)
1	1	-	1	3	360°	3	6.061	-	-	1.76
2	1	-	2	5	360°	4	3.250	5	20.265	3.01
3	2	02	06	11	180°	5	10.132	-	-	3.98
	2	04	02	11	360°	5	10.103	-	-	3.98
4	2	16	12	27	180°	6	32.033	-	-	4.77
5	2	16	22	45	180°	6	5.281	-	-	4.77
6	2	064	042	115	180°	7	23.434	-	-	5.44
	2	036	064	123	360°	7	20.429	-	-	5.44
7	2	024	132	205	180°	8	96.202	-	-	6.02
	2	164	026	253	360°	8	79.119	-	-	6.02
8	2	070	322	411	180°	8	13.646	-	-	6.02
	2	124	320	413	360°	8	13.289	-	-	6.02

$\gamma_{64CROSS}$ = 0.0 dB.

TABLE 3.14(a)

TRELLIS CODED 2×128CROSS

K = 6.5 bit/sym, q=0, d_u^2 = 2, N_u = 19.828 (2×128CROSS).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	-	1	3	180°	2	6.688	3	47.635	0.00
2	2	-	-	1	3	5	90°	3	35.938	-	-	1.76
3	2	-	-	06	02	11	180°	4	60.850	-	-	3.01
4	2	-	-	10	06	23	90°	4	17.681	5	315.766	3.01
5	3	-	14	30	02	41	180°	4	6.5	5	157.883	3.01
	3	-	16	24	06	57	360°	4	6.5	5	138.322	3.01
6	4	020	030	062	004	115	180°	5	83.886	-	-	3.98
	4	004	010	024	042	111	360°	5	83.792	-	-	3.98

$\gamma_{64CROSS}$ = 0.35 dB.

TABLE 3.14(b)

TRELLIS CODED 2×128CROSS

K = 6.0 bit/sym, q=1, d_u^2 = 2, N_u = 3.344 (1×64CROSS).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	1	3	90°	4	61.620	-	-	3.01
2	1	-	-	2	5	90°	4	17.681	6	293.842	3.01
3	2	-	04	02	11	360°	4	6.5	6	184.306	3.01
4	3	04	14	06	23	180°	6	102.715	-	-	4.77
	3	02	04	12	21	360°	6	102.192	-	-	4.77
5	3	06	14	22	43	180°	6	28.731	-	-	4.77
6	3	024	030	056	103	180°	8	530.186	-	-	6.02
7	3	034	044	106	203	180°	8	112.917	-	-	6.02
	3	044	070	106	203	360°	8	107.250	-	-	6.02

$\gamma_{64CROSS}$ = 0 dB

TABLE 3.15

TRELLIS CODED 1×256CIRC

$K = 7.0$ bit/sym, $d_u^2 = 2$, $N_u = 3.609$ (1×128CIRC).

ν	$\tilde{k}$	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{free}	γ (dB)
1	1	-	1	3	360°	3	6.711	-	-	1.76
2	1	-	2	5	360°	4	3.438	5	24.223	3.01
3	2	02	06	11	180°	5	12.112	-	-	3.98
	2	04	02	11	360°	5	12.110	-	-	3.98
4	2	16	12	27	180°	6	39.681	-	-	4.77
	2	16	04	23	360°	6	39.611	-	-	4.77
5	2	16	22	45	180°	6	6.230	-	-	4.77
	2	34	16	45	360°	6	6.177	-	-	4.77
6	2	026	042	117	180°	7	29.313	-	-	5.44
	2	036	064	115	360°	7	26.276	-	-	5.44
7	2	050	162	211	180°	8	125.480	-	-	6.02
	2	056	150	223	360°	8	106.540	-	-	6.02
8	2	124	204	537	180°	8	17.755	-	-	6.02
	2	272	304	455	360°	8	17.514	-	-	6.02

$\gamma_{128CIRC} = 0.0$ dB.

determining the free distances of the codes given. For example, in the 3×16QAM signal sets, we could change t^9, t^{10}, and t^{11} to $[0\ 0\ 8]^T$, $[0\ 8\ 0]^T$, and $[8\ 0\ 0]^T$. This simplifies both the signal set mapper and the branch metric calculator within a Viterbi decoder without having any affect on d^2_{free} (since most codes have $d^2_{free} \leq 8$). Only very complex codes with $d^2_{free} > 8$ would require the full set of cosets that are given. If the cosets are changed (thereby changing the signal set mapping) one should perform another code search in order to find the codes with the smallest N_{free}.

For codes that are either 180° or 360° invariant it is not

TABLE 3.16(a)

TRELLIS CODED 2×256CIRC

K = 7.5 bit/sym, q=0, d_u^2 = 2, N_u = 21.048 (2×256CIRC).

v	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	-	1	3	180°	2	7.219	3	51.427	0.00
2	2	-	-	1	3	5	90°	3	39.136	-	-	1.76
3	2	-	-	06	02	11	180°	4	67.714	-	-	3.01
4	2	-	-	10	06	23	90°	4	19.903	5	363.416	3.01
5	3	-	14	30	02	41	180°	4	6.875	5	181.708	3.01
	3	-	16	24	06	57	360°	4	6.875	5	159.484	3.01
6	4	004	014	020	046	113	180°	5	96.908	-	-	3.98
	4	004	010	024	042	111	360°	5	96.895	-	-	3.98

$\gamma_{128CIRC}$ = 0.30 dB.

TABLE 3.16(b)

TRELLIS CODED 2×256CIRC

K = 7.0 bit/sym, q=1, d_u^2 = 2.0, N_u = 3.609 (1×128CIRC).

v	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	1	3	90°	4	69.817	-	-	3.01
2	1	-	-	2	5	90°	4	19.903	6	360.320	3.01
3	2	-	04	02	11	360°	4	6.875	6	227.181	3.01
4	3	04	14	02	21	180°	6	125.944	-	-	4.77
	3	10	04	02	21	360°	6	125.842	-	-	4.77
5	3	24	14	06	43	180°	6	35.762	-	-	4.77
6	3	024	014	042	103	180°	8	683.545	-	-	6.02
7	3	034	044	106	203	180°	8	140.894	-	-	6.02
	3	044	070	106	203	360°	8	138.492	-	-	6.02

$\gamma_{128CIRC}$ = 0 dB

TABLE 3.17

TRELLIS CODED 1×512STAR

$K = 8.0$ bit/sym, $d_u^2 = 2$, $N_u = 3.711$ (1×256STAR).

ν	$\tilde{k}$	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{free}	γ (dB)
1	1	-	1	3	360°	3	7.045	-	-	1.76
2	1	-	2	5	360°	4	3.594	5	26.143	3.01
3	2	02	06	11	180°	5	13.072	-	-	3.98
	2	04	02	11	360°	5	13.071	-	-	3.98
4	2	16	12	27	180°	6	43.571	-	-	4.77
5	2	16	22	45	180°	6	6.626	-	-	4.77
6	2	064	042	115	180°	7	32.749	-	-	5.44
	2	036	052	115	360°	7	29.821	-	-	5.44
7	2	024	132	205	180°	8	143.926	-	-	6.02
	2	056	106	275	360°	8	123.283	-	-	6.02
8	2	130	306	513	180°	8	20.078	-	-	6.02

$\gamma_{256STAR} = 0.0$ dB.

necessary to add any cosets modulo-4. Instead, all the cosets can be added modulo-2, simplifying the signal set mapper. Again, to find the best codes, one should repeat the code search with the new signal set.

3.3 Conclusions

A method has been described for obtaining 2-D constellations suitable for trellis coding and for use in constructing multi-dimensional signal sets. A variety of codes, some of which are fully invariant to 90° phase rotations, have been found using a systematic code search.

These codes can be used with other multi-D signal sets. However,

they may not be optimum in terms of the minimum number of nearest neighbors. The small size signal sets (especially 16QAM) may be useful in satellite communication systems where high bandwidth efficiency is required at the expense of more linear amplifiers. The large size signal sets (up to 512 points) should be useful for high capacity microwave links and telephone modems where high data rates are required in limited bandwidth channels.

CHAPTER FOUR

IMPLEMENTATION OF A VITERBI DECODER

As a demonstration of the performance capabilities of trellis codes using multidimensional signal sets, a Viterbi decoder for one of the codes given in Chapter 2 was designed. The choice of code was based on two factors.

The first factor was its application as a possible replacement for the coding scheme currently used on the Hubble Space Telescope (HST). The HST at present uses the rate 1/3 $\nu = 6$ convolutional code with BPSK modulation. With the modulator restricted to 3 Msym/s, this implies a data rate of only 1 Mbit/s, since K = 1/3 bit/sym. This is a very bandwidth inefficient scheme, although the system has the advantage of simplicity and large coding gain.

The basic requirement from NASA was for a scheme that has as large a K as possible. Since a satellite channel was being used, 8PSK modulation was selected. This allows a K of between 2 and 3 bit/sym. The next influencing factor was the 2.33 bit/sym Periodically Time Varying Trellis Code (PTVTC) that was implemented by COMSAT [30]. This 16 state code was designed for 140 Mbit/s cable restoration service over the 72 MHz transponders onboard INTELSAT satellites.

As mentioned in Chapter 2, the equivalent 16 state 6D-8PSK trellis code has many advantages over this PTVTC. For this reason the rate 7/8, 2.33 bit/sym, 6D-8PSK, 90° invariant trellis code was chosen.

A direct comparison can then be made between the two codes, illustrating the advantages that the multi-D signal set provides.

4.1 Encoder Implementation

At first, the systematic encoder in Figure 2.11 was used in the design. However, it was found that in designing a Viterbi decoder, it would be simpler if a non-systematic convolutional encoder was used. This is because the state transitions in a non-systematic encoder are highly structured, compared with the almost "random" transitions of a systematic encoder.

To convert the systematic encoder to a non-systematic form, the technique described in [57] is used. This method uses the fact that the impulse response of each shift register in a non-systematic encoder will produce output sequences that are equivalent to the generator polynomials. Since a systematic encoder must also produce the same sequences, it is relatively easy to find $\tilde{k}$ linearly independent output sequences from a systematic encoder that can be used as generators of a non-systematic encoder.

There is usually more than one set of possible generator polynomials. The polynomials are chosen so that the inputs $x^1(D)$ and $x^2(D)$ are affected by a 90° phase rotation in the same way as in a systematic encoder. Thus, the differential encoder for the systematic code can also be used for the non-systematic encoder. The non-systematic encoder equations that were found for the 6D-8PSK code are

$$z^2(D) = x^2(D) \oplus (D^2 \oplus D)x^1(D), \tag{4.1a}$$
$$z^1(D) = (D^2 \oplus D)x^2(D) \oplus (D \oplus 1)x^1(D), \tag{4.1b}$$
$$z^0(D) = Dx^1(D). \tag{4.1c}$$

Figure 4.1 illustrates the new non-systematic encoder. After a 90° phase rotation, we have $z_r^2(D) = z^2(D) \oplus 1(D)$, $z_r^1(D) = z^1(D)$, and $z_r^0(D) = z^0(D)$. Rotating the equations in (4.1) gives $x_r^2(D) = x^2(D) \oplus 1(D)$ and $x_r^1(D) = x^1(D)$, the same as for the systematic encoder.

The encoder uses a Phase Locked Loop (PLL) to generate the three times clock for transmitting the three 2D symbols. This PLL is based on the 74HC4046 Integrated Circuit (IC). The encoder is able to accept data either serially or in seven bit bytes.

4.2 Decoder Implementation

Due to the complexity of the decoder design, only a brief description is given here. As such, only the important design decisions are described.

To reduce the cost of the codec, a serial implementation of the decoder was chosen. That is, one clock cycle would be required for each state of the code. Since there are 16 states, at least 16 clock cycles are required to process each received 6D point. As will be described in more detail later, an extra seven clock cycles are required for start-up purposes. Thus, a total of 23 clock cycles are required for each iteration of the Viterbi algorithm.

The technology and clock speed in our design is the same as used in another Viterbi decoder designed by the author [47,48]. This gave us greater confidence that the design would work, even though the actual

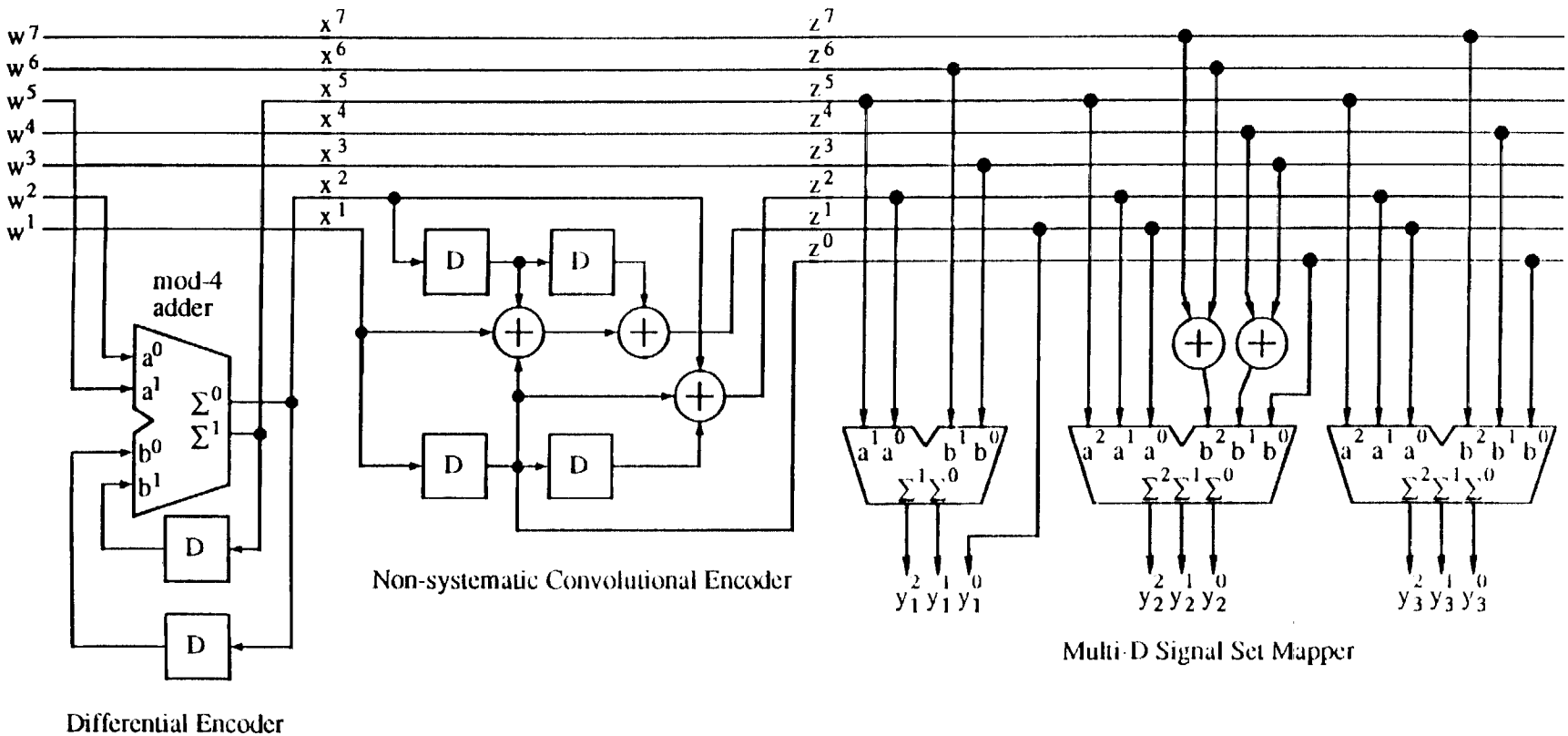

Figure 4.1: Non-systematic encoder block diagram for the16 state 2.33 bit/sym 6D-8PSK trellis code.

design is twice as complicated. Our design uses a 10 MHz clock (giving 100 ns clock cycles) and Schottky TTL logic for its ease of use and large variety of functions. The actual technologies used are 74LS (Low-power Schottky TTL) for non-time critical sections of the circuit and 74F (Advanced Schottky TTL) for time critical sections. Other technologies are used for functions not available in 74F or 74LS.

The decoder is operated asynchronously to the received data clock. This requires one of the seven extra clock cycles described above. Internally, the decoder operates synchronously to the 10 MHz clock. The decoder starts operation after detecting the first rising edge of the received 6D symbol clock. After 23 clock cycles, the decoder stops and waits for the next rising edge of the 6D symbol clock. This allows the decoder to operate at any data rate from 0 to 3 Mbit/s.

Each iteration of the Viterbi algorithm decodes seven bits for each received 6D signal point (since the code rate is 7/8). The maximum 6D symbol rate of the decoder is the internal clock speed divided the number of clock cycles required to decode the seven bits, i.e., 4.35×10^5 6D symbols per second. Therefore, the maximum bit rate of the decoder is 3.04 Mbit/s. For the HST, this code could achieve a data rate up to 7 Mbit/s. For actual use on the HST, it is intended that the decoder would be implemented on a VLSI chip, where the required decoding speed would be achieved.

There are six main sections in the Viterbi decoder. These are

- Branch Metric Calculator (BMC)
- State Metric Calculator (SMC)

- Survivor Sequence Memory (SSM)
- Signal Set Synchronizor (SSS)
- Minimum State Metric Selector (MSMS)
- Branch Point Selector (BPS)

Figure 4.2 illustrates a block diagram of the decoder. The above sections are described as follows.

4.2.1 Branch Metric Calculator

For each transition of the trellis there are 32 parallel paths (due to the five unchecked bits in the encoder). The BMC must determine which of the paths is closest to the received 6D signal point (the Branch Point (BP)) as well as the Branch Metric (BM) for this path. The BM can be calculated in a number of ways. The optimum BM's for AWGN channels with quantization are log-likelihood metrics [48]. Alternatively, one could make an approximation based on the squared Euclidean distance between the received point and the points along the transitions.

In our design we have chosen to use Read Only Memory's (ROM's) to store the precalculated BP (five bits are used to represent each parallel path) and BM (based on log-likelihood metrics). The encoder can produce one of eight (i.e., $2^{\bar{k}+1}$) sets of parallel paths (each containing 32 paths). The BP and BM must be determined for each of these eight sets of parallel paths.

We have chosen four bits to represent the BM value. This gives a BM range from 0 (closest to the received 6D point) to 15 (furthest from the 6D point). Decoder simulations in [28] indicate that this amount of

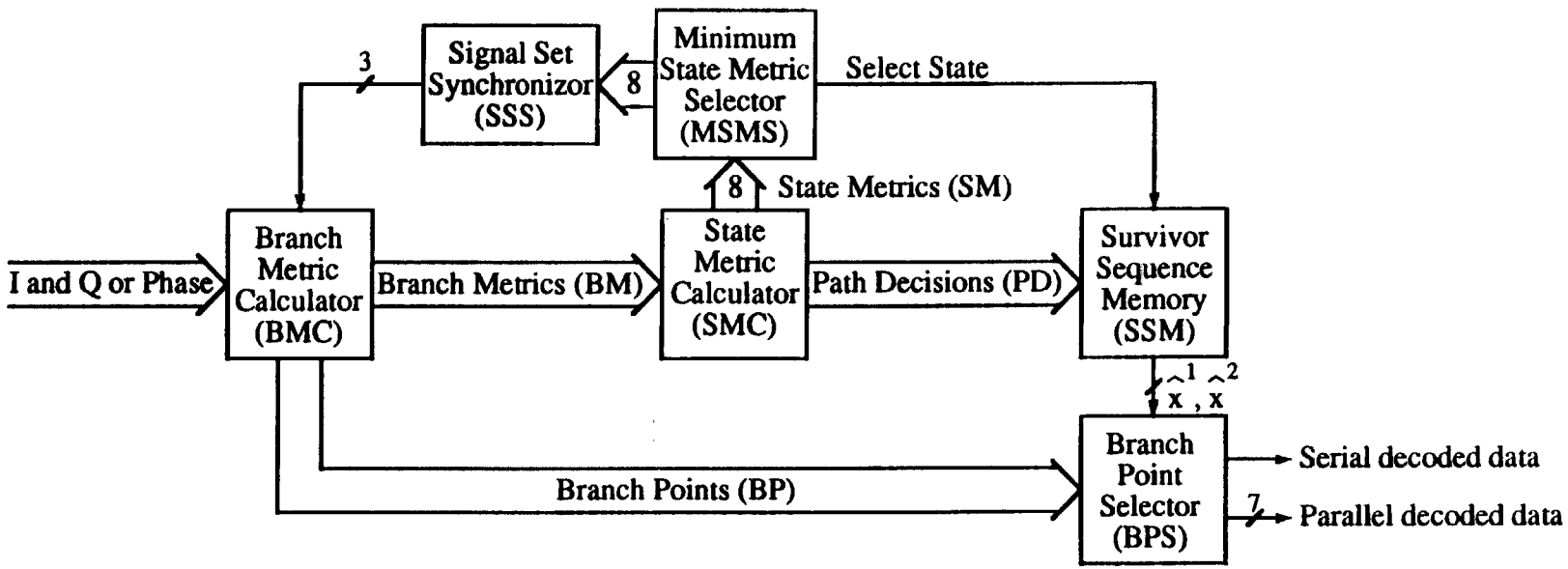

Figure 4.2: Block diagram of a Viterbi decoder for the 16 state 2.33 bit/sym 6D-8PSK trellis code.

quantization results in little performance degradation.

To minimize the number of address bits to the ROM, each received 2D signal point has been quantized to five bits. After extensive simulations in [28], it was found that pie-chart or angular quantization results in the least performance degradation (0.2 to 0.3 dB). The simulations included the "dartboard" quantization pattern proposed in [56]. Cut-off rate calculations in [42] have also confirmed this result.

Each ROM therefore has an address space of 15 bits (five bits for each 2D symbol). The ROM's used for the BMC are 32K×8 27C256's. A total of nine ROM's are required. five for determining the eight BP's and four for the eight BM's.

Alternative BMC schemes which exploit the finite length trellis structure of the parallel transitions were also considered. That is, a Viterbi like decoder can be used to decode the parallel transitions. However, their large complexity (in a discrete implementation) led us to choose the simpler ROM look-up method. For a VLSI implementation, though, the trellis decoding method would be preferable due to the flexibility that VLSI provides in designing circuits. Thus, the Viterbi decoder (with the BMC) could be implemented on a single chip.

4.2.2 State Metric Calculator

The SMC updates the State Metrics (SM) for each state of the code in each iteration of the Viterbi algorithm. A SM is an indication of how close the received sequence is to the closest path of all paths leading into a particular state. Since the code has two checked bits, there are four paths leading into each state (since we choose the

closest path among the 32 parallel paths in the BMC). For each of the four paths, we must add the BM for that path to its corresponding SM (also known as the old SM) from the previous iteration. The new SM for the four paths leading into a state is the smallest of these summations. This path is selected and all other paths are eliminated. This is called the Add-Compare-Select (ACS) operation.

With four paths into each state a 4:1 ACS circuit is required. With 16 states in our code, the ACS operation needs to be performed 16 times (explaining the need for 16 clock cycles). The ACS circuit also produces two Path Decision (PD) bits which indicate which of the four paths was chosen. This information is passed to the SSM where it is stored.

Since the decoder operates serially, only one ACS circuit is required. The 16 SM's are stored in two 74AS870 dual 16×4 static Random Access Memory (RAM) chips. Eight bits are used to represent each SM. As shown in [28], this is more than enough bits when two's complement arithmetic is used in the ACS circuit to prevent overflow [29,48]. Before the first new SM can be calculated, four old SM's are read out from the RAM's. This takes four clock cycles. It takes another two clock cycles to perform the ACS operation. To achieve a slightly higher speed, we could have done the ACS operation in one clock cycle. However, this would have required six comparator chips to find the minimum SM. An increase of one clock cycle and the use of three comparator chips was chosen to decrease the complexity of the design.

Another clock cycle is used to write to the other half of the dual 16×4 RAM's. Since all the read and ACS operations are pipelined, an additional 15 clock cycles are required to write the 15 remaining

new SM's. In the next iteration of the algorithm we read from where the SM's were written in the previous iteration and write to where the old SM's had been stored. The process then repeats.

For the ACS circuit, the appropriate BM's must be added to the correct old SM's. Twelve 2:1 multiplexers and a copy of the convolutional encoder are needed to accomplish this task.

4.2.3 Survivor Sequence Memory

The SSM has two tasks. It must store the Path Decisions (PD's) generated by the SMC and "traceback" through the previously stored PD's to determine the final decoded bits for x^2 and x^1. This requires alternating write and read (for the traceback) operations on the memory. The traceback depth is the required number of PD sets (each set consists of 16 two bit PD's) that the SSM must trace back through.

The PD's must be stored in the remaining 16 clock cycles that are available. There are two ways this can be achieved. Storing two PD bits in each clock cycle or storing four PD bits in every other cycle, leaving the alternate cycle to perform part of the traceback. With the first method at least two separate memories are required since the traceback operation cannot be performed simultaneously with the storage of the new set of PD's (due to the design of memory chips). Since there is a finite amount of memory, the oldest PD set must be written over.

There is usually a point where one method is better than the other (in terms of the total memory size required) based on the number of clock cycles available and the traceback depth. A traceback depth of around 25 to 30 results in little performance degradation [28,42]. Comparing the implementation complexity of the two methods, the

alternating read/write method proved superior.

With this design only eight clock cycles are available to perform a traceback. To maintain integer power of 2 address spaces for the memories (and thus efficiently use practical memory designs), a traceback depth of seven is used for each SSM memory chip. To achieve the required traceback depth, four 64×4 memories are required. This gives a traceback depth of 28. The traceback is performed in a pipelined fashion, switching between memories when required and waiting for the next received set of data to continue with the traceback. Four separate memories are required since there are four tracebacks in operation at any one time.

Since there are no 64×4 RAM's commercially available, larger 256×4 93422A RAM's were used. This chip has separate input and output data buses which simplifies the SSM design. We use the state with the smallest SM to start the traceback. This is the best state the SSM could start with (since it corresponds to the path that is closest to the received signal) and helps give the decoder a slight performance improvement over choosing a random or a fixed state. The Minimum State Metric Selector (MSMS) provides the information needed to achieve this.

At the correct time and place in the circuit, the two decoded bits $\hat{x}^1$ and $\hat{x}^2$ are produced. The two bits are passed to the Branch Point Selector (BPS) where they are re-encoded to select one of the eight 5 bit branch points. The branch points are delayed by 34 6D symbol periods, 28 due to the traceback, 4 due to the pipeline delay in the traceback, and 2 due to the re-encoding of the decoded data.

The seven decoded bits are then differentially decoded (optional) and then parallel to serial converted for the final decoder output.

Precoding and postdecoding are optional as there are some communication systems that do not require phase synchronization. For example, a burst modem can provide phase information in the preamble of a burst. A 74HC4046 PLL is used to generate the required seven times clock for the serial data. This PLL is tuned to lock within 0 to 3 MHz, but as expected for PLL's the lower frequency limit will be somewhat greater than DC. The decoded data is also available in seven bit bytes.

4.2.4 Signal Set Synchronizer

The SSS has the task of synchronizing the decoder to the received sequence of 2D symbols. Since the signal set consists of three 2D symbols, the decoder must synchronize to one of the three possible ways the received data can arrive. Also, since the code is only 90° invariant, the decoder must synchronize to either a 0° or a 45° rotation of the received signal set. Thus, there are a total of six possible combinations of delay and rotation for the received signal.

The decoder is asynchronously locked to DATCLK, which is the received 2D symbol clock whose frequency has been divided by three. A delay of zero, one, or two 2D symbol periods of DATCLK is used for timing synchronization. Since the received Inphase (I) and Quadrature (Q) samples can be converted to polar format (or are already in polar format), the received signal can easily be rotated by 0° or 45° for phase synchronization.

To achieve phase synchronization, the method of using the received 6D signal set as described in Chapter 2 was considered. As pointed out in [28], this method is quite complex. By slightly increasing the complexity of the SSS used for timing synchronization, a

separate synchronizer for phase is avoided.

The SSS works by examining the rate of increase of the minimum SM from the MSMS. If the rate is high, this indicates that the decoder is out of synch and needs to be resynchronized. A variable threshold in the SSS is used for this purpose. If the threshold is exceeded, the SSS will toggle the current phase rotation.

The phase rotation is toggled first since it is more likely that a phase slip has occurred within the demodulator than a 2D symbol slip. The SSS then ignores the decoder for 128+V 6D symbol periods (V is a variable from 0 to 63) to allow the decoder to settle into its new signal set configuration. If the decoder still displays an abnormally high rate of increase for the minimum SM, the SSS will again toggle the phase. Otherwise the decoder goes back to its original monitoring state.

We toggle the phase for a second time under the assumption that the demodulator has experienced a second phase slip (or a burst of noise on the channel created a false alarm, causing the SSS to place the decoder in the incorrect phase). After waiting for 128+V 6D symbol time periods, if the threshold is again exceeded in the next V 6D symbol periods, we toggle the phase for a third time. If the threshold is exceeded again (for the fourth time) we assume that a symbol delay has occurred and we increase the DATCLK delay by 1 (mod 3).

The whole process then repeats until we have the correct 2D symbol delay and phase rotation (indicated by a normal rate of increase of the minimum SM). As can be seen, we give the decoder every opportunity to correct for the more likely phase slip (or even a false alarm from the decoder) than a 2D symbol slip. This implies that phase

slips are corrected quickly, while symbol slips take a very long time to correct.

4.3 Other Decoder Features

The decoder can be mounted within a 3U high 19 inch rack. On the front panel, three Light Emitting Diodes (LED's) are used to indicate the 2D symbol delay and two LED's are used to display the current phase rotation of the SSS.

To test the decoder, the 2D symbol delay and phase rotation can be independently set to manual control. In this way, the SSS can be isolated from the rest of the circuitry so that any problems with the rest of the decoder can be fixed without the SSS interfering. It can also be used to test the SSS by manually introducing phase rotations and delays into the received signal. There are four switches used for this. For 2D symbol control, there is a manual/automatic switch and a three position switch to manually select one of the symbol delays. The phase control also has a manual/automatic switch and a two position switch to select one of the phase rotations.

Two rotary type switches are used to select the format of the received data. One switch is used to select between 3, 4, or 5 bit quantization while the other switch selects between signed magnitude, reverse binary, straight binary, two's complement, and phase data formats. The first four types of data format are used with I & Q samples while the phase data format is used when only quantized phase information is received from the demodulator.

There are also switches for disabling the postdecoder from the

decoder and the precoder from the encoder. The encoder has another switch to select between seven bit parallel or bit serial data. One final switch is used to loopback the data produced by the encoder to the Viterbi decoder. This provides a useful self-test for the encoder/decoder system. The encoder/decoder interface diagram is given in Figure 4.3.

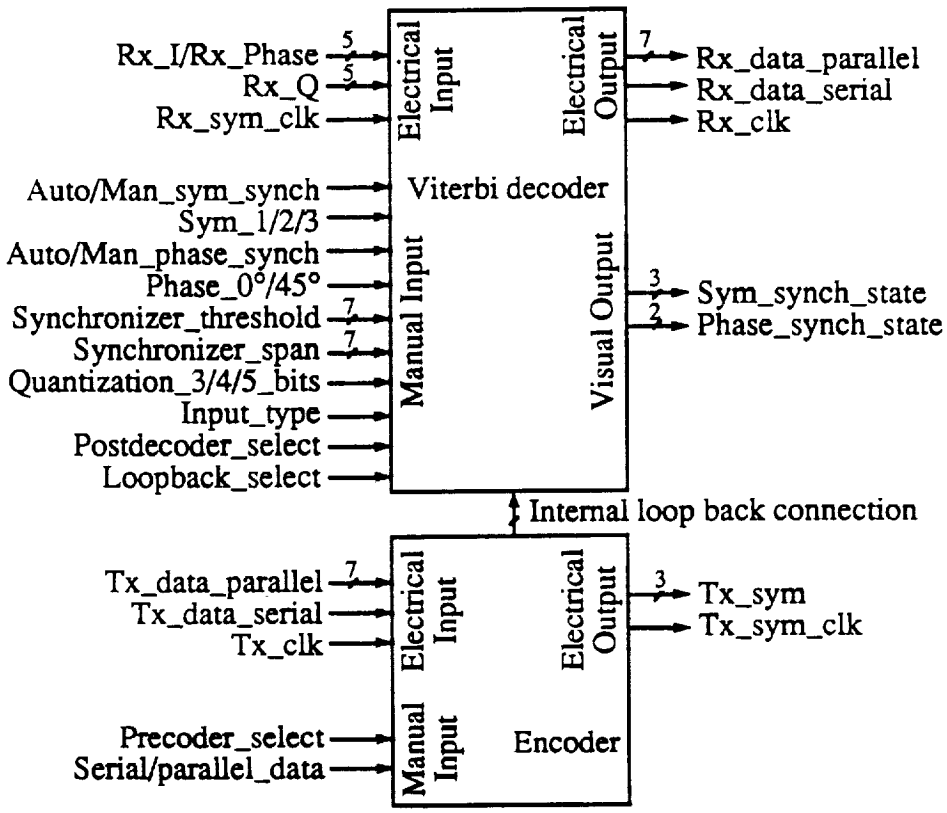

Figure 4.3: Viterbi decoder/encoder interface diagram for 16 state 2.33 bit/sym 6D-8PSK trellis code.

The 176 integrated circuits of the design are placed on two double height Speedwire Eurocards (233.4×220 mm). Speedwire allows quick and reliable connections (if it is done correctly) between the chips that can be easily changed. The speedwire boards also have good

groundplanes, critical when operating at high clock speeds. The Viterbi decoder (which operates at 10 MHz) is placed on one board (taking 96 chips) while the encoder, SSS, and various interface chips are placed on the other board.

BNC connectors are used at the back of the rack for external data and clock connections. It is assumed that all received data changes on the rising edge of its clock. Similarly, the codec produces its signals in the same format. TTL 50/70 Ω interface signals are used for these external interfaces.

4.4 An Alternative Viterbi Decoder

The decoder described above can be modified to implement the 16 state 2.5 bit/sym 4D-8PSK trellis code given in Table 2.16(a). This code has the advantage of being fully transparent and of using a smaller size signal set (4D instead of 6D). This implies that decoder synchronization is much easier to perform. Its disadvantage is a decrease in coding gain of 0.7 dB at $P_b = 10^{-5}$ compared to the 2.33 bit/sym code [46].

The 2.5 bit/sym code could be used for the new SONET standard requiring 155.52 Mbit/s through a 72 MHz transponder. With a parallel implementation of the decoder (requiring only one clock cycle to decode five bits in each iteration of the Viterbi algorithm) an internal decoder speed of 31.104 MHz (1/5 the bit rate) is required. The 2D symbol rate is equal to the bit rate divided by K, i.e., 62.204 Msym/s. With 72 MHz of bandwidth available, a bandwidth expansion factor of no more than 1.157 (equal to the bandwidth divided by the 2D symbol rate)

needs to be met by the modulation system. Converting the parity check polynomials from Table 2.16(a) to non-systematic form we obtain,

$$z^2(D) = x^2(D) \oplus (D^2 \oplus 1)x^1(D), \tag{4.2a}$$
$$z^1(D) = D^2x^2(D) \oplus (D^2 \oplus D \oplus 1)x^1(D), \tag{4.2b}$$
$$z^0(D) = Dx^2(D). \tag{4.2c}$$

An implementation of the non-systematic encoder for the 2.5 bit/sym code is given in Figure 4.4. Converting the 2.33 bit/sym design will require changes to the BMC and various interface circuitry as well as a new simpler SSS. Due to the smaller size signal set, the total chip count can be expected to decrease. Also, since only five bits are decoded during each iteration, the decoder speed will decrease to 2.17 Mbit/s.

4.5 Conclusions

A serial implementation of a Viterbi decoder for the 16 state 2.33 bit/sym code with a 6D-8PSK signal set has been described. This decoder can provide high data rates (up to 3 Mbit/s) and is intended for future use on the Hubble Space Telescope. Due to its serial implementation the decoder design is quite complex, but could be implemented on a single VLSI integrated circuit.

The Branch Metric Calculator has been implemented through the use of large look-up table ROM's. A VLSI implementation may use a Viterbi type decoding algorithm to allow single chip implementation. An alternative 2.5 bit/sym 4D-8PSK code has been proposed for use in the new SONET 155.52 Mbit/s transmission system.

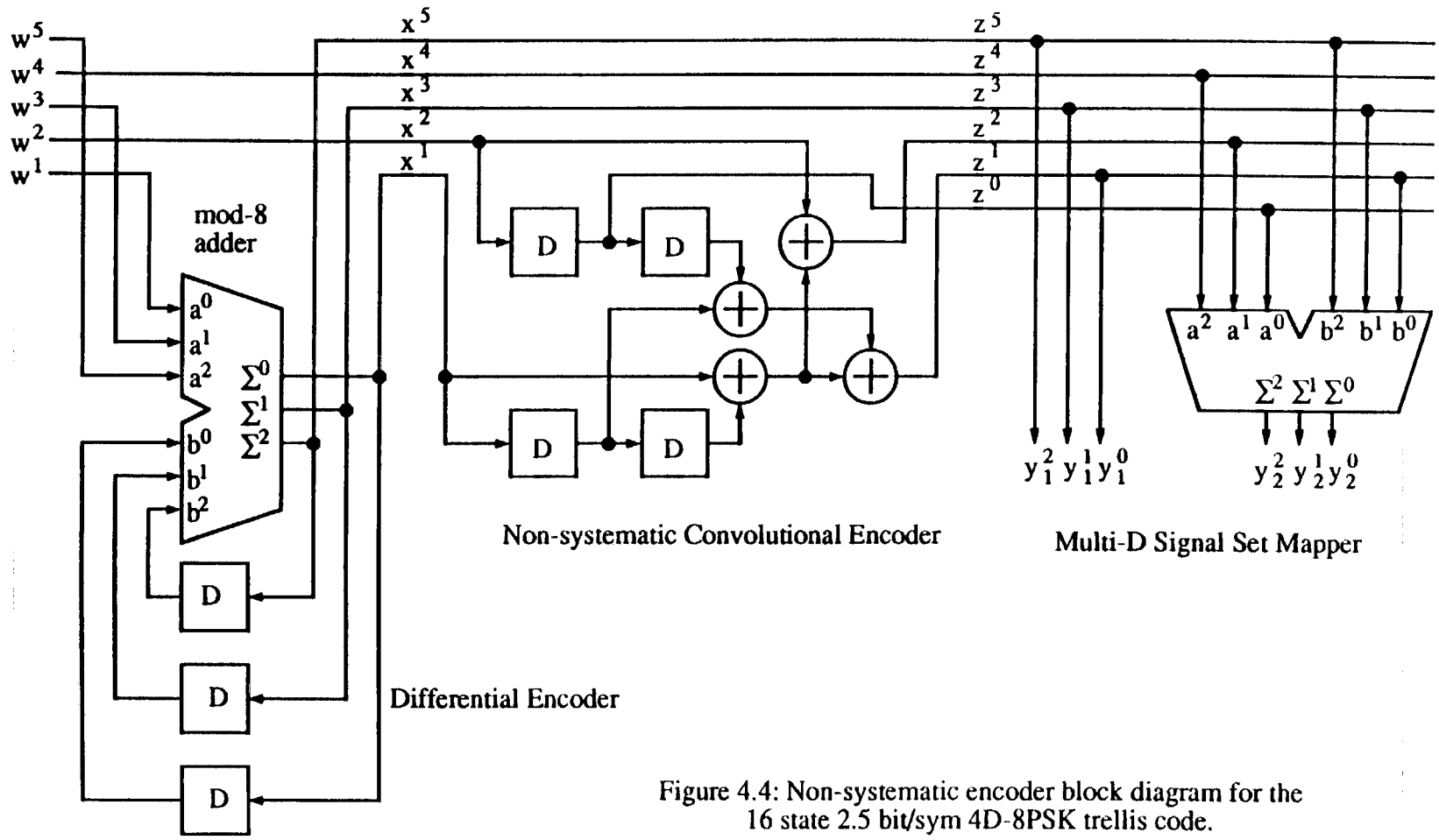

Figure 4.4: Non-systematic encoder block diagram for the 16 state 2.5 bit/sym 4D-8PSK trellis code.

CHAPTER FIVE

ROTATIONALLY INVARIANT TRELLIS CODES

One aspect of trellis codes that has come under increasing study is the search for codes that are invariant to phase rotations of the received signal set [1,3,19,32,40,44,48,56,70,73-75,81]. The rotations that we are considering are those caused by a demodulator in a communications system. When the signal set has rotational symmetries, e.g., MPSK or 16QAM, the demodulator has no knowledge of which of the symmetries were transmitted. Thus, the demodulator selects one of the symmetries in which to demodulate the received signal, regardless if it is the correct or incorrect symmetry.

In uncoded systems, this problem is easily corrected by differentially encoding (*precoding*) the data before transmission. After demodulation, differential decoding (*postdecoding*) of the received data is then used to return the data to its original form. The precoding of the data also allows the recovery of data caused by phase slips within the demodulator. This occurs when noise in the received signal causes the demodulator to lose lock and results in another of the signal set symmetries being selected.

For trellis codes the situation is much more complicated. Here, we are dealing with sequences of symbols in the code space rather than independent symbols, as in the uncoded case. In fact, convolutional and trellis codes can be thought of as subclasses of *sequence codes*. Unlike

block codes, sequence codes have code words of infinite length, consisting of sequences of symbols taken from a finite or infinite size signal set. Any finite or infinite set of sequences can be considered as a sequence code. If a coded sequence has been rotated, the resulting code sequence may or may not be in the code space.

The *transparency* or *rotational invariance* of a sequence code is the minimum non-zero phase rotation that all code sequences in the code space can be rotated such that the rotated sequences are still in the code space of the sequence code. A sequence code is *rotationally invariant* or *transparent* if the invariance of the code is equal to the minimum non-zero phase symmetry of the two-dimensional (2-D) signal set. If there are some sequences which are not in the code space after a phase rotation, a decoder will produce erroneous data if the received sequence has been rotated by this amount.

A good example of this is the industry standard (2,1,6) convolutional code with Gray mapped QPSK modulation. This code is not 90° transparent (and is therefore not rotationally invariant), but it is 180° transparent. A decoder will produce erroneous data after a 90° or -90° rotation. To overcome this, the decoder needs to recognize that a 90° rotation has occurred and rotate the received sequence. This process can be slow, resulting in many errors being produced before the decoder is properly synchronized. A rotationally invariant code, however, will only produce a small number of errors after a phase rotation, since there is no need to detect and then correct for a phase rotation.

In order to describe and study rotationally invariant sequence codes, we will be using parity check equations (PCE). For rate k/(k+1)

codes, a single PCE fully describes the relationship between the 2-D symbols in a coded sequence. However, the PCE gives no information about the input/output relationship of an encoder, i.e., it is independent of the encoder implementation. This allows us to minimize the number of variables in finding good rotationally invariant codes, thus simplifying the code search.

Rotationally invariant rate 1/2 QPSK codes based on a non-linear PCE have been found in [70]. This Chapter extends this work by presenting a general PCE (actually, it is two equations that can be combined into one), from which good rate k/(k+1) rotationally invariant trellis codes can be found.

We first show that linear codes cannot be rotationally invariant. This is followed by the presentation of the general PCE and how it is used to find good rotationally invariant trellis codes. Finally, we present the results of a systematic code search for rotationally invariant QPSK, 8PSK, 16PSK, and QAM signal set codes.

5.1 Sequence Codes with Linear Parity Check Equations

For a 2-D signal set with 2^{k+1} points, let $(y^0,y^1,...,y^k)$, where $y^i \in \{0,1\}$ for $0 \le i \le k$, be a binary representation of each point in the signal set. Also, let $y^0(D)$, $y^1(D),...,y^k(D)$ be the binary sequences that form a sequence of signal set points (where D is the delay operator).

We formally define a *parity check equation* (PCE) for a rate k/(k+1) sequence code as an equation which defines the relationship between the encoded binary sequences $y^0(D)$, $y^1(D),...,y^k(D)$. A PCE is

linear (under modulo-2 addition) if it is in the following form,

$$H^k(D)y^k(D) \oplus \cdots \oplus H^1(D)y^1(D) \oplus H^0(D)y^0(D) = 0(D), \tag{5.1}$$

where $H^i(D)$ for $0 \le i \le k$ are binary polynomials and $0(D)$ is the all zero sequence. The $H^i(D)$'s are also known as *parity check polynomials* and are of the form

$$H^i(D) = h_i^{\nu}D^{\nu} \oplus \cdots \oplus h_i^1 D \oplus h_i^0, \tag{5.2}$$

where $h_i^j \in \{0,1\}$ for $0 \le i \le k$, $0 \le j \le \nu$, and ν is the maximum degree of all the parity check polynomials. For practical codes, ν is finite and is called the *constraint length* of the code. The *memory* (m) of a sequence code is the minimum number of delay elements required to implement an encoder. It has been shown in [20] that $m = \nu$ for linear PCE's.

When a phase rotation of a signal set occurs, the binary sequences representing the signal set sequence will change. We call these the *rotated sequences* which are labeled by $y_r^0(D)$, $y_r^1(D),\ldots,y_r^k(D)$. These rotated sequences are a function of the original unrotated sequences. To determine the effect of a phase rotation we substitute these rotated sequences into the PCE (i.e., $y^i(D)$ is replaced by $y_r^i(D)$). If the resulting equation is exactly the same as the original PCE, the code has a transparency equal to that phase rotation.

Example 5.1 Rate 1/2 convolutional code

Let us examine the standard rate 1/2, memory six convolutional code with Gray QPSK mapping. Figure 5.1 shows the Gray mapped QPSK signal set. This code is usually expressed in its encoder form, i.e.,

$$y^0(D) = (D^6 \oplus D^4 \oplus D^3 \oplus D \oplus 1)x(D) = g^0(D)x(D), \tag{5.3}$$
$$y^1(D) = (D^6 \oplus D^5 \oplus D^4 \oplus D^3 \oplus 1)x(D) = g^1(D)x(D), \tag{5.4}$$

where x(D) is the binary input sequence of the encoder. Combining (5.3) and (5.4) we obtain the PCE

$$g^0(D)y^1(D) \oplus g^1(D)y^0(D) = 0(D). \tag{5.5}$$

That is $H^1(D) = g^0(D)$ and $H^0(D) = g^1(D)$.

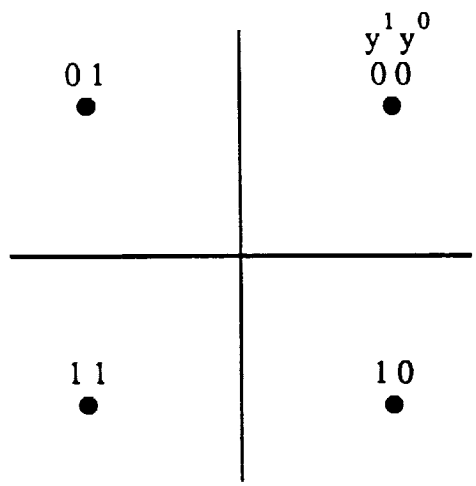

Figure 5.1: Gray mapped QPSK.

We now investigate what happens when the signal set is rotated. From Figure 5.1 we see that after a 90° rotation the rotated sequences are

$$y_r^0(D) = y^1(D) \oplus 1(D) = \overline{y^1(D)}, \tag{5.6}$$

$$y_r^1(D) = y^0(D), \tag{5.7}$$

where 1(D) is the all ones sequence. Replacing $y^0(D)$ with $y_r^0(D)$ and $y^1(D)$ with $y_r^1(D)$ we have

$$\begin{aligned} g^0(D)y^0(D) \oplus g^1(D)(y^1(D) \oplus 1(D)) &= 0(D), \\ g^0(D)y^0(D) \oplus g^1(D)y^1(D) &= g^1(D)1(D). \end{aligned} \tag{5.8}$$

Since there is an odd number of non-zero terms in $g^1(D)$ and any delay of 1(D) is equal to 1(D), then $g^1(D)1(D) = 1(D)$. That is,

$$g^0(D)y^0(D) \oplus g^1(D)y^1(D) = 1(D). \tag{5.9}$$

Comparing (5.5) and (5.9), we see that they are not the same, thus implying that this code is not 90° transparent. In fact, no good code (i.e., $g^0(D)$ and $g^1(D)$ are at least different from each other) can be 90° transparent for a rate 1/2 code with a linear PCE and Gray QPSK mapping. Using a similar method, it can be shown that this code is 180° transparent.

As shown above, the effect a phase rotation has on the PCE depends a great deal on the form of the rotated sequences. More specifically, it depends on the particular signal set mapping that is used. We discuss this matter in the next section.

5.1.1 Signal Set Mappings

Since we can't have all the bits in a mapping unaffected by a phase rotation, we should try to minimize the effect as much as possible. The mapping should also be consistent with the signal set

partitioning schemes descibed in [65], since these partitions help us to find good sequence codes.

The so-called "natural" mapping has these properties. The naturally mapped MPSK signal set is a good example of this. For example, naturally mapped 8PSK. It has the desired partition properties and the following rotated sequence equations,

$$y_r^0(D) = y^0(D) \oplus 1(D) = \overline{y^0(D)}, \tag{5.10a}$$

$$y_r^1(D) = y^1(D) \oplus y^0(D), \tag{5.10b}$$

$$y_r^2(D) = y^2(D) \oplus y^1(D) \cdot y^0(D). \tag{5.10c}$$

The multiplication in (5.10c) involve the bit wise logical AND of the coefficients of the polynomials, e.g., if $y^0(D) = D^2 \oplus D \oplus 1$ and $y^1(D) = D \oplus 1$ then $y^0(D) \cdot y^1(D) = D \oplus 1$. That is, this polynomial multiplication is a non-linear operation. Note that each rotated bit is a function of itself plus some added term. Thus, when substituted into a PCE, the original PCE is produced plus some extra terms. We will discuss this in greater detail later.

A simpler way of describing what happens to the binary sequences after a phase rotation is with integer notation and modulo-M addition. We let the integer representation of $y^0(D)$, $y^1(D)$,..., $y^k(D)$ be

$$y(D) = \sum_{i=0}^{k} 2^i y^i(D). \tag{5.11}$$

Thus, with natural mapping, the rotated sequence $y_r(D)$ (in integer notation) can be expressed as

$$y_r(D) = y(D) + 1(D) \pmod{M}. \tag{5.12}$$

With rectangular signal sets we should also try to obtain a "natural" mapping. We will show how this can be achieved by using the 16QAM signal set. Similar to MPSK, we use a mapping based on the partitioning of the 16QAM signal set as an example. Figure 5.2 illustrates the first two levels of the partition. Note that the four subsets are related to each other by an appropriate number of 90° phase rotations. Each of these four subsets can be labeled by the first two bits of the mapping, i.e., y^0 and y^1. After a 90° rotation the rotated sequences for y^0 and y^1 are given by (5.10a) and (5.10b), respectively.

The remaining two bits (y^2 and y^3) are used to define the four way partition of each of the four subsets. However, as shown in Figure 5.2, these last two bits are mapped such that they are not affected by a 90° rotation, i.e.,

$$y_r^2(D) = y^2(D), \tag{5.13a}$$

$$y_r^3(D) = y^3(D). \tag{5.13b}$$

This considerably simplifies finding a PCE, since only two sequences are affected by a rotation. A similar approach can be used with other rectangular signal sets (e.g., 32CROSS). For the 32CROSS mapping in [1] (and which was accepted as the V.32 modem standard) the first *three* bits are affected by a rotation. This results in a more complicated encoder requiring two AND gates (all our codes require only one AND gate).

We will be constructing codes based on these signal sets since they have relatively simple rotation equations.

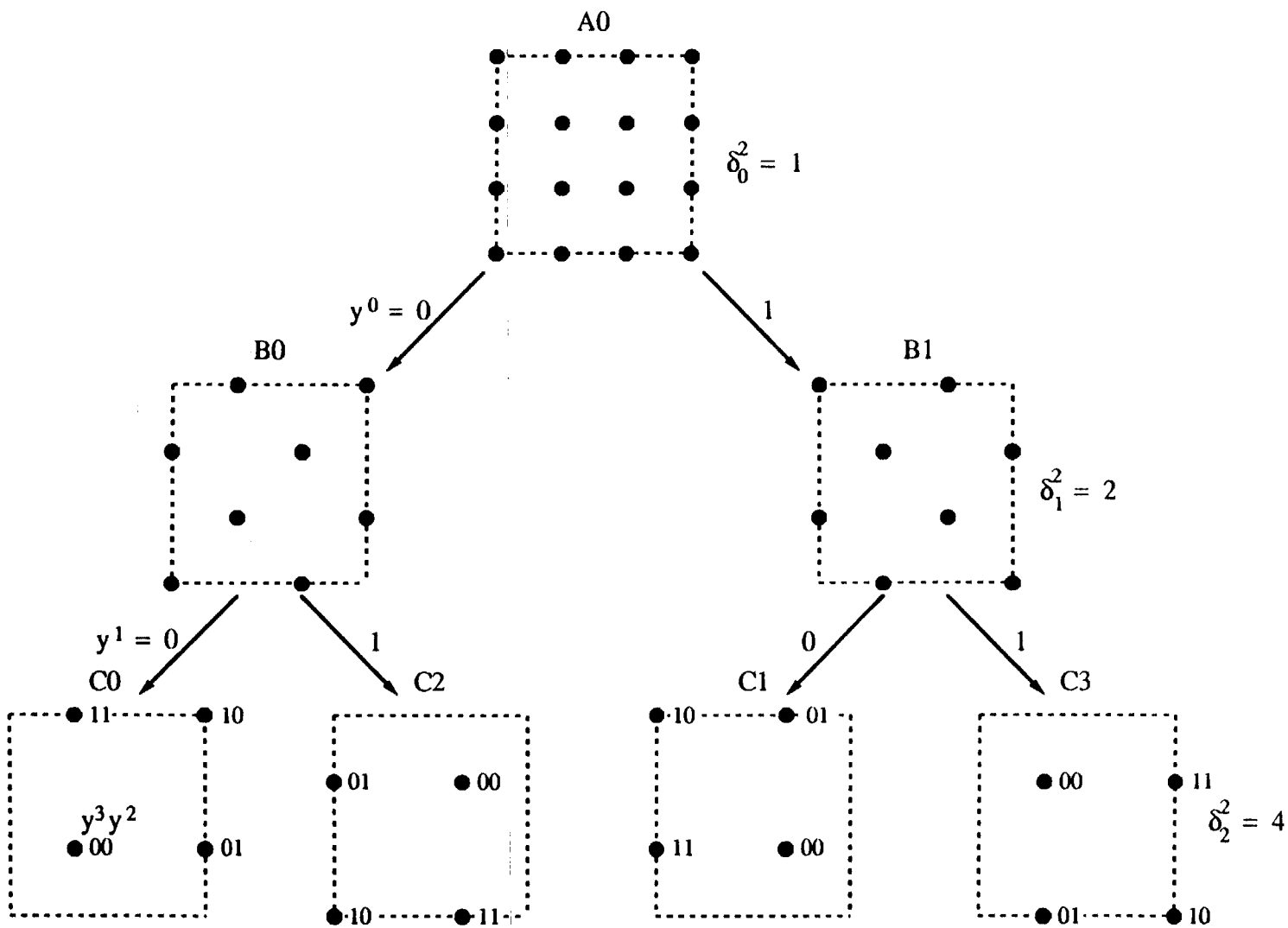

Figure 5.2: Partitioning of "naturally" mapped 16QAM.

5.1.2 More on Linear Parity Check Equations

We will now show that linear PCE's cannot be made invariant when two or more binary sequences are affected by a rotation. We will use the rotation equations given in Section 5.1.1. For simplicity, we will only consider two bits are affected by a rotation, i.e., equations (5.10a) and (5.10b). Substituting equations (5.10a) and (5.10b) for $y^0(D)$ and $y^1(D)$ into the linear PCE (5.1) we obtain

$$H^1(D)(y^1(D) \oplus y^0(D)) \oplus H^0(D)(y^0(D) \oplus 1(D)) = 0(D),$$
$$H^1(D)y^1(D) \oplus (H^0(D) \oplus H^1(D))y^0(D) = E[H^0(D)](D), \qquad (5.14)$$

where $E[H^0(D)]$ is the modulo-2 number of non-zero terms in $H^0(D)$. Even if $E[H^0(D)] = 0$, (5.14) is not equal to the original PCE (5.1). This shows that if two sequences being affected, we cannot have an invariant linear code. When more than two sequences are affected, the situation is even worse. For these cases, non-linear terms (such as $y^0(D)\cdot y^1(D)$) are produced.

5.2 A General Parity Check Equation for Invariant Sequence Codes

We now introduce our general parity check equation(s) (GPCE) that can be used to find any rotationally invariant rate k/(k+1) sequence code. To simplify the derivation, we will assume that all k bits are checked by the encoder and that all I = k+1 bits in the signal set are affected by a rotation. This basic equation can then be modified for other sequence codes where these conditions are not true (e.g., rate 2/3 8PSK with one checked bit or rate 3/4 16QAM).

The GPCE is actually two equations, one that uses modulo-M

arithmetic and the other modulo-2 arithmetic. The two equations need to be combined in order to produce the final invariant PCE (IPCE). After the introduction of the GPCE, we will illustrate its use by deriving the rate 1/2 IPCE as found in [70]. This is followed by the derivation of a rate 2/3 IPCE for 8PSK modulation and two checked bits.

5.2.1 The Equations

Our basic aim is the introduction of a non-linear term (as in [70]) into the linear PCE so that after a rotation the non-linear term "generates" additional terms which cancel the terms generated by the linear part of the IPCE. We start by assuming that $E[H^0(D)] = 1$. This may seem contrary to what is desired (we have that $H^0(D)y_r^0(D) = H^0(D)y^0(D) \oplus 1(D)$), but this assumption results in a small simplification in the final IPCE. We could have started with $E[H^0(D)] = 0$, but as shown later, there is an equivalent IPCE with $E[H^0(D)] = 1$.

Since the $H^0(D)y^0(D)$ term generates $1(D)$ after a rotation, the non-linear term must also generate $1(D)$. We also assume that all $I = k+1$ bits are affected by a rotation. The modulo-M GPCE is

$$z(D) = (D^a + (M/2-1)D^b)y(D) \pmod{M}, \tag{5.15}$$

where $y(D)$ is defined in (5.11), $z(D) = \sum_{i=0}^{k} 2^i z^i(D)$, $z^i(D)$ are binary sequences, and $\nu > a > b > 0$. The restriction on a and b allows us to find good sequence codes.

Before presenting the modulo-2 GPCE, we examine what happens to $z(D)$ on a phase rotation. Substituting $y_r(D)$ from (5.12) for $y(D)$, we obtain (all additions are modulo-M),

$$
\begin{aligned}
z_r(D) &= (D^a + (M/2\text{-}1)D^b)y_r(D) \\
&= (D^a + (M/2\text{-}1)D^b)(y(D) + 1(D)) \\
&= (D^a + (M/2\text{-}1)D^b)y(D) + D^a 1(D) + D^b(M/2\text{-}1)(D) \\
&= z(D) + M/2(D).
\end{aligned}
$$

Note that a number before the term "(D)" is an infinite sequence of that number, e.g., M/2(D) is an infinite sequence of the number M/2.

We see that z(D) has the M/2(D) sequence added to it. In terms of the binary sequences, we have $z_r^i(D) = z^i(D)$ for $0 \le i \le k\text{-}1$ and $z_r^k(D) = z^k(D) \oplus 1(D)$, i.e., all the binary sequences in z(D) are unaffected, except the most significant binary sequence, which is inverted. In the modulo-2 GPCE all the terms in the equation must be unaffected by a rotation or produce terms that cancel each other.

Since $z^k(D)$ is the most significant bit of z(D), it is a function of all $y^i(D)$. Thus, by always including this term, the IPCE will check all k input bits to the encoder and avoid parallel transitions. Also, after a rotation, $z^k(D)$ generates a 1(D) sequence which will cancel the 1(D) generated by $H^0(D)y^0(D)$. Since the remaining $z^i(D)$ terms are unaffected by a phase rotation, we include a linear combination of $z^i(D)$, $0 < i < k$ to increase the number of invariant codes that can be examined in a code search. We don't include $z^0(D)$ since it is a linear function of $y^0(D)$ (which is taken care of by $H^0(D)y^0(D)$).

The modulo-2 GPCE therefore is

$$z^k(D) \oplus h_z^{k-1}z^{k-1}(D) \oplus \cdots \oplus h_z^1 z^1(D) \oplus H^0(D)y^0(D) = 0(D), \qquad (5.16)$$

where $h_z^i \in \{0,1\}$ for $0 < i < k$. From (5.15) we can obtain expressions

for $z^i(D)$ in terms of $y^i(D)$ and substitute them into (5.16) to obtain the IPCE.

If $E[H^0(D)] = 0$ the modulo-M PCE will be of the form

$$\begin{aligned} z(D) &= (D^a + (M-1)D^b)y(D) \\ &= (D^a + M/2D^b + (M/2-1)D^b)y(D) \\ &= (D^a + (M/2-1)D^b)y(D) + M/2D^b y(D) \\ &= (D^a + (M/2-1)D^b)y(D) + M/2D^b y^0(D). \end{aligned}$$

Thus, it can be seen that $z^k(D)$ will be the same as the $z^k(D)$ obtained from (5.15), except that $D^b y^0(D)$ will be added to it. When $z^k(D)$ is substituted into (5.16), the extra $D^b y^0(D)$ term will be added to $H^0(D)y^0(D)$ (forming the new parity check polynomial $\tilde{H}^0(D)$). This makes $\tilde{H}^0(D)$ have an odd number of non-zero terms, or $E[\tilde{H}^0(D)] = 1$. Therefore, the restriction $E[H^0(D)] = 1$ covers all possible codes and also results in a simpler IPCE.

Note that the form of (5.15) is not unique. Providing that $z^k(D)$ is a function of all $y^i(D)$, then three or more delay terms in (5.15) could be used. For example, with $E[H^0(D)] = 1$ and $I = 3$,

$$z(D) = (D^a + 4D^b + 7D^c)y(D) \pmod 8. \tag{5.17}$$

For the codes found in this chapter we have used (5.15) for z(D). The reason for this is to simplify the code search. Also, work done in [48] for rotationally invariant QPSK codes indicates that having three or more delay terms does not give an increase in free distance compared to using only two delay terms. The affect on the number of nearest neighbors is not known though.

5.2.2 The Rate 1/2 Invariant Parity Check Equation

The original rate 1/2 IPCE equation found by Ungerboeck was obtained by hand. Here (in a slightly modified form) we present its derivation using the GPCE. This equation can be used in rate k/(k+1) MPSK codes where the encoder checks only one of the k input bits to the encoder, in rate k/(k+1) QAM codes where there are four rotational symmetries, and in other 2-D or multi-dimensional signal sets.

We let the binary input sequence be

$$x(D) = x^1(D), \tag{5.18}$$

and the output sequence be

$$y(D) = y^0(D) + 2y^1(D). \tag{5.19}$$

From (5.15) the modulo-4 GPCE is

$$z(D) = (D^a + D^b)y(D) \pmod 4. \tag{5.20}$$

The next step is to express z(D) in terms of $y^0(D)$ and $y^1(D)$. Substituting (5.19) into (5.20) and expanding z(D), we obtain

$$z^0(D) + 2z^1(D) = (D^a + D^b)y^0(D) + 2(D^a + D^b)y^1(D) \pmod 4. \tag{5.21}$$

To find $z^0(D)$ and $z^1(D)$ in terms of $y^0(D)$ and $y^1(D)$, the logic equations of a two bit adder are used. Figure 5.3 illustrates how two 2 bit adder blocks can be joined to give the required $z^0(D)$ and $z^1(D)$ outputs.

We have for each 2 bit adder that

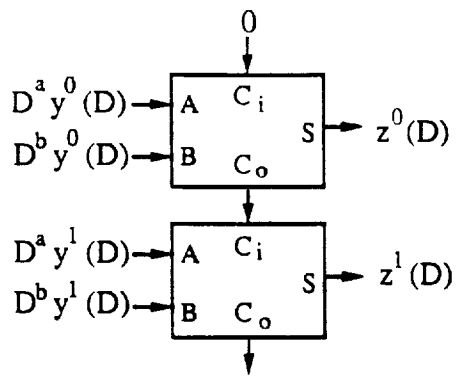

Figure 5.3: Two bit adders used to form z(D) for rate 1/2 encoder.

$$S = A \oplus B \oplus C_i, \tag{5.22a}$$

$$C_o = A \cdot B \oplus C_i \cdot (A \oplus B). \tag{5.22b}$$

We thus have

$$z^0(D) = (D^a \oplus D^b) y^0(D), \tag{5.23a}$$

$$z^1(D) = (D^a \oplus D^b) y^1(D) \oplus D^a y^0(D) \cdot D^b y^0(D). \tag{5.23b}$$

Notice that $z^1(D)$ contains a non-linear term. A logic AND gate is used to implement this function. From (5.16) we obtain,

$$z^1(D) \oplus H^0(D) y^0(D) = 0(D). \tag{5.24}$$

Substituting (5.23b) into (5.24) the final IPCE is

$$(D^a \oplus D^b) y^1(D) \oplus D^a y^0(D) \cdot D^b y^0(D) \oplus H^0(D) y^0(D) = 0(D). \tag{5.25}$$

As for a linear PCE, the parity check polynomial of $y^1(D)$ is $H^1(D) = D^a \oplus D^b$. Since this polynomial gives the values of a and b, the code can be completely described by $H^0(D)$ and $H^1(D)$.

Note that (5.25) is similar to the linear PCE (5.1), but with

$H^1(D)$ having only two delay terms and a non-linear term that is added. Codes for rate 1/2 QPSK based on (5.25) can be found in [70], where the encoder implementation is discussed in detail. Substituting the rotated forms of $y^0(D)$ and $y^1(D)$ into (5.25) will easily show that this equation is invariant.

As in [70], we will use systematic encoders to implement the IPCE. Since the IPCE is non-linear, it is not always possible to find a minimal (i.e., $m = \nu$) implementation of the encoder. An equation to determine m is given in [70]. It can be shown, however, that for $a - b \leq 2$, a minimal implementation of the encoder can be found. For $a - b > 2$, a restriction on $H^0(D)$ needs to be made in order to obtain a minimal implementation.

Example 5.2 Rate 2/3 8PSK with one checked bit

In this example we present a simple eight state code for rate 2/3 8PSK and one checked bit. With $\nu = 3$, we must have $a = 2$ and $b = 1$. Thus our parity check equation is

$$(D^2 \oplus D)y^1(D) \oplus D^2y^0(D)\cdot Dy^0(D) \oplus (D^3 \oplus h_0^2D^2 \oplus h_0^1D \oplus 1)y^0(D) = 0(D). \quad (5.26)$$

There are no $y^2(D)$ terms since $H^2(D) = 0$. An implementation of this encoder is shown in Figure 5.4. It is similar to a linear systematic encoder, except for the non-linear term that is added. Notice that the AND gate multiplies the sequences $y^0(D)$ and $D^{-1}y^0(D)$ to form $y^0(D)\cdot D^{-1}y^0(D)$. The output of the AND gate is then twice delayed to obtain the correct non-linear term.

There are two possible codes, one with $h^0 = 13_8 \equiv 1011_2 \equiv D^3 \oplus D \oplus 1 = H^0(D)$ (all codes will be given in octal notation) and

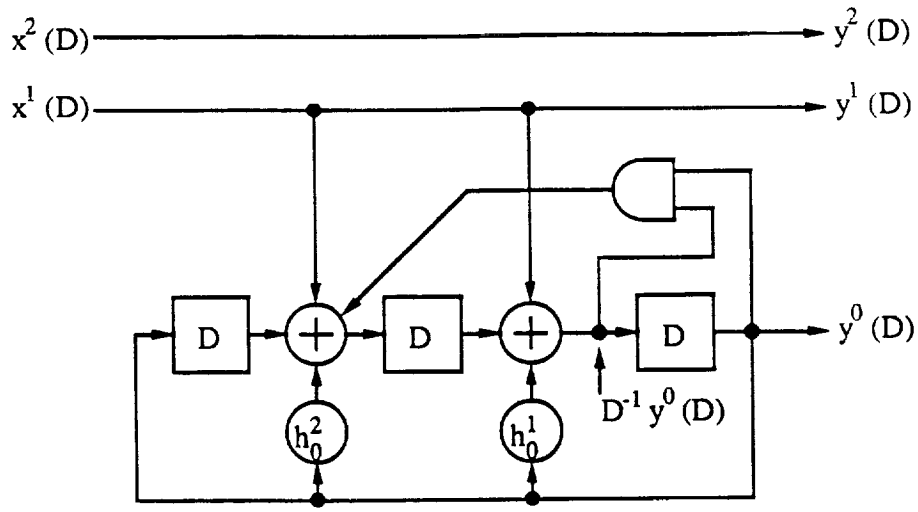

Figure 5.4: Rotationally invariant rate 2/3 8PSK encoder ($\tilde{k} = 1$, $m = 3$).

the other with $h^0 = 15$. Since the free distance of a code is the same going forwards or backwards in time, these two codes must have the same free distance (the bit reverse of $h^1 = 06$ is 06 and of $h^0 = 13$ is 15). This time reversal technique (first given in [65]) can be used to reduce the number of codes that need to be examined in a search for more complex codes. The minimum free squared Euclidean distance (d^2_{free}) is 4.0, which occurs along parallel transitions. Thus an asymptotic coding gain (γ) of 3.01 dB is achieved. The number of nearest neighbors (N_{free}) is equal to one. The smallest squared Euclidean distance that occurs along non-parallel transitions (d^2_{next}) is 4.586. Its average multiplicity (N_{next}) is 0.25. An eight state linear code has $d^2_{free} = 4.586$, but is only 180° transparent [56].

5.2.3 Rotationally Invariant QAM Codes

As shown in Section 5.1.1, only the two least significant bits of a QAM signal set mapping are affected by a 90° phase rotation. With an

appropriate mapping, these two least significant bits are affected in the same way as the two bits in naturally mapped QPSK.

The IPCE in (5.25) can be used if there is only one checked bit. However, the d^2_{free} will be at most limited to 4 (with the minimum free squared Euclidean distance of the uncoded signal set equal to 2). For $\nu = 3$ and one checked bit, (5.26) gives a code with $d^2_{free} = 4$ (along the parallel transitions). To increase the free distance, the number of checked bits ($\tilde{k}$) needs to be increased. The IPCE for this more general case is

$$H^{\tilde{k}}(D)y^{\tilde{k}}(D) \oplus \cdots \oplus H^2(D)y^2(D) \oplus (D^a \oplus D^b)y^1(D) \oplus H^0(D)y^0(D)$$
$$= D^a y^0(D) \cdot D^b y^0(D). \tag{5.27}$$

The $y^{\tilde{k}}(D)$ to $y^2(D)$ terms can be added just as in the linear PCE since these terms are unaffected by a phase rotation.

Example 5.3 Rate 3/4 16QAM with $\tilde{k} = 2$

We let $\nu = 3$, $a = 2$, $b = 1$, $h^0 = 13$, $h^1 = 06$, and $h^2 = 02$. Figure 5.5 illustrates a systematic encoder implementation. This code is very similar to the code given in [32] except that a 16QAM signal set is used instead of 32CROSS. It has $d^2_{free} = 5$ (when normalized to a signal energy of one, the minimum free squared Euclidean distance is 2.0, equal to that of uncoded QPSK), which gives an asymptotic coding gain of 3.98 dB. The average number of nearest neighbors is 0.875, compared with the best 180° invariant linear code, which has an N_{free} of 3.781. This indicates that the fully invariant code should perform better than the best linear code.

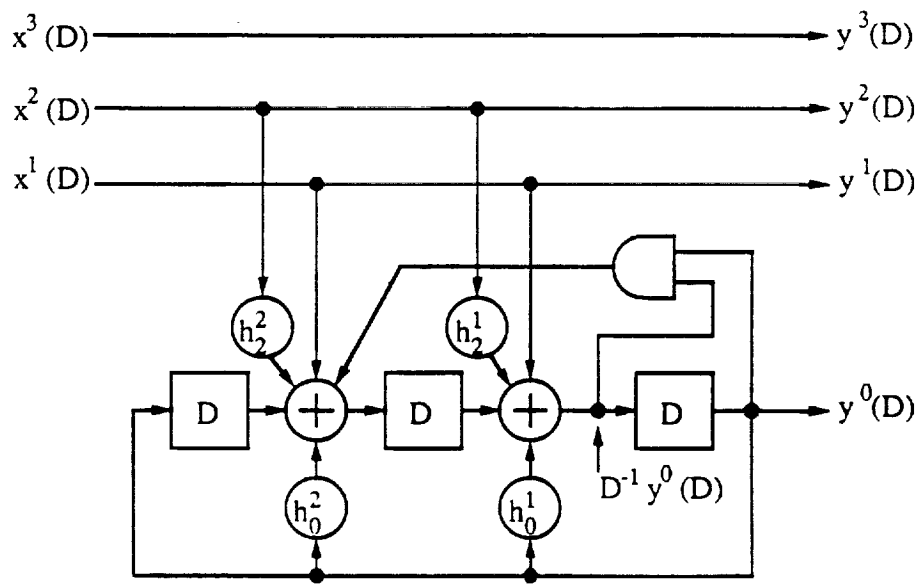

Figure 5.5: Rotationally invariant rate 3/4 16QAM encoder ($\tilde{k} = 2$, $m = 3$).

5.2.4 The Rate 2/3 Invariant Parity Check Equation

Obtaining a rate 2/3 IPCE by hand is very difficult and was one of the main reasons for finding the GPCE. The only use found thus far for this equation is for rate 2/3 8PSK with two checked bits. The best rate 3/4 16PSK codes have only one checked bit up to $\nu = 7$. Thus the rate 1/2 IPCE can be used for 16PSK.

For the rate 2/3 IPCE, we start with the modulo-8 GPCE. We have

$$z(D) = (D^a + 3D^b)y(D) \pmod 8 \quad (5.28a)$$

$$z(D) = (D^a + D^b)y^0(D) + 2((D^a + D^b)y^1(D) + D^b y^0(D)) + 4((D^a + D^b)y^2(D) + D^b y^1(D)). \pmod 8 \quad (5.28b)$$

Figure 5.6 illustrates how z(D) can be implemented with two bit logic adders.

We have

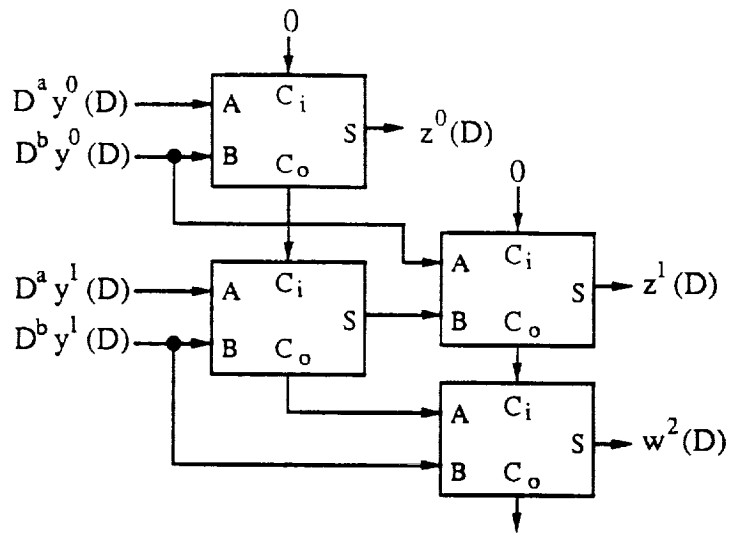

Figure 5.6: Two bit adders for rate 2/3 encoder.

$$z^1(D) = (D^a \oplus D^b)y^1(D) \oplus \overline{D^a y^0(D)} \cdot D^b y^0(D), \tag{5.29a}$$

$$z^2(D) = (D^a \oplus D^b)y^2(D) \oplus w^2(D), \tag{5.29b}$$

where

$$w^2(D) = \overline{D^a y^1(D)} \cdot D^b y^1(D) \oplus D^b y^0(D) \cdot ((D^a \oplus D^b)y^1(D) \cdot \overline{D^a y^0(D)} \oplus D^a y^0(D)). \tag{5.30}$$

The rate 2/3 IPCE is then

$$(D^a \oplus D^b)y^2(D) \oplus w^2(D) \oplus h_z^1 z^1(D) \oplus H^0(D)y^0(D) = 0(D). \tag{5.31}$$

Notice that in $w^2(D)$ we have non-linear terms involving $y^1(D)$. In an encoder implementation we always have $D^{-1}y^0(D)$, $y^0(D)$, and $Dy^0(D)$ available to form non-linear terms involving $y^0(D)$. This is not true for $y^1(D)$. Thus, delay terms need to be added to the encoder to produce the required non-linear terms involving $y^1(D)$. In other words, a minimal implementation is not possible and we have $m \geq \nu + a - b$ (the

inequality is due to the non-linear nature of these codes, similar to the rate 1/2 IPCE described in Section 5.2.2).

Since both $y^1(D)$ and $y^0(D)$ are outputs of a delay element in this case, the two least significant bits in a symbol are fixed in leaving a state, i.e., only two of the eight possible signal points can be chosen. However, there are four paths leaving a state, and so the minimum squared distance leaving a state is 0 (not δ_1^2 as desired). The minimum squared distance entering a state is δ_1^2 as expected. Therefore, the minimum free squared distance is lower bounded by δ_1^2 (the same minimum squared distance as the uncoded case).

Example 5.4 Rate 2/3 8PSK with two checked bits.

As in Example 5.2, we let $\nu = 3$, $a = 2$, and $b = 1$. From (5.31), the IPCE is

$$(D^2 \oplus D)y^2(D) \oplus w^2(D) \oplus h_z^1 z^1(D) \oplus (D^3 \oplus h_0^2 D^2 \oplus h_0^1 D \oplus 1)y^0(D) = 0(D). \quad (5.32)$$

An implementation of this encoder is shown in Figure 5.7. Note the extra delay element required to produce $D^{-1}y^1(D)$ and $y^1(D)$. Also note how $w^2(D)$ and $z^1(D)$ are obtained through the use of a times three modulo-8 multiplier and a modulo-8 addition element. In this case we have $m = \nu + a - b = 4$. The d_{free}^2 of both of the two possible codes (varying h_z^1) is 2, giving a 0 dB asymptotic coding gain compared to uncoded QPSK. $N_{free} = 0.25$ for both codes.

For an 8PSK signal set, it can be shown that d_{free}^2 will always be equal to $\delta_1^2 = 2$, no matter how large ν is. Let us assmune that $h_z^1 = 0$. For this case any code that has (5.31) as its PCE will always have the

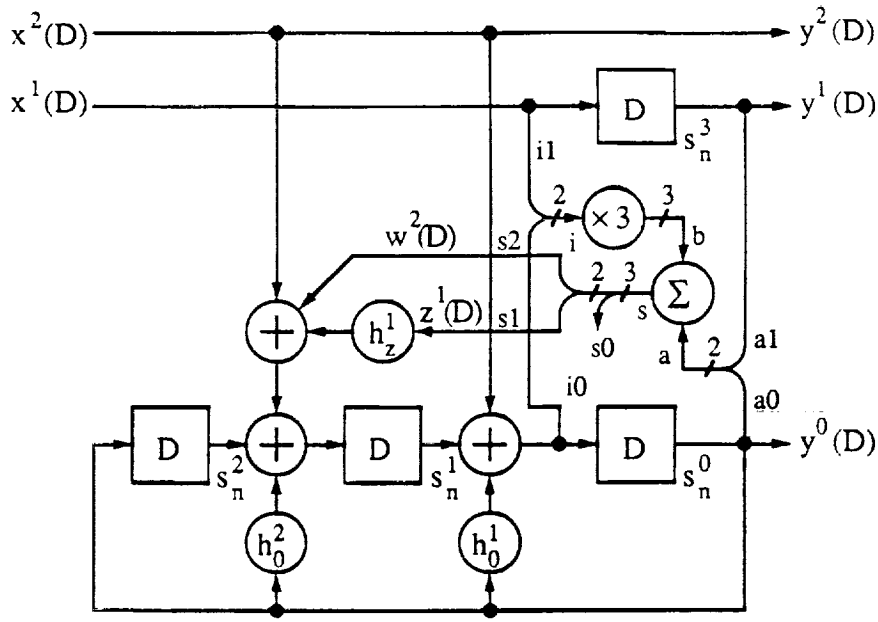

Figure 5.7: Rotationally invariant rate 2/3 8PSK encoder ($\tilde{k} = 2$, $m = 4$).

following sequences as valid code sequences.

$$y(D,i) = 6D^{i} + 6D^{i+1} + 6D^{i+2} + \cdots, \text{ for } -\infty < i < \infty.$$

This can be seen from (5.28a) where $z(D) = (D^{a} + 3D^{b})y(D,i) = 2D^{i+b} + \cdots + 2D^{i+a-1}$. Thus we have $y^{0}(D) = 0(D)$ and $z^{2}(D) = 0(D)$ which satisfies the GPCE (5.16).

The squared Euclidean distance between the sequences $y(D,i)$ and $y(D,i+1)$ is equal to 2. Since the minimum d^{2}_{free} is always at least 2, then $d^{2}_{free} = 2$ for $h^{1}_{z} = 0$ and any $H^{0}(D)$. For $h^{1}_{z} = 1$, a similar argument can be used (with $y(D,i) = 2D^{i} + 2D^{i+1} + 2D^{i+2} + \cdots$, for $-\infty < i < \infty$), to show that $d^{2}_{free} = 2$.

To try and overcome this problem we looked at other forms of $z(D)$. The only other practical $z(D)$ (with two delay terms) is $(3D^{a} \oplus D^{b})y(D)$, but this still gives $d^{2}_{free} = 2$ for all codes. As shown by Example 5.4, however, N_{free} is very small and this may compensate

C.3

for the small free distance at low signal to noise ratios. By letting $z(D) = (D^a + D^b + 2D^c)y(D)$, it may be possible to have $d^2_{free} > 2$.

5.2.5 Precoding and Postdecoding

The function of the precoder is to differentially encode (*precode*) the data so that phase rotations will not affect the data after decoding and differential decoding (*postdecoding*). This subject is covered in greater detail in [70] for the rate 1/2 IPCE.

Here we present a general method for obtaining a precoder and postdecoder as well as show how the precoder can be combined with the encoder. Let w(D) be the input to the precoder and x(D) be the input to the encoder, where

$$w(D) = \sum_{i=1}^{k} 2^{i-1} w^i(D) \text{ and } x(D) = \sum_{i=1}^{k} 2^{i-1} x^i(D). \tag{5.33}$$

From Chapter 2, the precoder equation is

$$2x(D) = 2Dx(D) + 2w(D) + (D + (M-1))y^0(D) \pmod{M}, \tag{5.34}$$

and the postdecoder equation is

$$2\hat{w}(D) = ((M-1)D + 1)(2\hat{x}(D) + \hat{y}^0(D)) \pmod{M}, \tag{5.35}$$

where the symbol ^ over a sequence indicates the decoder's estimate of an original sequence. Equations (5.34) and (5.35) can be implemented in a real system, with the precoder before the encoder and the postdecoder after the decoder. However, since the encoder is itself invariant to phase rotations, it is possible to combine the encoder and precoder, thus eliminating the need for a postdecoder. This allows the decoder to

make the best estimate of the transmitted sequence without having extra errors introduced by a postdecoder.

There are two steps in combining the precoder with the encoder. The first involves the precoding of $x^k(D)$. The remaining input bits are then precoded. Let the outputs of the systematic encoder binary delay elements (not counting any extra delay elements) be labeled $s_n^{\nu-1}$ to s_n^0 from left to right (or $s^{\nu-1}(D)$ to $s^0(D)$, respectively in sequence notation). By examining the systematic encoder structure, it can be shown that

$$y^0(D) = (D^b h_0^b \oplus D^{b-1} h_0^{b-1} \oplus \cdots \oplus D h_0^1) y^0(D) \oplus D^b y^k(D) \oplus D^b s^b(D), \quad (5.36)$$

where b is defined in Section 5.2.1. To determine the equation needed to precode $x^k(D)$, we rearrange (5.36) to obtain

$$s^b(D) = y^k(D) \oplus B(D) y^0(D), \quad (5.37)$$

where

$$B(D) = D^{-b} \oplus D^{-b+1} h_0^1 \oplus \cdots \oplus h_0^b. \quad (5.38)$$

In order that $w^k(D)$ is unaffected by a phase rotation, we now let our precoding equation be

$$x^k(D) = y^k(D) = w^k(D) \oplus s^b(D) \oplus E[B(D)] y^0(D). \quad (5.39)$$

Substituting (5.37) into (5.39) and assuming a noiseless channel we obtain the postdecoding equation (not used in practise since postdecoding occurs within the decoder) as

$$w^k(D) = (B(D) \oplus E[B(D)]) y^0(D). \quad (5.40)$$

After a phase rotation we have $y_r^0 = y^0 \oplus 1$. Thus,

$$\begin{aligned} w_r^k(D) &= (B(D) \oplus E[B(D)])y_r^0(D) \\ &= (B(D) \oplus E[B(D)])(y^0(D) \oplus 1(D)) \\ &= (B(D) \oplus E[B(D)])y^0(D) = w^k(D), \end{aligned}$$

which shows that the kth bit is correctly precoded. To precode the remaining input bits, we use a modified form of (5.34). For our systematic encoder implementation and $k \geq 2$, we have that

$$y^i(D) = D^{a-b}x^i(D);\ 1 \leq i \leq k-1,\ a-b \geq 1. \tag{5.41}$$

These delayed sequences of $x^i(D)$ for $1 \leq i \leq k-1$ are used by the precoder as follows. Let

$$w'(D) = \sum_{i=1}^{k-1} 2^{i-1}w^i(D) \text{ and } x'(D) = \sum_{i=1}^{k-1} 2^{i-1}x^i(D). \tag{5.42}$$

Our precoder equation now becomes

$$2x'(D) = 2Dx'(D) + 2w'(D) + (D + (M/2-1))y^0(D) \pmod{M/2}. \tag{5.43}$$

All the sequences that $x'(D)$ depends on in (5.43) are available in the encoder, thus allowing an implementation without any increase in the number of delay elements.

<u>Example 5.5</u> Rate 2/3 8PSK

Let $\nu = 3$, $m = 4$, $a = 2$, $b = 1$, $h_z^1 = 0$, and $H^0(D) = D^3 \oplus D \oplus 1$. To precode $x^2(D)$, first note that $B(D) = 1 \oplus D^{-1}$. From (5.39) we have

$$x^2(D) = w^2(D) \oplus s^1(D). \tag{5.44}$$

From (5.43) it follows that

$$2x^1(D) = 2Dx^1(D) + 2w^1(D) + (D + 3)y^0(D) \pmod 4. \tag{5.45}$$

Expressing (5.45) in binary notation, it can be shown that

$$x^1(D) = Dx^1(D) \oplus w^1(D) \oplus \overline{Dy^0(D)}\cdot y^0(D). \tag{5.46}$$

Figure 5.8 illustrates an implementation of the complete precoder/encoder.

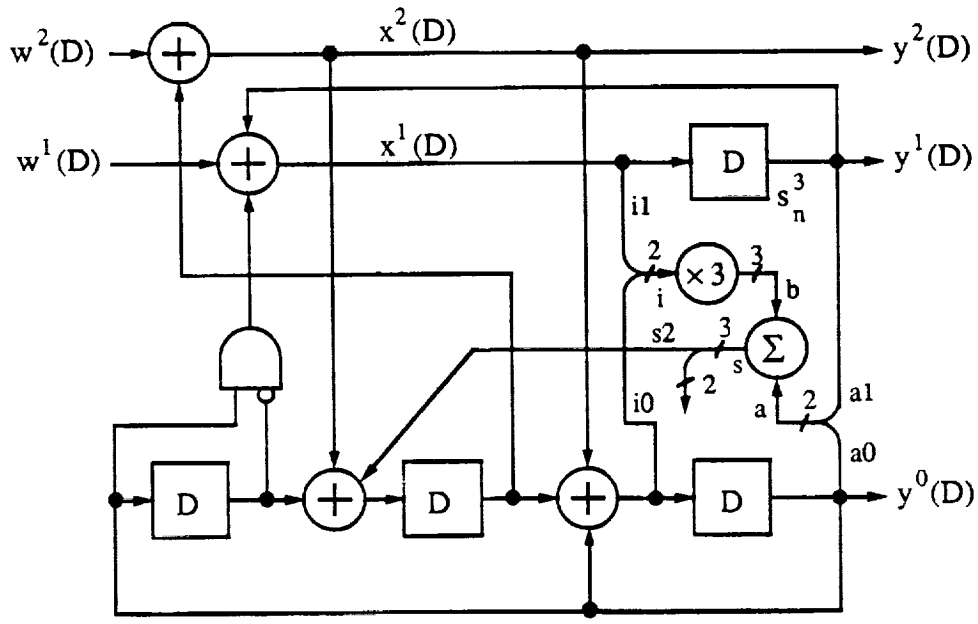

Figure 5.8: Rotationally invariant rate 2/3 8PSK encoder ($\tilde{k} = 2$, $m = 4$) combined with differential encoder (precoder).

5.3 Systematic Code Search

Our criteria for finding good rotationally invariant codes are based on maximizing the asymptotic coding gain (γ) and minimizing the probability of an event error (Pe). γ and Pe can be expressed in terms

of the minimum free squared Euclidean distance (d^2_{free}) and the number of nearest neighbors (N_{free}), i.e.,

$$\gamma = 10 \log_{10}\left(\frac{d^2_{free}}{\delta^2_1}\right) \text{ (dB) and } Pe \gtrsim N_{free} Q\left(\sqrt{\frac{d^2_{free} K E_b}{2 \quad N_0}}\right), \tag{5.47}$$

where d^2_{free} is normalized so that the average energy of the signal set is equal to one, δ^2_1 is the minimum squared distance of the uncoded signal set, K is the number of information bits transmitted per 2-D symbol, E_b/N_0 is the energy per information bit to one sided noise density ratio, and $Q(\cdot)$ is the Gaussian error probability function. The expression for Pe approaches equality as E_b/N_0 becomes large. For small values of E_b/N_0, additional terms in the distance spectrum of the code must be taken into account.

Therefore, the code search is designed to find all the codes which have the maximum d^2_{free} (to maximize γ), and from those codes to select the code with the smallest N_{free} (to minimize Pe). For some classes of trellis codes, all the codes have the same d^2_{free} and N_{free}. In this case we find the codes with the largest next nearest squared Euclidean distance (d^2_{next}), and from those codes the code with the smallest number of next nearest neighbors (N_{next}) is selected.

Since the codes are non-linear, we cannot use algorithms such as the bidirectional search algorithm to find d^2_{free} (or d^2_{next}). Instead, our algorithm must look at all pairs of paths produced in the trellis, not just the path pairs which have the all zero path as one of the paths. The Double Dynamic Programming Algorithm (DDPA) from Ungerboeck [66] is an efficient algorithm to achieve this. This algorithm uses the

properties of the trellis in order to reduce the computation time.

To find N_{free} (or N_{next}), our algorithm is designed to find the number of nearest neighbors for paths starting at state I and ending at state J of the trellis ($N_{free}(I,J)$). We compute $N_{free}(I,J)$ for all combinations of I and J and then take the average to determine N_{free}. The computation of $N_{free}(I,J)$ is determined from two different characteristics of the trellis. The first is the number of mergers at state J of pairs of paths of length L (which have a distance of d^2_{free} between them) diverging from state I ($N_m(I,J,L)$). The second is the number of paths between state I and state J of length L ($N_p(I,J,L)$). The length L is the number of 2-D symbols between states I and J.

Knowing the number of mergers and the number of paths, the number of nearest neighbors of paths of length L between states I and J ($N_{free}(I,J,L)$) is

$$N_{free}(I,J,L) = 2N_m(I,J,L)/N_p(I,J,L). \tag{5.48}$$

Equation (5.48) is derived from the fact that each merger indicates that there is one nearest neighbor for one path and another nearest neighbor for the other path. Since the number of nearest neighbors is defined as the total number of nearest neighbors seen by all paths divided by the total number of paths, (5.48) results.

$N_{free}(I,J)$ is calculated by summing $N_{free}(I,J,L)$ over all L up to the length where the distance between paths from state I and J is greater than d^2_{free}. Our implementation of this algorithm uses the structure of the trellis (similar to the DDPA) to reduce the number of computations required to find N_{free}.

Determining N_{free} for these codes is not simple and requires many

computations. Therefore, we have limited the code complexity (the encoder memory (m) plus the number of checked input bits ($\tilde{k}$)) to a maximum of eight. Also, the code search was limited to those codes where $m = \nu$ except the rate 2/3 8PSK codes with two checked bits where $m = \nu + 1$ (where we limit codes to having $a-b = 1$).

When all the sequences in a code are reversed, the new code for these sequences has the same d^2_{free} as the original code. Thus, we have used the bit-reversal technique [65] to reduce the number of codes that must be examined.

5.4 Results and Discussion

Table 5.1 lists the best rotationally invariant rate 1/2 codes for QPSK modulation. Note that d^2_{free} is twice as large as normally given for these codes since we have normalized the energy of the QPSK signal set to one. For $\nu = 3$ to 5, d^2_{free} is 2 less and for $\nu = 6$ and 7, d^2_{free} is 4 less than the best linear codes [56] (some of which are only 360° invariant). The N_{free}'s for the codes in Table 5.1 are quite small, ranging from 3 to 10 times smaller than the N_{free}'s of the best linear codes [56]. This reduction in N_{free} partly compensates for the smaller d^2_{free} of these codes.

In Tables 5.2 and 5.3, our results for rate 2/3 8PSK are presented. For $\tilde{k} = 1$, the parallel transitions result in $d^2_{free} = 4$ and $N_{free} = 1$ for all codes (Table 5.2). However, the next nearest distances increase with increasing ν. For $\nu = 3$ to 6, d^2_{next} is the same as d^2_{free} for the best linear codes [56]. For $\nu = 7$, d^2_{next} is 0.37 dB larger than d^2_{free} for the best $\nu = 7$ linear code [56]. The $\nu = 6$ code

TABLE 5.1

ROTATIONALLY INVARIANT RATE 1/2 QPSK CODES

$K = 1.0$ bit/sym, $d_u^2 = 4$, $N_u = 1$ (BPSK).

				90° Inv.		180° Inv.		360° Inv.		Inv.
ν	$\tilde{k}$	h^1	h^0	d^2_{free}	N_{free}	d^2_{free}	N_{free}	d^2_{free}	N_{free}	γ (dB)
3	1	06	13	10	0.250	12	2	12	1	3.98
4	1	06	23	12	0.333	12	1	14	2	4.77
5	1	30	45	14	0.667	16	2	16	1	5.44
6	1	050	105	16	1.612	20	11	-	-	6.02
7	1	120	253	16	0.201	20	2	20	1	6.02

may be an alternative to Viterbi's pragmatic rate 2/3 code (which is 90° invariant) [71]. It should perform as well or better than Viterbi's code with the added advantage that synchronization of the receiver with the signal set is not required. In addition, N_{next} is less than one, as it is for all of the nonlinear codes listed in this table.

TABLE 5.2

ROTATIONALLY INVARIANT RATE 2/3 8PSK CODES

WITH ONE CHECKED BIT

$K = 2.0$ bit/sym, $d_u^2 = 2.0$, $N_u = 2$ (QPSK).

				45° Inv.				180° Inv.		360° Inv.		Inv.
ν	$\tilde{k}$	h^1	h^0	d^2_{free}	N_{free}	d^2_{next}	N_{next}	d^2_{free}	N_{free}	d^2_{free}	N_{free}	γ (dB)
3	1	06	13	4.0	1.0	4.586	0.250	4.586	2	-	-	3.01
4	1	06	23	4.0	1.0	5.172	0.333	5.172	4	5.172	2.25	3.01
5	1	30	45	4.0	1.0	5.757	0.417	5.172	0.25	5.757	2	3.01
6	1	060	105	4.0	1.0	6.343	0.467	6.343	3.25	-	-	3.01
7	1	030	203	4.0	1.0	7.172	0.333	6.343	0.125	6.586	0.5	3.01

TABLE 5.3

ROTATIONALLY INVARIANT RATE 2/3 8PSK CODES

WITH TWO CHECKED BITS

$K = 2.0$ bit/sym, $d_u^2 = 2.0$, $N_u = 2$ (QPSK).

ν	m	$\tilde{k}$	h^2	h^1	h^0	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
3	4	2	06	00	13	2.0	1/4	4.0	0.375	0.00
4	5	2	06	00	23	2.0	1/8	4.0	0.333	0.00
5	6	2	06	00	43	2.0	1/16	4.0	0.167	0.00

The results for the rate 2/3 codes with $\tilde{k} = 2$ appear disappointing. The aim of going to two checked bits was to remove the parallel transitions in the hope of increasing d^2_{free} beyond 4. However, as shown in Section 5.2.4, the reverse happened, and d^2_{free} is only 2 when $\tilde{k} = 2$. The number of nearest neighbors though, divides by two with each increase in ν (beginning at 1/4 with $\nu = 3$). Also, the next nearest distance is 4 with N_{next} being smaller than N_{free} for the $\tilde{k} = 1$ codes. Thus, it may be that these codes perform better than the $\tilde{k} = 1$ codes at moderate E_b/N_0 ratios due to a lower number of nearest neighbors. Another strange aspect of these codes is that all the properties examined (d^2_{free}, N_{free}, d^2_{next}, and N_{next}) were the same for every code. The reason that $h^1 = 00$ for all three codes is that $h^1_z = 0$ (if $h^1_z = 1$ then h^1 would be non-zero).

The rate 3/4 16PSK codes in Table 5.4 follow a pattern similar to the rate 1/2 QPSK codes. That is, d^2_{free} is usually less than the best linear codes, but N_{free} is smaller [56]. The exception is the $\nu = 7$ code where d^2_{free} occurs along parallel transitions. However, d^2_{next} and N_{next} follow the familiar pattern.

TABLE 5.4

ROTATIONALLY INVARIANT RATE 3/4 16PSK CODES

$K = 3.0$ bit/sym, $d_u^2 = 0.586$, $N_u = 2$ (8PSK).

				22.5° Inv.				45° Inv.		90° Inv.		Inv.
ν	$\tilde{k}$	h^1	h^0	d^2_{free}	N_{free}	d^2_{next}	N_{next}	d^2_{free}	N_{free}	d^2_{free}	N_{free}	γ (dB)
3	1	06	13	1.324	0.250	-	-	1.476	8	1.476	4	3.54
4	1	06	23	1.476	0.333	-	-	1.476	4	1.628	4	4.01
5	1	30	45	1.628	0.417	-	-	1.781	8	1.910	8	4.44
6	1	060	105	1.781	0.467	-	-	2.0	2	2.0	2	4.83
7	1	030	203	2.0	2.0	2.062	0.333	2.0	2	-	-	5.33

In Table 5.5 we give the results for rate 3/4 16QAM. Unlike the codes for MPSK modulation, most of the codes listed have the same d^2_{free} as the best linear code. Only the memory 4 code has a smaller d^2_{free}. Also the N_{free}'s are less than the linear codes given in Chapter 3. For the codes where the d^2_{free}'s are the same, this indicates that the best rotationally invariant codes should be better (surprisingly) than the best linear codes. For the $\nu = 4$ code, N_{free} is 38 times smaller than the best linear code. This may compensate for the reduction in d^2_{free} for this code.

In Table 5.6, rotationally invariant codes for rate 4/5 32CROSS, rate 5/6 64CIRC, rate 6/7 128CROSS, rate 7/8 256CIRC and rate 8/9 512STAR are given, respectively. The signal sets for these codes are described in Chapter 3. In the code search all these signal sets produced the same codes, although with different N_{free}'s. Note that the N_{free}'s do not increase monotonicly with the size of the signal set (for linear codes N_{free} does increase monotonicly). This is probably due to the unusual non-linear structure of the encoder.

TABLE 5.5

ROTATIONALLY INVARIANT RATE 3/4 16QAM CODES

$K = 3.0$ bit/sym, $d_u^{2*} = 2$, $N_u = 2.25$ (8AMPM).

					90° Inv.		180° Inv.		360° Inv.		Inv.
ν	$\tilde{k}$	h^2	h^1	h^0	d^{2*}_{free}	N_{free}	d^{2*}_{free}	N_{free}	d^{2*}_{free}	N_{free}	γ (dB)
3	2	02	06	13	5	0.875	5	3.781	5	3.656	3.98
4	2	06	14	23	5	0.25	6	9.594	6	9.156	3.98
5	2	10	30	45	6	0.547	6	1.891	6	1.812	4.77
6	2	020	014	103	7	2.668	7	6.172	7	4.828	5.44

*Divide by 2.5 for normalized d^2_{free}.

TABLE 5.6

ROTATIONALLY INVARIANT QAM CODES

						32	64	128	256	512	
ν	$\tilde{k}$	h^2	h^1	h^0	d^2_{free}	N_{free}	N_{free}	N_{free}	N_{free}	N_{free}	γ (dB)
3	2	02	06	13	5	4.5	3.350	2.847	4.813	4.329	3.98
4	2	06	14	23	5	1.292	0.957	0.811	1.370	1.230	3.98
5	2	10	30	45	6	2.875	2.135	1.817	3.266	2.929	4.77
6	2	020	014	103	7	19.992	14.109	11.695	23.650	20.885	5.44

It is interesting comparing the eight state invariant QAM code proposed in [1] which was accepted as the V.32 modem standard with the equivalent code in Table 5.6. The V.32 code has 9.124 nearest neighbors, double that of the code we have found. Also, the V.32 code requires a seperate precoder, whereas our code can combine the precoder with the encoder, eliminating the need for a seperate postdecoder. The reason for this difference in N_{free} most likely lies in the extra complexity of the V.32 code (which requires two AND gates). Complexity also seems to have an affect on our codes as well. Nearly all codes

have a-b = 1 and no codes were found that had a-b > 2. It appears that the more non-linear a code is, the worse it will perform. The very non-linear rate 2/3 8PSK codes with two checked bits performed very poorly.

5.5 Conclusions

A systematic method of obtaining rotationally invariant trellis codes for a variety of signal sets has been presented. Since codes based on linear parity check equations are not invariant (for signal sets with more than two points) an alternative general parity check equation was found. This GPCE allows the construction of invariant codes for signal sets that are "naturally" mapped.

A general method of combining the precoder with a systematic encoder without increasing the encoder memory was also given. This eliminates the need for a postdecoder, since the precoder is part the encoder trellis.

When a signal set has 90° rotational symmetries or only one input bit is checked by the encoder, the invariant parity check equation is relatively simple, with only one non-linear term. The best codes for the QPSK and 16PSK signal sets were found to have smaller free distances when compared to the corresponding linear codes. However, their low number of nearest neighbors may make up for this loss at moderate E_b/N_0 ratios. The QAM codes were found to be very good. Most of these codes had the same free distance as the best linear codes as well as a smaller N_{free}.

The results for 8PSK with two checked bits seem to be

disppointing. These codes have 0 dB asymptotic coding gain. However, their low N_{free} may be useful at small E_b/N_0 ratios. The 8PSK codes with one checked bit all have 3 dB asymptotic coding gains and so are more likely to be of practical interest.

CHAPTER SIX
CONCLUSIONS

In this dissertation we have explored two important aspects of trellis coding. These are the use of multidimensional signal sets to find new trellis codes and the construction of codes that are rotationally invariant. Trellis codes that use multi-D signals for both Phase Shift Keyed (PSK) and Quadrature Amplitude Modulation (QAM) signal sets were found. Many of these codes were found to be rotationally invariant.

However, trellis codes using 2-D signal sets and linear convolutional encoders cannot be made rotationally invariant. A general Parity Check Equation (PCE) was found which produced non-linear trellis codes that are rotationally invariant.

The key to finding good trellis codes that use multi-D signal sets is in the construction of the multi-D signal set mapper. The mapper takes the n coded bits from the convolutional encoder and maps them into L 2-D signals. To do this effectively, the multi-D signal set must be partitioned. Due to the large number of signal points in the multi-D signal set, doing the partitioning by hand becomes impractical.

The use of length L block codes, the partitoning of the basic 2-D signal set, and the concept of subcodes and cosets led to a method of easily partitioning a multi-D signal set. By adding the cosets either modulo-M/modulo-2 for MPSK signal sets and modulo-4/modulo-2 for QAM

signal sets, it became possible to find good rotationally invariant codes using multi-D signal sets.

These codes provide large coding gains while allowing fractional values of K and high decoding speeds, as demonstrated by a serial implementation of a 16 state, 2.33 bit/sym, 6D-8PSK code. In our code search, we have presented the best codes (in terms of d^2_{free} and N_{free}) for each of the two (and sometimes more) possible phase transparencies of a coded system. Thus, the code most suited to the communication requirement can be found.

With the general PCE, the key to finding good rotationally invariant codes with 2-D signal sets was in splitting the parity check equation into two equations. One equation used modulo-M arithmetic, while the other equation used modulo-2 arithmetic. By writing the modulo-M equation in terms of its binary sequences and combining these sequences with the modulo-2 equation, an invariant PCE (with the required non-linear terms) can be found.

Using the invariant PCE, rotationally invariant codes for 2-D MPSK and QAM signal sets have been found. For MPSK signal sets, the code performance in terms of d^2_{free} was found to be inferior to trellis codes using linear convolutional encoders. These codes, however, should still be useful in channels where the phase needs to be corrected often and a constant signal amplitude is required. The mobile satellite channel is a good example.

Unfortuantely, we have not been able to find rotationally invariant rate 2/3 8PSK codes with $d^2_{free} > 4.0$. If such codes could be found, their non-parallel transitions, large free distance, constant amplitude, bandwidth efficiency, and rotational invariance would make

them especially well suited to the fading channel enviroment of mobile satellite systems.

Surprisingly, the non-linear rotationally invariant 2-D QAM codes performed better than the best linear codes. Nearly all the non-linear codes had the same d^2_{free} and a smaller N_{free} compared to the best linear codes. This better performance and the rotational invariance makes these codes very attractive for telephone modems and microwave-links where high values of K are required.

For the rotational invariant codes, a method of combining a differential encoder with the convolutional encoder is given. Thus, the receiver does not require a seperate differential decoder. This may lead to a small increase in performance since there is no postdecoder making "hard decisions".

Codes with only moderate complexity have been found. It is expected that only small increases in coding gain can be obtained with more complex codes. This is due to the fact that the existing codes are already fairly close to the theoretical Shannon limit. However, we do not expect this to impede a further search for more complex codes. The increasing performance capabilities of computers (to find the codes) and miniaturization of integrated circuits (for implementing the decoders) will insure that more complex codes will eventually be used. There may even be machines that are constructed specifically to search for extremely complex codes.

REFERENCES

[1] AT&T Information Systems, "A trellis coded modulation scheme that includes differential encoding for 9600 bit/sec, full duplex, two-wire modems," CCITT SG XVII Contribution COM XVII, no. D159, Aug. 1983.

[2] Benedetto, S., M. A. Marsan, G. Albertengo, and E. Giachin, "Combined coding and modulation: Theory and applications," *IEEE Trans. Inform. Theory*, vol. 34, pp. 223-236, Feb. 1988.

[3] Berg, L., M. Turgeon, A. Fung, and P. McLane, "Design procedure for optimum or rotationally invariant trellis codes," *Canadian Conf. Elec. and Comp. Eng.*, Vancouver, BC, Nov. 1988.

[4] Bertelsmeier, M. and G. Komp, "Trellis-coded 8PSK with embedded QPSK," *IEEE J. Sel. Areas Commun.*, vol. 7, pp. 1296-1306, Dec. 1989.

[5] Biglieri, E. and M. Elia, "Multidimensional modulation and coding," *IEEE Trans. Inform. Theory*, vol. 34, pp. 803-809, Jul. 1988.

[6] Buz, R., "Design and performance analysis of multi-dimensional trellis coded modulation," M.Sc. Thesis, Dept. of Elec. Eng., Queen's Univ., Kingston, Ont., Canada, Feb. 1989.

[7] Calderbank, A. R. and J. E. Mazo, "A new description of trellis codes," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 784-791, Nov. 1984.

[8] Calderbank, A. R. and N. J. A. Sloane, "Four-dimensional modulation with an eight state trellis code," *AT&T Tech. J.*, vol. 64, no. 5, pp. 1005-1018, May-Jun. 1985.

[9] Calderbank, A. R. and N. J. A. Sloane, "An eight-dimensional trellis code," *Proc. IEEE*, vol. 74, pp. 757-759, May 1986.

[10] Calderbank, A. R. and N. J. A. Sloane, "New trellis codes based on lattices and cosets," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 177-195, Mar. 1987.

[11] Calderbank, A. R., "Multilevel codes and multistage decoding," *IEEE Trans. Commun.*, vol. 37, pp. 222-229, Mar. 1989.

[12] Chouly, A. and H. Sari, "Six-dimensional trellis-coded modula-

tion," *GLOBECOM'89 Rec.*, pp. 1522-1527, Dallas, TX, Nov. 1989.

[13] Cusack, E. L., "Error control codes for QAM signalling," *IEE Electron. Lett.*, vol. 20, no. 2, pp. 62-63, 19 Jan. 1984.

[14] Deng, R. H. and D. J. Costello, Jr., "High rate concatenated coding systems using multi-dimensional bandwidth efficient trellis inner codes," *IEEE Trans. Commun.*, vol. 37, pp. 1091-1096, Oct. 1989.

[15] Divsalar, D. and M. K. Simon, "Multiple trellis coded modulation (MTCM)," *IEEE Trans. Commun.*, vol. COM-36, pp. 410-419, Apr. 1988.

[16] Divsalar, D. and M. K. Simon, "The design of trellis coded MPSK for fading channels: Set partitioning for optimum code design," *IEEE Trans. Commun.*, vol. 36, pp. 1013-1021, Sep. 1988.

[17] Fang, R. and W. Lee, "Four-dimensionally coded PSK systems for combatting effects of severe ISI and CCI," *GLOBECOM'83 Rec.*, San Diego, CA, pp. 1032-1038, Nov.-Dec. 1983.

[18] Fang, R., M. Kappes, and S. Miller, "Rate 8/9 coded 8-PSK system for downlink applications," *Advanced Modulation and Coding Conf.*, NASA Lewis Research Center, Jun. 1989.

[19] Fettweis, G., "New rotationally invariant trellis codes for 8PSK modulation," *1989 Int. Conf. Commun. Rec.*, pp. 1388-1392, Boston, MA, Jun. 1989.

[20] Forney, G. D., Jr., "Convolutional codes I: Algebraic structure," *IEEE Trans. Inform. Theory*, vol. IT-16, pp. 720-738, Nov. 1970.

[21] Forney, G. D., Jr., R. G. Gallager, G. R. Lang, F. M. Longstaff, and S. U. Qureshi, "Efficient modulation for band-limited channels," *IEEE J. Sel. Areas Commun.*, vol. SAC-2, pp. 632-647, Sep. 1984.

[22] Forney, G. D., Jr., "Coset codes I: Introduction and geometrical classification," *IEEE Trans. Inform. Theory*, vol. 34, pp. 1123-1151, Sep. 1988, Part II.

[23] Forney, G. D., Jr., "Coset codes II: Binary lattices and related codes," *IEEE Trans. Inform. Theory*, vol. 34, pp. 1152-1187, Sep. 1988, Part II.

[24] Forney, G. D., Jr. and L.-F. Wei, "Multidimensional constellations -Part I: Introduction, figures of merit, and generalized cross constellations," *IEEE J. Sel. Areas Commun.*, vol. 7, pp. 877-892, Aug. 1989.

[25] Forney, G. D., Jr., "Multidimensional constellations-Part II: Voronoi constellations," *IEEE J. Sel. Areas Commun.*, vol. 7, pp. 941-958, Aug. 1989.

[26] Gersho, A. and V. B. Lawrence, "Multidimensional signal constellations for voiceband data transmission," *IEEE J. Sel. Areas Commun.*, vol. SAC-2, pp. 687-702, Sep. 1984.

[27] Ginzburg, V. V., "Multidimensional signals for a continuous channel," *Problemy Peredachi Informatsii [Probl. Inform. Transmission]*, vol. 20, no. 1, pp. 28-46, Jan-Mar. 1984 (in Russian).

[28] Gray, P. K., I. S. Morrison, and W. G. Cowley, "The development of a 45 Mbit/s modem/codec," *Proc. IREECON'89*, pp. 942-945, Melbourne, Australia, Sep. 1989.

[29] Hekstra, A. P., "Use of two's complement arithmetic as alternative to metric rescaling in Viterbi decoders," *IEEE Trans. Commun.*, vol. 37, pp. 1220-1222, Nov. 1989.

[30] Hemmati, F. and R. J. F. Fang, "Low complexity coding methods for high data rate channels," *COMSAT Tech. Rev.*, vol. 16, pp. 425-447, Fall 1986.

[31] Huber, K., "Combined coding and modulation using block codes," *IEE Electron. Lett.*, vol. 25, pp. 1130-1131, 17 Aug. 1989.

[32] IBM Europe, "Trellis-coded modulation schemes with 8-state systematic encoder and 90° symmetry for use in data modems transmitting 3-7 bits per modulation interval," CCITT SG XVII Contribution COM XVII, no. D180, Oct. 1983.

[33] Imai, H. and S. Hirakawa, "A new multilevel coding method using error correcting codes," *IEEE Trans. Inform. Theory*, vol. IT-23, pp. 371-377, May, 1977.

[34] Kasami, T., T. Takata, T. Fujiwara, and S. Lin, "On linear structure and phase rotation invariant properties of block 2^l-PSK modulation codes," *IEEE Trans. Inform. Theory*, to appear.

[35] Kasami, T., T. Takata, T. Fujiwara, and S. Lin, "On multi-level block modulation codes," submitted to *IEEE Trans. Inform. Theory*, 1989.

[36] Kschichang, F. P., P. G. de Buda, and S. Pasupathy, "Block coset codes for M-ary phase shift keying," *IEEE J. Sel. Areas Commun.*, vol. 7, pp. 900-913, Aug. 1989.

[37] Lafanachère, A. and D. J. Costello, Jr., "Multidimensional coded PSK systems using unit-memory trellis codes," *Proc. 23rd Allerton Conf. Commun., Cont., and Comput.*, pp. 428-429, Monticello, IL, Sep. 1985.

[38] Larson, K. J., "Comments on 'An efficient algorithm for computing free distance'," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 437-439, May 1972.

[39] Lin, D. W., "Wide-band digital subscriber access with multidimensional block modulation and decision-feedback equalization," *IEEE J. Sel. Areas Commun.*, vol. 7, pp. 996-1005, Aug. 1989.

[40] Massey, J. L. and T. Mittelholzer, "Convolutional codes over rings," *Proc. 4th Joint Swedish-USSR Inr. Workshop Inform. Theory*, Gotland, Sweden, Aug.-Sep. 1989.

[41] Michie, A. and M. J. Miller, "Forward error correction using block coded multidimensional signals and soft decision decoding," *Proc. IREECON'87*, pp. 242-244, Sydney, Australia, Sep. 1987.

[42] Nowack, J. M., "On the decoding of bandwidth-efficient codes," Ph.D. dissertation, Univ. of Notre Dame, Notre Dame, IN, May 1990.

[43] Odenwalder, J. P., "Optimal decoding of convolutional codes," Ph.D. dissertation, Univ. of California, Los Angeles, CA, 1970.

[44] Oerder, M., "Rotationally invariant trellis codes for MPSK modulation," *1985 Int. Commun. Conf. Rec.*, pp. 552-556, Chicago, IL, Jun. 1985.

[45] Perez, L. C. and D.J. Costello, Jr., "On the performance of multi-dimensional phase modulated trellis codes," *Proc. 27th Allerton Conf. Commun., Cont., and Comput.*, Monticello, IL, Sep. 1989.

[46] Perez, L., "On the performance of multi-dimensional phase modulated trellis codes," NASA Tech. Report #89-10-02, Oct. 1989.

[47] Pietrobon, S. S., "Discrete implementation of a NASA planetary standard Viterbi decoder," *Proc. IREECON'87*, pp. 249-252, Sydney, Australia, Sep. 1987.

[48] Pietrobon, S. S., "Rotationally invariant convolutional codes for MPSK modulation and implementation of Viterbi decoders," M.Eng. Thesis, School of Electron. Eng., South Australian Inst. of Technol., Adelaide, Australia, Jun. 1988.

[49] Pietrobon, S. S. and G. Ungerboeck, "Rotationally invariant rate 1/2 convolutional codes for QPSK modulation," *Proc. IEEE Int. Symp. Inform. Theory*, pp. 92-93, Kobe, Japan, Jun. 1988.

[50] Pietrobon, S. S., D. J. Costello, Jr., G. Ungerboeck, R. H. Deng, and A. Lafanechère, "Channel coding with multi-dimensional trellis coded phase modulation," *Proc. IEEE Int. Symp. Inform. Theory.*, p. 187, Kobe, Japan, Jun. 1988.

[51] Pietrobon, S. S., G. Ungerboeck, and D. J. Costello, Jr., "Trellis coded phase modulation," *Proc. Int. Forum Inform. Theory and Applicat.*, pp. LW3.1-LW3.11, Tokyo, Japan, Jul. 1988.

[52] Pietrobon, S. S., D. J. Costello, Jr., and G. Ungerboeck, "A

general parity check equation for rotationally invariant trellis codes," *IEEE Inform. Theory Workshop*, Cornell Univ., Ithaca, NY, Jun. 1989.

[53] Pietrobon, S. S., A. N. McLean, I. S. Morrison, and P. K. Gray, "Construction and testing of a flexible Viterbi codec," *Proc. IREECON'89*, pp. 702-705, Melbourne, Australia, Sep. 1989.

[54] Pietrobon, S. S., D. J. Costello, Jr., and G. Ungerboeck, "Rotationally invariant trellis codes," *Proc. IEEE Int. Symp. on Inform. Theory*, pp. 107-108, San Diego, CA, Jan. 1990.

[55] Pietrobon, S. S. and D. J. Costello, Jr., "Trellis coding using multi-dimensional QAM signal sets," *Proc. IEEE Int. Symp. Inform. Theory*, p. 108, San Diego, CA, Jan. 1990.

[56] Pietrobon, S. S., R. H. Deng, A. Lafanechère, G. Ungerboeck, and D. J. Costello, Jr., "Trellis-coded multidimensional phase modulation," *IEEE Trans. Inform. Theory*, vol. 36, pp. 63-89, Jan. 1990.

[57] Porath, J. E., "Algorithms for converting convolutional codes from feedback to feedforward form and vice versa," *IEE Electron. Lett.*, vol. 25, pp. 1008-1009, 20 Jul. 1989.

[58] Potter, G. J. and D. P. Taylor, "An approach to Ungerboeck coding for rectangular signal sets," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 285-290, Mar. 1987.

[59] Pottie, G. J. and D. P. Taylor, "Multilevel codes based on partitioning," *IEEE Trans. Inform. Theory*, vol. 35, pp. 87-98, Jan. 1989.

[60] Rouanne, M. and D. J. Costello, Jr., "A lower bound on the minimum Euclidean distance of trellis coded modulation schemes," *IEEE Trans. Inform. Theory*, vol. 34, pp. 1011-1020, Sep. 1988, Part I.

[61] Sayegh, S. I., "A class of optimum block codes in signal space," *IEEE Trans. Commun.*, vol. COM-34, pp. 1043-1045, Oct. 1986.

[62] Shannon, C. E., "A mathematical theory of communication," *Bell Sys. Tech. J.*, vol. 27, pp. 379-423, Jul. 1948 and pp. 623-656, Oct. 1948.

[63] Tanner, R. M., "Algebraic construction of large Euclidean distance combined coding/modulation systems," submitted to *IEEE Trans. Inform. Theory.*

[64] Tretter, S. A., "An eight-dimensional 64-state trellis code for transmitting 4 bits per 2-D symbol," *IEEE J. Sel. Areas Commun.*, vol. 7, pp. 1392-1395, Dec. 1989.

[65] Ungerboeck, G., "Channel coding with multilevel/phase signals," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 55-67, Jan. 1982.

[66] Ungerboeck, G., private commun., Jan. 1986.

[67] Ungerboeck, G., "Trellis-coded modulation with redundant signal sets part I: Introduction," *IEEE Commun. Mag.*, vol. 25, pp. 5-11, Feb. 1987.

[68] Ungerboeck, G., "Trellis-coded modulation with redundant signal sets part II: State of the art," *IEEE Commun. Mag.*, vol. 25, pp. 12-21, Feb. 1987.

[69] Ungerboeck, G., "Trellis coded phase modulation," *IEEE Commun. Theory Workshop*, Howey-in-the-Hills, FL, Apr. 1987.

[70] Ungerboeck, G. and S. S. Pietrobon, "Codes for QPSK modulation under 90° rotation," *Proc. Mobile Satellite Conf.*, pp. 277-282, San Diego, CA, May 1988.

[71] Viterbi, A. J., J. K. Wolf, E. Zehavi, and R. Padovani, "A pragmatic approach to trellis-coded modulation," *IEEE Commun. Mag.*, Vol. 27, pp. 11-19, Jul. 1989.

[72] Wei, L.-F., "Rotationally invariant convolutional channel coding with expanded signal space - Part I: 180°," *IEEE J. Sel. Areas Commun.*, vol. SAC-2, pp. 659-671, Sep. 1984.

[73] Wei, L.-F., "Rotationally invariant convolutional channel coding with expanded signal space - Part II: Nonlinear codes," *IEEE J. Select. Areas Commun.*, vol. SAC-2, pp. 672-686, Sep. 1984.

[74] Wei, L.-F., "Trellis-coded modulation with multidimensional constellations," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 483-501, Jul. 1987.

[75] Wei, L.-F., "Rotationally invariant trellis-coded modulations with multidimensional M-PSK," *IEEE J. Sel. Areas Commun.*, vol. 7, pp. 1281-1295, Dec. 1989.

[76] Wilson, S. G., "Rate 5/6 trellis-coded 8PSK," *IEEE Trans. Commun.*, vol. COM-34, pp. 1045-1049, Oct. 1986.

[77] Wu, J., X. Zhu, and D. Lu, "Trellis-block coded modulation with multiple MPSK," *Proc. Int. Workshop on Inform. Theory*, pp. E7.1-E7.4, Beijing, China, Jul. 1988.

[78] Yamaguchi, K. and H. Imai, "Highly reliable multilevel channel coding system using binary convolutional codes," *IEE Electron. Lett.*, vol. 23, pp. 939-941, 27 Aug. 1987.

[79] Zehavi, E. and J. K. Wolf, "On the performance evaluation of trellis codes," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 196-202, Mar. 1987.

[80] Zhang, L. and B. Vucetic, "Bandwidth efficient block codes for

Rayleigh fading channels," *IEE Electron. Lett.*, vol. 26, pp. 301-303, 1 Mar. 1990.

[81] Zhu, Z. C. and A. P. Clark, "Rotationally invariant coded PSK signals," *IEE Proc.*, Pt. F, vol. 134, pp. 43-52, Feb. 1987.