DETECTION PROCESSES FOR

DIGITAL SATELLITE MODEMS

BY

STEPHEN BASIL AFTELAK, B.Sc.

A Doctoral Thesis

Submitted in partial fulfilment of the requirements for

the award of Doctor of Philosophy of the

Loughborough University of Technology

October, 1985

Supervisor: A.P. CLARK

Department of Electronic & Electrical Engineering

CONTENTS

PAGE NO.

PAGE NO.

PAGE NO.

ABSTRACT

The aim of this study is to devise detectors for digital satellite modems, that have tolerances to additive white Gaussian noise which are as close as possible to that for optimal detection, at a fraction of the equipment complexity required for optimal detection. Computer simulation tests and theoretical analyses are used to compare the proposed detectors.

Current proposals for digital satellite modems are discussed in relation to business and mobile radio systems. Two recent modulation methods, correlative phase shift keying and convolutionally encoded eight phase shift keying (coded 8PSK), are introduced as the schemes for which detectors are to be devised.

Maximum Likelihood detection implemented as the Viterbi Algorithm is considered, and is the preferred detector for the correlative phase shift keying modulation method. Near-maximum likelihood detectors, originally developed for data transmission systems with intersymbol interference, are investigated for coded 8PSK. They are shown to yield a tolerance to noise which is inferior to that of the Viterbi Algorithm detector, for similar levels of equipment complexity. The tests include the incorporation of suboptimal distance measures. A number of low complexity, but suboptimal, detectors for coded 8PSK are shown to have a low tolerance to noise. Two techniques, sequential decoding and a novel noise-adaptive Viterbi-type detector , which adapt the number of computations undertaken to suit the prevailing noise level, are considered. Extensive computer simulation results of the latter technique are presented. These results suggest that the technique is potentially much superior to the others tested.

ACKNOWLEDGEMENTS

The author would like to express his gratitude to his supervisor, Professor A.P. Clark, for his considerable help and guidance.

The financial support of the Loughborough University of Technology is gratefully acknowledged.

The author would like to thank Judith for her patience in typing the thesis.

Glossary of More Important Symbols and Terms

Tx	transmitter
Rx	receiver
T	duration of a signal element called the symbol interval
D	the delay operator (a delay of T seconds)
Y(f)	the baseband channel frequency response
Y	the vector of sample values $\{y_i\}$ of the impulse response of the baseband channel
E_b	energy per transmitted data bit
$\frac{1}{2}N_0$	two-sided power spectral density of additive white Gaussian noise at the receiver input
s_i	i^{th} four-level data symbol
$s_i(j)$ for j=1,2	the Gray Coded data bits carried by s_i
q_i	i^{th} four-level data symbol at the output of the precoder
$q_i(j)$ for j=1,2	the Gray Coded data bits carried by q_i
c_i	i^{th} code symbol
$c_i(j)$ for j=1,2,3	binary code symbols carried by c_i (for a rate-2/3 convolutional code)
p_i	i^{th} complex-valued symbol at the input to the modulator
w_i	sample value of the noise waveform w(t) at the output of the demodulator
r_i	i^{th} received sample of the demodulator output signal r(t)

ϕ_i i^{th} sample value of the phase, $\phi(t)$, of a signal

$\Delta\phi_i$ phase shift of a signal over the time interval $(i-1)T \leq t \leq iT$

Q'_i N-component vector, $[q'_{i-N+1}, q'_{i-N+2}, \ldots, q'_i]$ of possible values of the corresponding data symbols $\{q_j\}$

c'_i code symbol derived by coding the possible data symbol values in a vector Q'_i

$c'_i(j)$ for $j=1,2,3$ binary code symbols carried by c'_i (for a rate-2/3 convolutional code)

p'_i complex-valued symbol derived from the mapping of code symbol c'_i (for coded systems), or possible data symbol value q'_i (for uncoded systems).

$|w'_i|^2$ cost of a stored vector Q'_i

Φ_i the state of a Finite-State Machine at time $t=iT$

$\alpha(t)$ the frequency modulating pulse, which is proportional to the instantaneous rate of change of the phase of the signal at the output of a premodulation filter.

$\alpha'(t)$ the composite frequency modulating pulse, derived by incorporating the effects of coding into $\alpha(t)$

$\beta(t)$ the phase response function, which is proportional to the instantaneous phase of the signal at the output of a premodulation filter.

$\beta'(t)$ the composite phase response function, derived by incorporating the effects of coding into $\beta(t)$

$G(D)$ the generator matrix for a convolutional code, which has elements which are polynomials in D

$H^T(D)$ the syndrome former for a convolutional code, which is a matrix with elements which are polynomials in D

g_{ij} $(i,j)^{th}$ convolutional code sub-generator, which is the vector $[g_0(i,j), g_1(i,j), \ldots, g_k(i,j)]$ where k is the constraint length of the code. The vector elements have the possible values 0 or 1.

BER bit error rate in the detected data

binary-valued having the possible values 0 or 1

frequency-limited a frequency response is frequency-limited if it has non-zero values over a finite range of frequencies

time-limited a frequency response is time-limited if its sampled impulse response has non-zero values over a finite time interval

baseband channel the linear function which transforms the sequence of complex values $\{p_i\}$ into the complex waveform r(t)

Intersymbol interference occurs when each complex sample r_i at the input to the detector is a function of more than one complex number p_j at the transmitter, in the absence of noise, (where $j \leqslant i$).

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

As the demand for all areas of communications services expands, it is inevitable that the demand for satellite services will increase. Traditionally, satellite communication has been the preserve of a small number of international consortia, such as the International Telecommunications Satellite Organisation, INTELSAT, and the corresponding European organisation, EUTELSAT. These organisations are owned by the official telecommunications entities in the member countries, which operate the services. The services offered have been mainly trunk communications, using very large and costly earth stations. The traffic has comprised mainly of voice-circuits and television (TV) channels, and there is reason to believe that for many years to come, these services will remain dominant[3]. Increasingly though, there is a demand for a variety of new services, such as video-conferencing, direct broadcast of TV, radio communication between mobiles, and a multitude of new data services, including inter-computer links.[4-10] Clearly, such services are far removed from the more traditional trunk services, in that they are moving closer to the end-user. In particular, such traffic carried over satellite systems would probably require the provision of earth stations to the end-users.

The rising demand for the traditional trunk services alone is causing increased congestion in the geostationary orbit[3]. The additional demands expected for the new services will inevitably add to the problem. A number of techniques are available to "squeeze in" more services, given orbit and bandwidth restrictions. Higher frequency bands are now becoming available, in particular the 14/12GHz band and in the future the 30/20GHz band.[11,12] Since antenna gain increases with frequency, smaller antennas than those used in the 6/4GHz band can be used[3]. Unfortunately, the higher frequency bands are more susceptible to signal fades and depolarisation, due to atmospheric conditions.[7,12] Also, radio frequency (RF) equipment operating at these higher frequencies tends to be less efficient, so that the transmitter output power is more limited.[12] Another technique involves the "reuse" of the available frequencies by the use of signal carriers with orthogonal senses of polarisation, or by employing multiple spot-beam antennas.[7,12] In addition, orbital slot "reuse" can be achieved by placing a number of satellites in the same orbital slot operating at different frequencies, or by employing a large space platform on which a number of different communications payloads can be operated, using multiple spot-beam antennas.[12] Such techniques increase the level of co-channel (CCI)[3] and adjacent-channel (ACI)[3] interference, as the systems are "squeezed" together, both spatially and in terms of frequency. The above approaches basically provide new resources for satellite systems, whereas the approach of using more efficient transmission techniques is basically one of conserving existing resources. These techniques are aimed at improving the tradeoffs between bandwidth and power efficiency, and system complexity. (The latter is at least to some extent self-limiting in terms of maximum data rate, in that for a given level of equipment complexity, the receiver requires a given minimum amount of time to process each incoming signal element.) Such techniques include novel modulation and coding methods.

Increasingly, new satellite systems are digital rather than analogue. There are a number of reasons for this, in addition to the fact that much of the new traffic described earlier is digital in nature[3]. Time-

Division Multiple-Access (TDMA) can be used as the multiple-access method in digital systems.[3] This can achieve an increased capacity compared with analogue multiple-access systems[3]. TDMA transponders relay only one digitally modulated carrier so that intermodulation is not as critical as in analogue systems.[3] This increased capacity, linked to the decreasing cost of Large Scale Integration (LSI) digital circuit components, enhances the economic viability of digital satellite systems, compared with the corresponding analogue systems. Digital systems are inherently more robust in an interference environment, and can be operated at lower transmitter power levels than the corresponding analogue, Frequency-Division Multiplex, Frequency Modulated, (FDM-FM), systems[3]. The bit stream which comprises the digital signal has similar properties, whether it is a TV signal, a voice signal, or computer data, which in analogue form all have very different properties. Thus signal multiplexing and processing is very much simpler in a digital system. Satellite signals are more easily interfaced to terrestrial systems, (optical fibre, cable, or microwave), when in digital form. The predominance of digital circuit components in a digital satellite system provides a very predictable and repeatable performance, which is not subject to drift with time. The increasing reliability of digital circuit components, and the relative ease with which "soft-fail" systems can be designed, (where failure in one component leads to a degraded service rather than total breakdown), are also important factors.[6,8-10]

The increased confidence in the reliability of digital systems has led, in the last few years, to consideration of the feasibility of performing more complex functions on board the satellite, thereby simplifying the earth station hardware.[13-16] This is a very important step in reducing the cost of earth stations to the end-user who wishes to take advantage of the new satellite services. This philosophy is typified by the Communications Engineering Research Satellite (CERS) project.[5,6,8,10] The on board processing envisaged in this project can be grouped into three categories. Firstly, the satellite becomes the Master Access Controller (MAC) for the system, controlling the (TDMA) timing. Secondly, the satellite would include a processor for control and monitoring purposes, and as an exchange for re-routing incoming data (a "switchboard in the sky"). Thirdly, the proposed satellite includes on board demodulation and remodulation, commonly termed regeneration. In such a regenerative transponder only the up-link errors are re-transmitted on the down-link, whereas with conventional transparent transponders the up-link noise is amplified and re-transmitted on the down-link. Compared to a balanced transparent system, (equal signal to noise ratios on the up-link and down-link), the regenerative system would require between 2.5dB and 3dB less power on both links.[8] When the comparison is with an unbalanced transparent system, (lower signal to noise ratio on the down-link), which is more common, the potential power saving in the earth station transmitters is even greater, possibly as much as 9dB[8]. This benefit is available with direct phase regeneration as well as demodulation/remodulation methods,[8,17] but conversion of the signal stream into a baseband data sequence is essential for on board traffic processing. It is envisaged that the earth station modulators would operate in burst-mode, requiring a relatively complex on board burst demodulator, (requiring very fast synchronisation to incoming data). In contrast the satellite would re-transmit in continuous-mode, further reducing the complexity of the earth stations, where relatively simple continuous-mode demodulators would be deployed. Additional power savings are to be achieved by adaptive coding techniques, separately matched to noise, interference, and fading conditions, on the up-links and down-links.[8] These power-saving techniques allow the complex and costly earth station Travelling Wave Tube Amplifiers (TWTA) to be replaced by cheaper and more reliable ("soft-fail") Solid State Amplifiers (SSA)[8]. This philosophy of incorporating many of the more complex functions on board the satellite, could in fact reduce the cost of the earth stations to such an extent, that satellite communication becomes attractive to businesses and mobile radio users. For example, the CERS project also includes a mobile radio experiment, in which the potentially massive cost and equipment size reductions are very evident. The project envisaged very low cost printed array antennas which could incorporate only limited electronic steering, glued flat to the mobiles' roofs[8]. This was possible because of the chosen 12 hour eliptical (Molniya) orbit, which places the satellite within 15° of the vertical during 8 hours of the day, for the UK user. The remainder of the earth station would consist of equipment similar in size to a car radio. The orbit also alleviates many of the problems associated with mobile radio communications within built-up areas because of the Radio Frequency (RF) shadows thrown by tall buildings, since the satellite is essentially overhead during its operating period. Such a system could therefore be an attractive part of the solution to any proposed, Europe-wide, mobile radio system.

The above discussion, in relation to the CERS project, indicates that the technology is available to provide a viable service for business and mobile radio users. Despite this, it is generally accepted that satellites cannot compete with terrestrial fibre optic systems of the future, in the provision of general communications services. The "niche" for satellite services is where the broadcast nature of the service is of prime importance, in multi-point to point and point to multi-point communications such as mobile radio, electronic news gathering, remote printing, database transfer and updating, and as a back-up to terrestrial services.

This thesis is concerned with one aspect of the new generation of digital satellite systems; the modulator/demodulator (MODEM), and in particular the detection processes which are required to generate the demodulated digital data stream at the output of the demodulator. The remaining two sections in this chapter are concerned with the choice of the modulation methods for which appropriate detection processes are investigated, and with an outline of the contents of the thesis.

1.2 MODULATION METHODS

The modulation methods commonly considered for application to satellite systems can be classified in a number of different ways, but they all have one thing in common. In all cases the signals are either constant envelope or near-constant envelope. The reason for this is that satellite transponders are power limited, so that it is imperative that they should operate at or near the High Power Amplifier's (HPA) output level at saturation. At this point the typical HPA has a very nonlinear characteristic, so that an input signal with significant envelope variations, will be amplified such that the output signal has a significantly increased effective bandwidth, and is nonlinearly distorted. Therefore constant envelope, or near-constant envelope signals, are imperative. This means that only frequency or phase modulated signals are considered.

Currently, by far the most popular signal is bandlimited Quaternary Phase Shift Keying (QPSK), since it is tried and tested, the hardware is available, and careful design yields a reasonably tight bandwidth with tolerable distortion, even in nonlinear channels. A carefully designed filtered QPSK modulation scheme can achieve a transmission rate of 1.4 bits per second per Hz of channel bandwidth.[34] Bandlimited QPSK is an example of the first class of signals, of three classes in all, considered for the new satellite services. This class consists of non-continuous-phase signals which in a sense are not constant envelope, in that the envelope does fall to zero momentarily upon phase reversals. Figures 1.2.1 and 1.2.2 show two typical phase characteristics, $\phi(t)$ for non-continuous-phase signals, over a few symbol intervals. The phase is with respect to the phase of the carrier. Figure 1.2.1 depicts the case of Phase Shift Keying (PSK) modulation, while Figure 1.2.2 depicts

Frequency Shift Keying (FSK) modulation. Coding can be incorporated so that the number of possible signal waveforms over a symbol interval is greater than the number of levels that a particular data symbol can have.[12,19-28] By this means asymptotic coding gains, (that is, the coding gain as the signal to noise ratio gets very large), of several decibels (dB) in tolerance to additive white Gaussian noise (AWGN) are achievable, with no significant increase in bandwidth.[12] Tolerance to noise is defined at a given bit error rate, as the value of the signal to noise ratio which is required to achieve the given bit error rate. (These gains in tolerance to noise are only achievable if Maximum Likelihood detection is used.) For example, a Rate-2/3 convolutional code can be used to increase the number of levels, from four for the uncoded data, to eight in the coded data. A rate-k_0/n_0 convolutional coder outputs n_0 binary code symbols for every k_0 data bits at the input to the coder, where $n_0 \geq k_0$. Eight Phase Shift Keying (8PSK) is used as the modulation method (see Appendix A4).[20-23] Systems using signals of this type can employ conventional methods of carrier-phase and element-timing synchronisation.[6] A coded scheme can achieve such gains in tolerance to noise through two related mechanisms. Firstly, redundancy is incorporated in that coded messages either contain extra symbols, or the code symbols have more possible values than the uncoded symbols. This redundancy accentuates the uniqueness of the whole message to be transmitted. The redundancy is arranged so that it is very unlikely that noise in the transmission channel will corrupt enough of the symbols in a message to destroy its uniqueness. The second mechanism is noise averaging. This is caused by making each code symbol dependent on a span of data symbols. In this way if one or more code symbols are corrupted by noise, enough information usually remains, carried by other code symbols, for the detector to determine the data symbols which were transmitted. A major weakness of these signals is that they require significant bandlimiting prior to the HPA, in order to reduce their bandwidth, if data rates approaching that quoted earlier (in Section 1.1) are required.[6] Such bandlimiting introduces envelope ripples into the signal. When this signal is amplified nonlinearly in an HPA operating at or near its output saturation level, the effective bandwidth of the signal is increased, and nonlinear distortion is introduced into the demodulated signal at the receiver.[6,29-32] The former effect is termed spectral spreading. The effect, for this signal class, is more marked than for the other two classes.[6] Despite this, this class of signals does allow the use of coding, and generally requires relatively simple equipment.[6]

The second set of signals is characterised by the fact that they are constant or near-constant envelope signals where the signal phase is continuous, and where the signal frequency is held constant over a symbol interval.[6,33] The collective name for such schemes is Continuous-Phase Frequency Shift Keying (CPFSK), (although strictly the classification includes signals in the third class as well).[34]

A typical phase characteristic for CPFSK modulation is given in Figure 1.2.3. Signals of this class include Minimum Shift Keying (MSK),[29,31] Offset-QPSK (OQPSK),[29,31] and Intersymbol Interference and Jitter-Free OQPSK (IJF-OQPSK).[13,35,36] Also included in this class is the Multi-h modulation method, (where the signal frequency is here taken to be constant over a symbol interval).[37-41] This scheme involves the cyclic variation of the modulation index, h, between a number of discrete values over consecutive symbol intervals, (h is constant over a symbol interval). The value h is proportional to the rate of change of phase ($d\phi(t)/dt$), and therefore determines the slope of the phase characteristic at any point. This is again a coding technique by which the number of possible signal shapes in the modulating waveform over a symbol interval, is greater than the number of values that a data symbol can have.[34] The advantage of this class of signals is that they are tolerant to the nonlinear effects of HPAs operating at, or near, their (output) saturation levels, even when the signals are bandlimited.[6]

The third class of signals is categorised by the fact that the signals are constant or near-constant envelope modulations, where both the phase and the frequency are continuous.[42-49,51-62] These essentially have rounded phase characteristics and are very often, (but not necessarily), correlatively coded, in that the shape of the phase trajectory over a symbol interval is a function of a number of successive data symbols. Such signals are usually referred to as Continuous Phase Modulation (CPM).[34] A typical phase characteristic is given in Figure 1.2.4. Note, as in Figure 1.2.4, that the signal phase may not be constrained to pass through fixed points, (multiples of $h\pi$ radians), at the symbol sampling instances, $t=iT$. (In particular, many of the schemes of References 49 and 55 to 61 are of this type.) The smoothed-phase characteristics of such schemes restrict the bandwidth by reducing the maximum rate of change of phase. Correlation between the phase shapes over a number of symbol intervals can be achieved by explicit correlative-level phase coding,[34,46-48,62] or by specifying frequency modulating pulses which extend over a number of symbol intervals.[42-45,55-57]

The two techniques are equivalent. (Appendix A2 describes the theory for a scheme using explicit correlative-level phase coding.) Since the coding, explicitly or implicitly applied, contributes to the smoothness of the phase, it can contribute to restricting the signal's bandwidth.[6] Alternatively, this coding can also be seen again as increasing the number of phase shapes, (phase trajectories), possible for the signal over a symbol interval, compared with the number of values that a data symbol can have. For example the modulation method termed CORPSK(4-7,1+D) incorporates correlative-level phase coding, followed by premodulation filtering and a frequency modulator, (see Appendix A2 and References 34 and 62). Each data symbol can have one of four different values, whereas the modulating waveform can have one of seven different shapes, (phase trajectories), over a symbol interval. Other modulation methods which come under this general heading include Gaussian filtered MSK (GMSK),[42-45] Tamed Frequency Modulation (TFM),[46-48] the so-called CORPSK signals,[34,62] and a whole class of partial response signals which do not explicitly include coding.[49,54-57] In addition, a number of partial response schemes which include convolutional encoding, have been proposed.[58-60] These are similar in many ways to the schemes within the first class of modulations defined above, which incorporate convolutional coding,[20-26] but are more complex in that the phase is smoothed. The main advantage that this third class of signals has is that, in non-bandlimited form, they are not subject to distortion or spectrum spreading when fed through an HPA operating at, or near, its saturation level.[6,34,49] Unfortunately they tend to have a wider bandwidth than many interesting signals in the other classes (when the latter are not subject to nonlinear distortion).[6]

The rounding of the phase waveform appears in general to result in a reduction of 1dB in tolerance to additive white Gaussian noise, compared with what is theoretically achievable without phase shaping[6]. Also, any bandlimiting of the signals results in quite severe non-linear distortion and spectrum spreading, when the signal is passed through an HPA operating at, or near, its saturation level[6]. This class of signals may also require quite sophisticated carrier-phase and element-timing synchronisation techniques, especially when the phase does not pass through fixed points at every symbol sampling instant.[49]

Having described the general classes of signal which are being considered for future satellite services, particular modulation schemes are now considered in relation to the technical requirements of the new services. From Section 1.1, an important feature is that the earth station should be simple, in order to make it cheap and reliable (and unmaned if possible). Because of this it will inevitably be power-limited, so that the modulation method which yields an advantage in tolerance to noise over QPSK would be very attractive, (in addition to the power advantages listed in Section 1.1 due to on board regeneration). Clearly, as the available spectrum becomes more congested, bandwidth efficiency will rise in importance.

Unfortunately, signals of the second class described above, despite having a relatively low level of spectrum spreading after non-linear amplification, cannot generally provide additional coding gain over QPSK. The need to maintain an approximately constant envelope precludes the types of convolutional coding schemes which were described for the other signals[6]. In addition, bandlimiting of the signal introduces intersymbol interference which results in a reduced tolerance to noise, if simple threshold-level detection is used. (Maximum Likelihood detection, (see Appendix A3), would be an added complication which only restores performance, in terms of tolerance to noise, to that of QPSK modulation. The exception with regard to coding gain is Multi-h modulation, which is considered later.)

The general property of the signals in the first and third groups, which yields advantages in tolerance to noise over QPSK, is that the modulating waveform over a symbol interval is a function of a number of data symbols, so that the signals are correlated. This is evident in that the number of possible shapes of the modulating waveform over a symbol interval, is greater than the number of possible values of a single data symbol. Such correlation increases the minimum Euclidean distance (or equivalently, the mean square error) between possible signal waveforms, as compared with the corresponding uncoded scheme.[19] This increased distance means that, given optimal detection, a higher level of noise is needed to give detection errors in the coded case, than in the uncoded case. In order to exploit the increased distances, the detector must now consider the received signal over a number of consecutive symbol intervals, in order to detect one data symbol. In the limit, the whole of the received message can be detected in one operation. This so-called Maximum Likelihood detection selects as the detected message the possible sequence of data symbols, for which there is the minimum Euclidean distance, (mean square error), between the possible received signal corresponding to this data sequence (in the absence of noise), and the signal actually received. The detection process extends over the whole received signal. If the different possible signals are equally likely, this process minimises the probability of choosing a wrong sequence of data symbols.[4] Under certain conditions, this process can be implemented by means of the Viterbi Algorithm.[63] Unfortunately, this usually results in a much more complex detector, than that required for QPSK modulation, (the threshold-level detector[1]). A signal of the first class which can gain substantially in tolerance to noise over QPSK, is convolutionally encoded and phase-mapped eight phase shift keying, referred to as coded 8PSK in this thesis, (see Appendix A4 and References 20 to 23). In terms of bandwidth, the signal is very similar to QPSK, so that a very attractive power advantage can be gained at no expense in terms of bandwidth. Multi-h signalling can also yield quite significant gains in tolerance to noise, but these schemes tend to be rather complex,[4] and need modulation index synchronisation in addition to carrier-phase and element-timing synchronisation.[40,49] Synchronization is, in fact, a major problem for such signals. (Reference 24 compares coded 8PSK modulation with Multi-h signalling, and concludes that the former technique is generally more attractive.) Synchronisation is also a problem for many CPM schemes, especially for those where the signal phase does not pass through fixed points at the end of each symbol interval.[6,49] Because of this, and because Maximum Likelihood detection is often unduly complex for these signals,[6,49] the latter schemes are not considered further. A signal of the third group for which synchronisation is somewhat simpler, (because the signal phase does pass through fixed points at the end of each symbol interval), is the class of correlative-level phase coded signals termed CORPSK.[34,62] A particularly attractive scheme is CORPSK(4-7,1+D), (see Reference 34 and 62 and Appendix A2), in that the scheme potentially gains 2dB in tolerance to noise (at high signal to noise ratios), compared with differentially coded QPSK (DQPSK), whilst its effective bandwidth is not much greater than that of QPSK (and its frequency spectrum has no sidelobes[62]). In addition, the scheme yields advantages in tolerance to noise at bit error rates (BER) as low as 1 in 100[62]. Coded 8PSK, on the other hand, is inferior in terms of tolerance to noise compared with QPSK, for BERs in excess of 1 in 100[12,21] A demodulator suitable for the projected new satellite services would not be required to operate at signal to noise ratios such that the BER is much less than 1 in 10^{4} at its output. This is because digitally coded speech, (the predominant type of traffic), can be transmitted at error rates as high as 1 in 100, so that much lower error rates are unnecessary[8]. To achieve the required bit error rate for computer data, (less than 1 in 10^{9}), adaptive coding external to the modem would be used[8]. For this reason the most promising signals are those which yield substantial gains in tolerance to noise in the region of BER, 1 in 10^{2} to 1 in 10^{4}. In this thesis the region of interest for the BER is defined as 1 in 10^{3} to 1 in 10^{4}.

The two modulation methods, coded 8PSK and CORPSK(4-7,1+D) were chosen as being the most promising signals for application to new business and mobile radio satellite systems. In both cases the correlation in the signals is exploited to gain advantages in tolerance to noise over QPSK. This exploitation is achieved by using Maximum Likelihood detection in the form of the Viterbi Algorithm. This results in a significant increase in detector equipment complexity compared with threshold-level detection for QPSK, and may under certain circumstances be unduly complex. This thesis is concerned with investigating near-maximum likelihood detection techniques which achieve, as closely as possible, the best tolerance to noise (which is achieved using Maximum Likelihood detection), with a significant reduction in detector equipment complexity, compared with Viterbi Algorithm detection. The discovery of practical alternatives to the Viterbi Algorithm for the detector should go a long way towards making these schemes feasible, justifying their increased complexity in comparison with QPSK modulation.

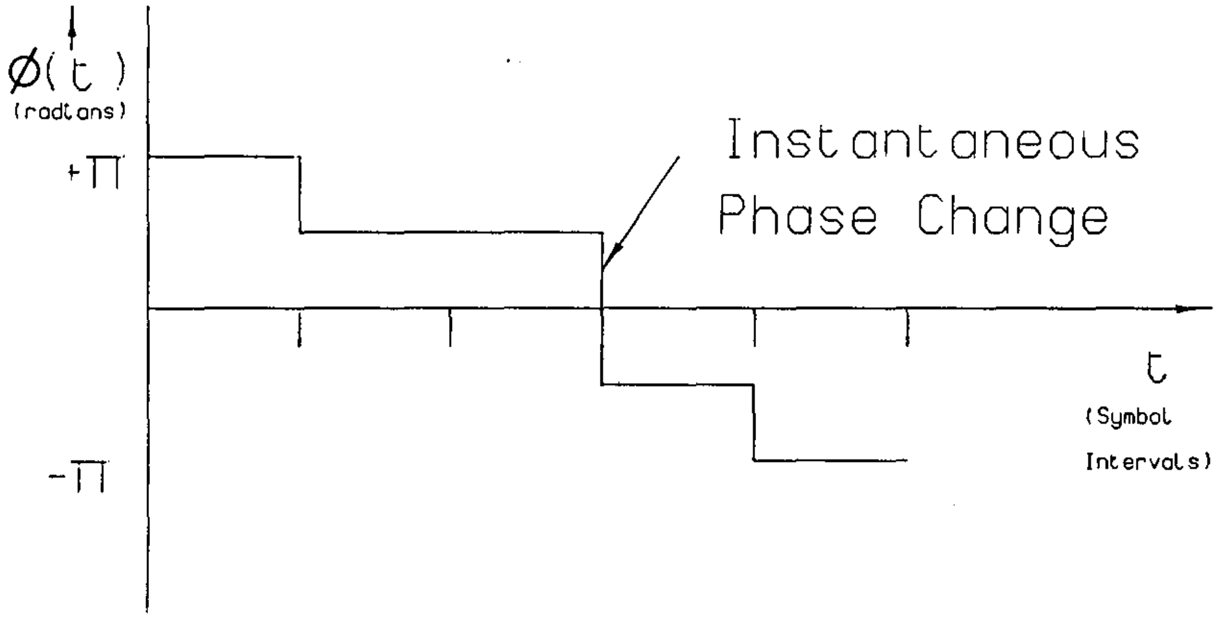


Figure 1.2.1 Typical Phase Characteristic for a PSK Signal

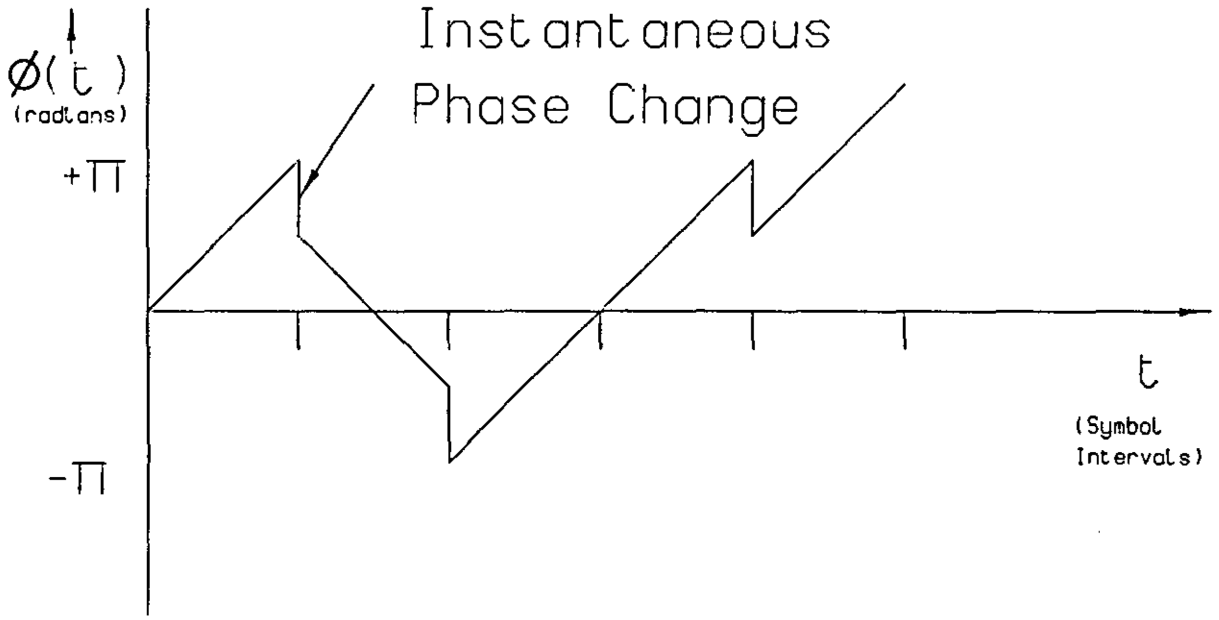


Figure 1.2.2 Typical Phase Characteristic for an FSK Signal

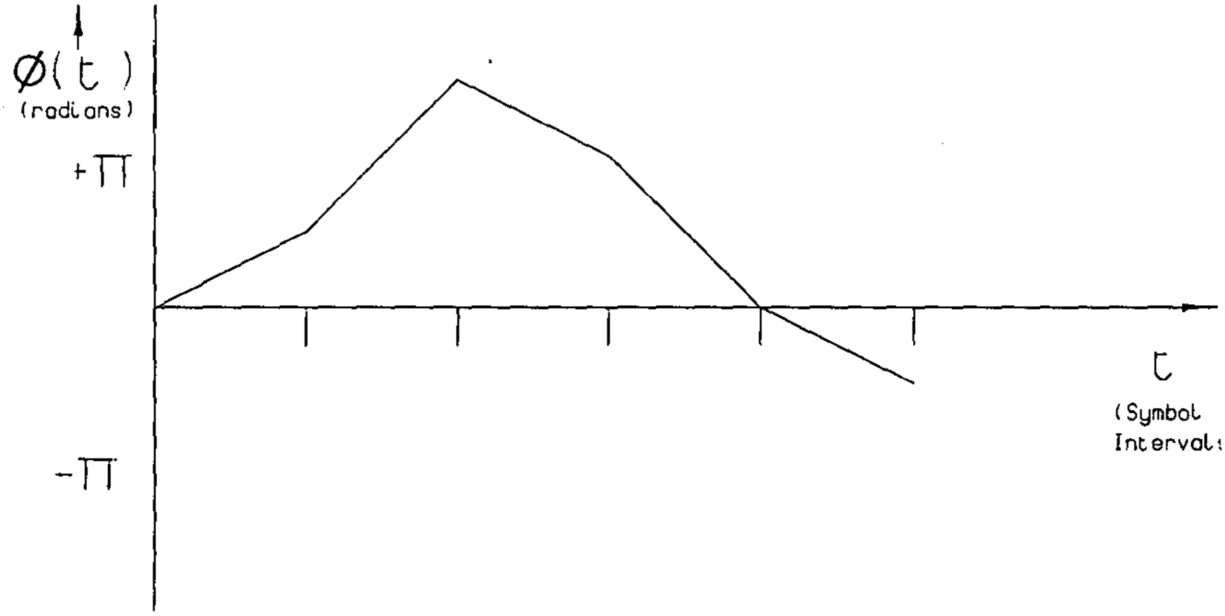


Figure 1.2.3 Typical Phase Characteristic for a CPFSK Signal

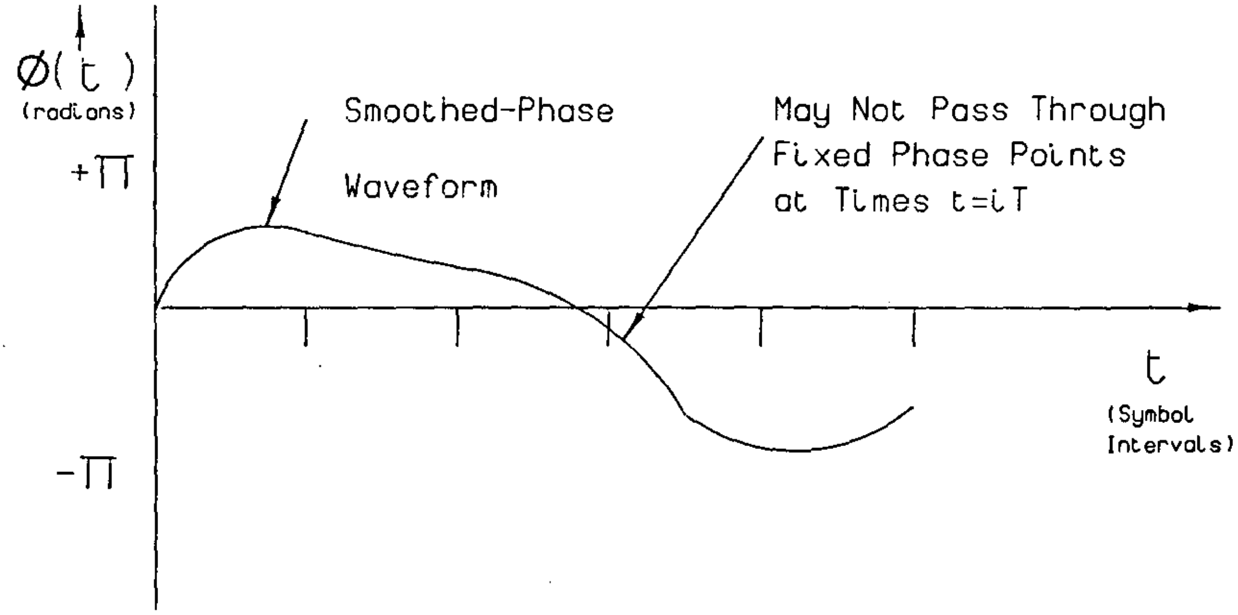


Figure 1.2.4 Typical Phase Characteristic for a General CPM Signal

1.3 OUTLINE OF INVESTIGATION

This investigation is primarily concerned with the study of suitable detection processes for coded modulation methods, as described in Section 1.2. The aim has been to devise detection schemes which are considerably less complex, in terms of hardware and cost, than Maximum Likelihood detection implemented by way of the Viterbi Algorithm. The performance of the simplified detectors, in terms of their tolerances to additive white Gaussian noise, should be as close as possible to the performance achieved by Maximum Likelihood detection. (As a rough indicator, the selected detectors' tolerances to noise should not be degraded by significantly more than Ø.5dB, at a bit error rate of 1 in $1Ø^4$.) Computer simulation tests have been used to compare the detection schemes, (see Appendix A5 for a description of the techniques).

Chapter 2 describes the system models used in the computer simulation tests, minus the detectors. Initially, a general system model is described, which applies to all the models. The following sections describe details of particular models, not dealt with in the general description. The models are of QPSK/DQPSK modulation, CORPSK(4-7,1+D) modulation, (both an ideal model, and a more practical model incorporating premodulation and equipment filtering), and coded 8PSK modulation.

Chapter 3 describes the optimal (or near-optimal) detection schemes for QPSK (threshold-level detection),[1,2] and the coded schemes (Viterbi Algorithm detection).[1,2] Simulation results are presented for the schemes, including results for QPSK with realistic equipment filtering.

Chapter 4 describes the application of near-maximum likelihood techniques,[64-68] originally applied to telephone channels with intersymbol interference, to coded 8PSK modulation. Pseudobinary detection schemes[69-71], and a number of extensions of the original System 1 scheme,[64,65] are considered.

Chapter 5 describes a number of different detection techniques for coded 8PSK, all of which yield very degraded tolerances to noise, compared to Maximum Likelihood detection. A non-linear equaliser-like scheme, and the (feedforward) inverse coder were tested. Also a technique of redefining the meaning of the state of a stored vector and soft-decision table look-up syndrome decoding, are tested.

Chapter 6 deals with two types of detector, wherein the number of computations performed changes from symbol interval to symbol interval, necessitating the provision of buffer storage for the received signal samples, and the detected data symbols. The class of detectors termed sequential decoders is addressed in the first section, but no simulation tests of such schemes are undertaken. The second section describes a rather different approach, termed noise-adaptive Viterbi-type detection, which unlike sequential decoding requires no back-up searches,[19] and is therefore a basically feedforward technique. Simulation results for a number of schemes are presented, including the schemes tolerances to noise and statistical measures of the processing load, (in terms of the number of possible transmitted sequences to be processed per symbol interval).

Chapter 7 compares the results for the preferred Viterbi Algorithm detector for CORPSK(4-7,1+D) modulation, with those for Viterbi Algorithm, near-maximum likelihood System 1,and noise-adaptive Viterbi-type detection, for coded 8PSK.

In order to deal with the many variants of the schemes, a unified method of describing the schemes being tested has been devised based on various parameters of the schemes. Appendix A8 details the system, and should be consulted in order to fully understand the presentation of the results.

CHAPTER 2

DATA TRANSMISSION SYSTEM MODELS

This chapter briefly describes the mathematical models of the data transmission systems, within which the various detectors of Chapters 3 to 6 are incorporated. The models cover uncoded QPSK modulation and the two chosen modulation schemes which use coded data (see Chapter 1). Since QPSK is a standard modulation method for satellite systems (Chapter 1), the results for the QPSK model are used as a reference by which the relative performance of the coded schemes incorporating different detectors can be gauged. The models upon which the computer simulation programs are based are all described in this one chapter. This avoids unnecessary duplication in Chapters 3 to 6 where the detectors are described. It also facilitates the description of techniques incorporated into the programs in order to reduce the computing time, during the simulation tests.

In addition, the three modulation methods have a large number of common features which are described in the general model of Section 2.1.

Sections 2.2 to 2.5 describe the functions of the various blocks introduced in the model of Section 2.1, for the four models used in the investigations. These descriptions are brief, since their function is to describe the signals which appear at the input to the detectors of Chapters 3 to 6. The characteristics of these signals clearly have a bearing on the complexity of the detectors' task. (More details concerning the two coded modulation methods are given in Appendices A2 and A4.)

Section 2.2 describes the model for QPSK modulation. Sections 2.3 and 2.4 both deal with the coded scheme called CORPSK(4-7,1+D)[62]. The model of Section 2.3 is very simple, and is used to gain an indication of the potential performance of the scheme. Section 2.5

describes the model for the coded 8PSK modulation method, which is the second coded modulation method chosen in Chapter 1.

2.1 GENERAL SYSTEM MODEL

A diagram of the generalised data transmission model for all schemes investigated is given in Figure 2.1.1.

The baseband signal generator produces a sequence of four-level data symbols $\{s_i\}$; s_i=0,1,2 or 3, the symbols being statistically independent and equally likely to have any of their four different values. Each four-level symbol carries two bits of information. The mapping from the four-level data to the binary data is given by the Gray Code as outlined in Table 2.1.1. It is assumed that $s_i=0$ for $i \leq 0$ so that s_i is the i^{th} transmitted symbol at time t=iT, where T is the symbol duration in seconds. In Figure 2.1.1, at the input to any filter or linear channel, the symbols are assumed to be carried by the corresponding impulses. For example the symbols at the input to the precoder are carried by the impulses $\{s_i\delta(t-iT)\}$. Figure 2.1.1 includes the option of precoding the data sequence. Precoding is used to reduce the lengths of the error bursts in the detected data, as explained in Appendix A1. The definition of an error burst is given in Appendix A5. At time t=iT, the output of the precoder is

$$q_i = [s_i - q_{i-1}]\text{MODULO-4} \qquad (2.1.1)$$

where the MODULO-4 rule is defined as,

$$\begin{aligned} &q_i < 0; \quad q_i = q_i+4 \\ &0 \leq q_i \leq 3 \; ; \quad q_i = q_i \\ &q_i > 3; \; q_i = q_i-4 \end{aligned} \qquad (2.1.2)$$

As an example if $s_i=1$ and $q_{i-1}=3$, then $q_i=[1-3]$MODULO-4=2.

The precoder output symbols $\{q_i\}$ are four-level, the symbols being statistically independent and equally likely to have any of their four different values.[51] It is assumed at the start of transmission that $q_{-1}=0$. For the remainder of Chapter 2 it will be assumed that precoding has been applied and therefore the symbols $\{q_i\}$ will be used. For schemes where precoding is not applied (or is optional), q_i can be directly replaced by s_i.

The four-level symbols $\{q_i\}$ are fed to the encoder and mapper shown in Figure 2.1.1. Coding is applied to all the systems investigated except QPSK. Coding is used to gain an improvement in tolerance to noise (termed a coding gain) over the corresponding uncoded system. Chapter 1 introduced the concept, and Appendices A2 and A4 detail the coded schemes considered in this thesis.

In general the coding process converts the four-level data symbols $\{q_i\}$ into ℓ-level symbols (which are integers) $\{c_i\}$. In general, any coding will involve correlating a number of symbols $\{q_j\}$, $j\leq i$, to give a code symbol c_i.

The ℓ-level symbols $\{c_i\}$ are mapped onto a sequence of complex numbers $\{p_i\}$ which have m possible values. (The $\{p_i\}$ are termed m-level numbers.) The mapping of the $\{c_i\}$ into the $\{p_i\}$ is detailed in the appropriate sections of Chapter 2.

In all cases $|p_i|^2=\{\mathrm{Re}(p_i)\}^2+\{\mathrm{Im}(p_i)\}^2=4.0$. This sequence is fed to a phase or frequency modulator which incorporates any premodulation filtering, and the transmitter equipment filters which are required to restrict the signal bandwidth.

The modulator, satellite channel, and demodulator, comprise the baseband channel. In practice the satellite channel is nonlinear, but for the purposes of this study, which is primarily concerned with detection processes, the channel is assumed to be linear. In particular this assumes that the Travelling Wave Tube Amplifier (TWTA) in the transmitter is backed-off sufficiently from saturation, so that operation is within the linear portion of its characteristic. In addition adjacent and co-channel interference[3] are neglected in this study. The demodulator includes all the receiver filters.

The baseband channels which are used in the computer simulation tests are now defined. These definitions do not include the effects of any premodulation filtering. These latter effects are described in the sections of Chapter 2 which describe the models incorporating premodulation filtering, (Sections 2.3 and 2.4). Unless otherwise stated the filtering is shared equally between the transmitter and the receiver such that the channel frequency response is of the form given in Equation 2.1.3. Schemes using this baseband channel frequency response are called perfect channel schemes, since this channel has the minimum bandwidth required to transmit the signal with no intersymbol interference.[1]

$$Y(f) = \begin{cases} T & ; \ |f| \leq 1/(2T) \\ \\ 0 & ; \ |f| > 1/(2T) \end{cases} \qquad (2.1.3)$$

This is termed the Ch=I1 channel, (see Appendix A8). Alternatively the baseband channel may be defined by the impulse responses given in Graph 2.1.1. These impulse responses are those of actual filters designed by Mr. M.J. Fairfield of Loughborough University, in conjunction with the UNIVERSE and CERS projects.[5-10] A second alternative channel is the Raised Cosine channel described by the frequency response in Equation 2.1.4, termed Ch=RC (see Appendix A8).

$$Y(f) = \begin{cases} \tfrac{1}{2}T(1+\cos\pi fT) \ ; & -\frac{1}{T} < f < \frac{1}{T} \\ 0 \ ; & \text{elsewhere.} \end{cases} \qquad (2.1.4)$$

Stationary white Gaussian noise, with zero mean and a two-sided power spectral density $\frac{1}{2}N_0$, is added to the modulated signal at the receiver input (see Appendix A5). The demodulator includes at its input a bandpass filter which removes the frequency components outside the frequency band of the data signal. The resultant passband signal is fed to two linear coherent demodulators whose reference carriers are in phase quadrature and have the same frequency as that of the received signal carrier. Perfect carrier frequency and phase synchronisation are assumed. The demodulated signals at the outputs of the in-phase and quadrature demodulators are taken to be real and complex-valued respectively.

The complex-valued sampled impulse response of the baseband channel is given by the inverse Fourier Transform of the channel frequency response Y(f) (which includes the effects of premodulation filtering, where appropriate).

$$y(t) = \int_{-\infty}^{\infty} Y(f)\exp(j2\pi ft)df \qquad (2.1.5)$$

where $j=\sqrt{-1}$ and f and t are, respectively, frequency in Hz and time in seconds. y(t) is assumed to be time-invariant so that y(t-iT) is a time-shifted version of y(t-kT) for $i \neq k$. The complex-valued Gaussian noise waveform at the output of the demodulator is w(t).

Hence the received and demodulated signal is

$$r(t) = \sum_{i=1}^{\infty} p_i y(t-iT) + w(t) \qquad (2.1.6)$$

The waveform r(t) is sampled once or twice per data element at the time instants $\{iT\}$ or $\{\frac{iT}{2}\}$ respectively to give the received samples $\{r_i\}$ where $r_i=r(iT)$, or $\{r_j\}$ where $r_j=r\{\frac{jT}{2}\}$, respectively. The case where r(t) is sampled once per symbol interval is now described. The extension to double sampling is described in the appropriate sections of the thesis. Perfect timing synchronisation is assumed. The noise component of the received sample, $w_i=w(iT)$, is a complex-valued Gaussian random variable. The receiver filtering is such that the real and imaginary parts of the $\{w_i\}$ are statistically independent Gaussian random variables with zero mean and fixed variance σ^2, unless otherwise stated. The waveform r(t) is sampled at or near the Nyquist rate.[1] The sampled impulse response of the baseband channel is the (g+1)-component vector $Y=[y_0,y_1,\ldots,y_g]$ where y_i is complex-valued. The delay introduced by the baseband channel has been neglected, so that the constituent filters are not physically realisable.

The sample at the input to the Decoder/Detector at time t=iT is the complex-valued quantity,

$$r_i = \sum_{h=0}^{g} p_{i-h} y_h + w_i \qquad (2.1.7)$$

The term Decoder/Detector is used to indicate that the detection processes inherently include the decoding operation for coded signals. The output of the Decoder/Detector is the sequence of symbols $\{q_i'\}$ where q_i' is the detector's decision as to the value of q_i. The decoder which follows is the inverse of the precoder at the transmitter, (see Appendix A1). At time t=iT; the output of the decoder is given by

$$s'_i = [q'_i + q'_{i-1}]\text{MODULO-4} \qquad (2.1.8)$$

where the MODULO-4 rule is defined to be

$$\begin{aligned} &s'_i < 0 \ ; \ s'_i = s'_i + 4 \\ &0 \leq s'_i \leq 3; \ s' = s'_i \\ &s'_i > 3 \ ; \ s'_i = s'_i - 4 \end{aligned} \qquad (2.1.9)$$

Clearly, if precoding is not used at the transmitter, the output of the Decoder/Detector is the sequence of symbols $\{s'_i\}$.

4-LEVEL SYMBOL	TWO BINARY SYMBOLS CORRESPONDING TO THE 4-LEVEL SYMBOL	
0	0	0
1	0	1
2	1	~~1~~0
3	1	~~0~~1

TABLE 2.1.1: Gray Code Mapping

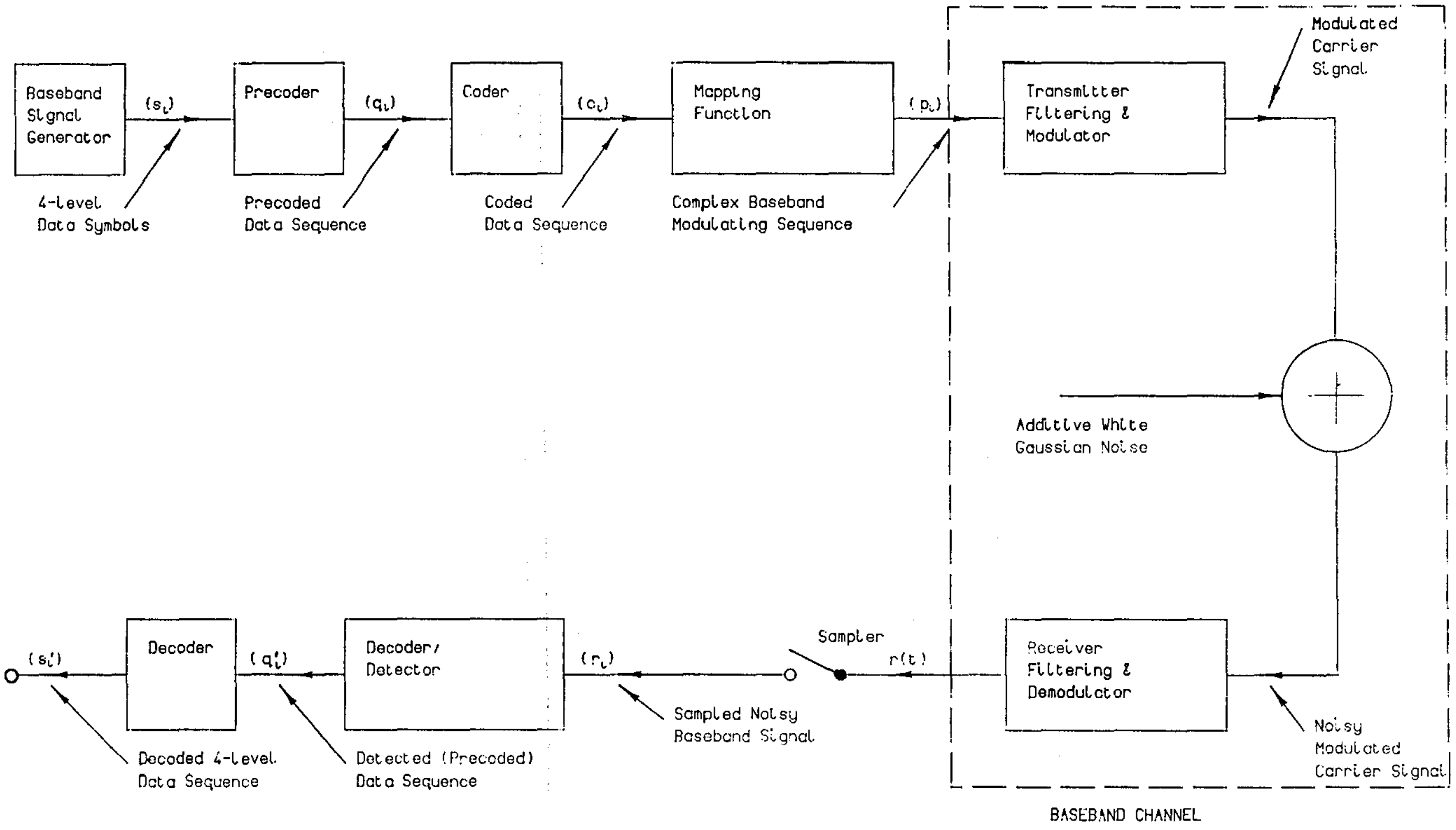
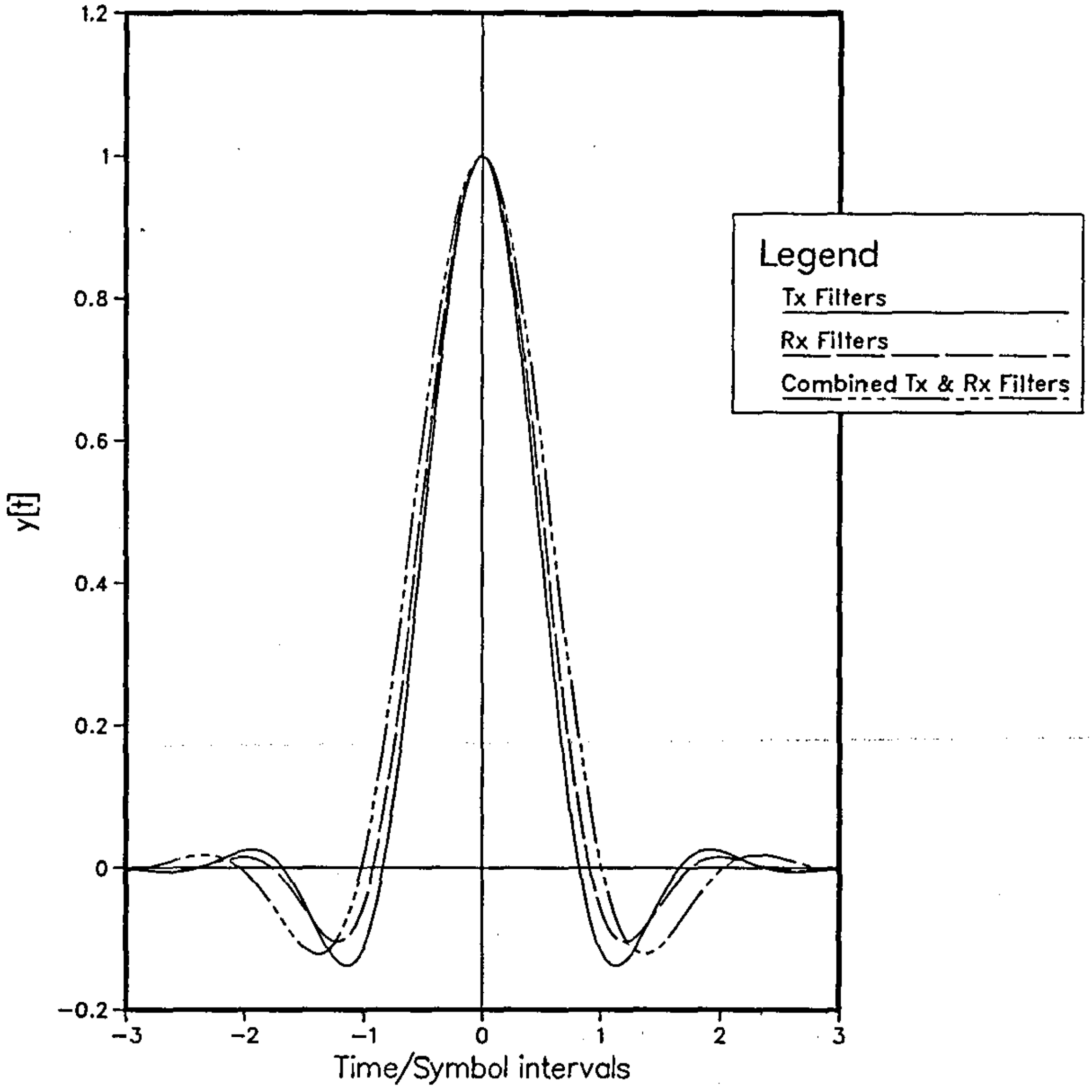

Baseband Signal Generator
(s_i)
4-Level Data Symbols
Precoder
(q_i)
Precoded Data Sequence
Coder
(c_i)
Coded Data Sequence
Mapping Function
(p_i)
Complex Baseband Modulating Sequence
Transmitter Filtering & Modulator
Modulated Carrier Signal
Additive White Gaussian Noise
Receiver Filtering & Demodulator
Noisy Modulated Carrier Signal
$r(t)$
Sampler
Sampled Noisy Baseband Signal
(r_i)
Decoder/ Detector
(q_i')
Detected (Precoded) Data Sequence
Decoder
(s_i')
Decoded 4-Level Data Sequence
BASEBAND CHANNEL

Figure 2.1.1 General System Model

Graph 2.1.1 Channel Ch=Mn
[See Appendix A8]



2.2 QUADRATURE PHASE-SHIFT-KEYING (QPSK) CHANNEL MODEL

In the case of the QPSK channel model, the "CODER" block in Figure 2.1.1 is not incorporated. Precoding is used to produce DQPSK (Differential QPSK). The modulator now becomes the appropriate QPSK modulator while the demodulator becomes the appropriate QPSK demodulator. (No assumptions regarding the actual hardware configurations are made.)

The representation of the $\{p_i\}$ in the complex number plane is given in the signal constellation of Figure 2.2.1. See Appendix B1 for the program listing.

In addition, this model allows transmission at a lower rate than the nominal rate of say ℓ bits/second. The additional lower rates $\ell/2$, $\ell/4$ and $\ell/8$ bits/second, are achieved at a constant transmitter symbol rate of $\ell/2$ bauds through repeated transmission of data symbols $\{s_i\}$, see Table 2.2.1. At the receiver, the received complex samples corresponding to one transmitted data symbol are simply added before being fed to the detector.

INFORMATION TRANSMISSION RATE (bits/second)	NUMBER OF THE $\{r_i\}$ WHICH ARE A FUNCTION OF A SINGLE DATA SYMBOL s_j
ℓ	1
$\ell/2$	2
$\ell/4$	4
$\ell/8$	8

TABLE 2.2.1: Transmission Rate Options

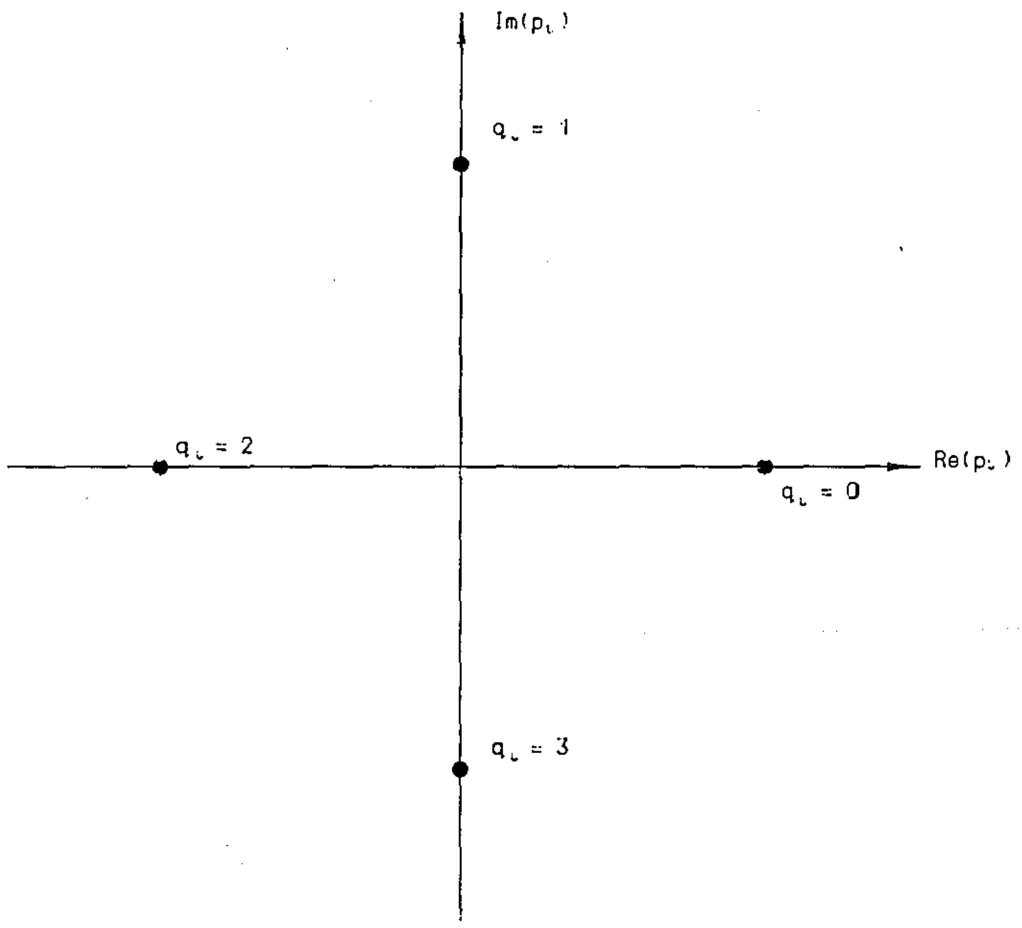


Figure 2.2.1 QPSK Signal Constellation

2.3 CORPSK(4-7,1+D) PERFECT CHANNEL MODEL

Three basic models are described in this section. The first is a simplified version of the differential-phase (Frequency Modulated) system described by Muilwijk.[62] The remaining two are direct phase-mapped derivatives, (Phase Modulation), of the differential scheme (termed direct phase-map schemes A and B). The schemes differ only in respect of the mapping function onto the complex number plane. Precoding is retained as an option in all cases. See Appendix B2 for the differential-phase scheme program listing and Appendix B3 for the direct phase-map scheme B program listing.

The encoder in all three models is a correlative-level encoder with transfer function (1+D) where D is the delay operator describing a delay of T seconds (see Appendix A2). The encoder operates on the sequence of symbols $\{q_i\}$ as follows, at time t=iT.

$$c_i = q_i + q_{i-1} \qquad (2.3.1)$$

The interaction of precoding with the correlative-level coding is discussed in Appendix A1. The sequence of symbols $\{c_i\}$ is seven-level c_i=0,1,2,3,4,5 or 6, where the seven levels are not all equally likely.

The modulator includes a premodulation filter at baseband, whose smoothing action on the coded and mapped data produces a continuous-phase waveform at the output of the modulator. The premodulation filter's smoothing action on the phase restricts the bandwidth of the signal in the channel, as discussed in Chapter 1.

In the differential-phase model, the sequence of code symbols $\{c_i\}$ is mapped onto phase shifts $\{\Delta\phi_i\}$ which occur over the symbol intervals $\{(i-1)T\leq t\leq iT\}$. The resultant phase samples, $\{\phi_i\}$, where $\phi_i=\phi(iT)$ are measured with respect to the phase of the carrier. The $\{\phi_i\}$ are the phase angles of the complex numbers $\{p_i\}$ in polar form. The mapping is given in Table 2.3.1. (ϕ_0 at the start of transmission is assumed to be zero radians.) Similarly, the mapping rules for the two direct phase-map schemes, A and B, are outlined in Table 2.3.2. In these cases the mapping is from the sequence of symbols $\{c_i\}$ to the complex numbers $\{p_i\}$ whose phase angles $\{\phi_i\}$ are given in Table 2.3.2. The direction of the phase trajectory for direct phase-map scheme B is a function of ϕ_i and ϕ_{i-1}, the present and previous phase samples respectively. For this reason, Table 2.3.2 distinguishes $c_i=2$ and $c_i=6$ by assigning phases $-\pi/2$ and $+3\pi/2$ radians to them respectively. The direction of the phase trajectory is found by considering ϕ_i and ϕ_{i-1}. The intermediate phase at time $t=(i-1/2)T$ is found by simply adding ϕ_i and ϕ_{i-1} and dividing by two (i.e. superposition). For $\phi_{i-1}=-\pi$ $(c_{i-1}=1)$ and $\phi_i=+\pi/2$ $(c_i=4)$, $\phi_{i-\frac{1}{2}}=-\frac{3\pi}{4}$ radians. Therefore the direction of the phase change is clearly anticlockwise (increasing phase). Since in all cases the mapping is basically of seven-level symbols onto a four-point constellation, a MODULO-4 constraint is inherent. In the direct phase-map schemes this is clearly apparent from the non-unique mappings indicated in Table 2.3.2. In the case of the differential-phase scheme, the phase shifts $\Delta\phi_i=+\pi/2$ and $-3\pi/2$ radians yield the same final phase ϕ_i, for a given initial phase ϕ_{i-1}. This is shown in Figure 2.3.1. Therefore, if the received signal $r(t)$ is sampled once per data element, these two different phase shifts cannot be distinguished. In order to distinguish between the seven different possible values of c_i, seven different phase trajectories are possible, given a particular initial phase at time t=(i-1)T. The seven different phase trajectories are depicted in Figure 2.3.2 for ϕ_{i-1}=O radians.

For the purposes of these initial investigations, it is assumed that the premodulation filter is such that, in the absence of noise, the demodulated baseband signal r(t) moves round the signal constellation at a constant rate, so that the value of r(t) at time t=(i-1/2)T is midway between the initial and final phase points on the envelope, given the direction of the phase trajectory outlined in Table 2.3.1 or Table 2.3.2. This assumption is not realistic, as will be seen in Section 2.4 when specific premodulation filtering is introduced. The results for this model are to be considered as an upper-bound to the results for the more realistic model of Section 2.4.

As a result of the detector's requirement for information concerning the phase trajectory during a symbol interval, the received signal must be sampled twice per signal element. This means that the channel frequency response (outlined in Equation 2.1.2), requires ammendment in order to satisfy Nyquist's sampling theorem, (so that the extra sample at time t=(i-1/2)T contains useful information)[1]. The ammended frequency response is given in Equation 2.3.2

$$Y(f) = \begin{cases} T & ; \ |f| \leqslant 1/T \\ 0 & ; \ |f| > 1/T \end{cases} \qquad (2.3.2)$$

This is the perfect channelfrequency response when the received signal is sampled twice per signal element (see Section 2.1).

The waveform r(t) is sampled twice per data symbol at the time instants $\{iT/2\}$. Clearly, given the frequency response in Equation 2.3.2, $r(t)$ is sampled at the Nyquist rate[1]. The impulse response of the channel includes the effects of the premodulation filter, as described earlier. The sampled impulse response is $Y=[y_{-1},y_0,y_1,\ldots,y_{2g}]$ where $y_j=(\frac{jT}{2})$. The values of the y_j are not specifically given. The assumption is that the $\{y_j\}$ are such that the received signal $r(t)$ is as described earlier for the three schemes under consideration. Equations 2.3.3 and 2.3.4 define the received samples $r_{i-\frac{1}{2}}$ at time $t=(i-\frac{1}{2})T$, and r_i at time $t=iT$

$$r_{i-\frac{1}{2}} = \sum_{h=0}^{g} p_{i-h}y_{2h-1} + w_{i-\frac{1}{2}} \qquad (2.3.3)$$

$$r_i = \sum_{h=0}^{g} p_{i-h}y_{2h} + w_i \qquad (2.3.4)$$

c_i	PHASE SHIFT $\Delta\phi_i$ (Radians)
0	$-3\pi/2$
1	$-\pi$
2	$-\pi/2$
3	0
4	$+\pi/2$
5	$+\pi$
6	$+3\pi/2$

+: Anti-clockwise rotation

-: Clockwise rotation

TABLE 2.3.1: Mapping Function for Differential Phase CORPSK(4-7,1+D)

c_i	DIRECT PHASE-MAP SCHEME A PHASE ϕ_i & DIRECTION TAKEN TO p_i (radians)	DIRECT PHASE-MAP SCHEME B PHASE ϕ_i (radians) (see text for direction)
0	0 (Clockwise)	$-3\pi/2$
1	$\pi/2$ (Clockwise)	$-\pi$
2	π (Clockwise)	$-\pi/2$
3	$3\pi/2$ (Clockwise)	0
4	0 (Anti-clockwise)	$+\pi/2$
5	$\pi/2$ (Anti-clockwise)	$+\pi$
6	π (Anti-clockwise)	$+3\pi/2$

TABLE 2.3.2: Mapping Functions for Direct Phase-Map CORPSK(4-7,1+D)

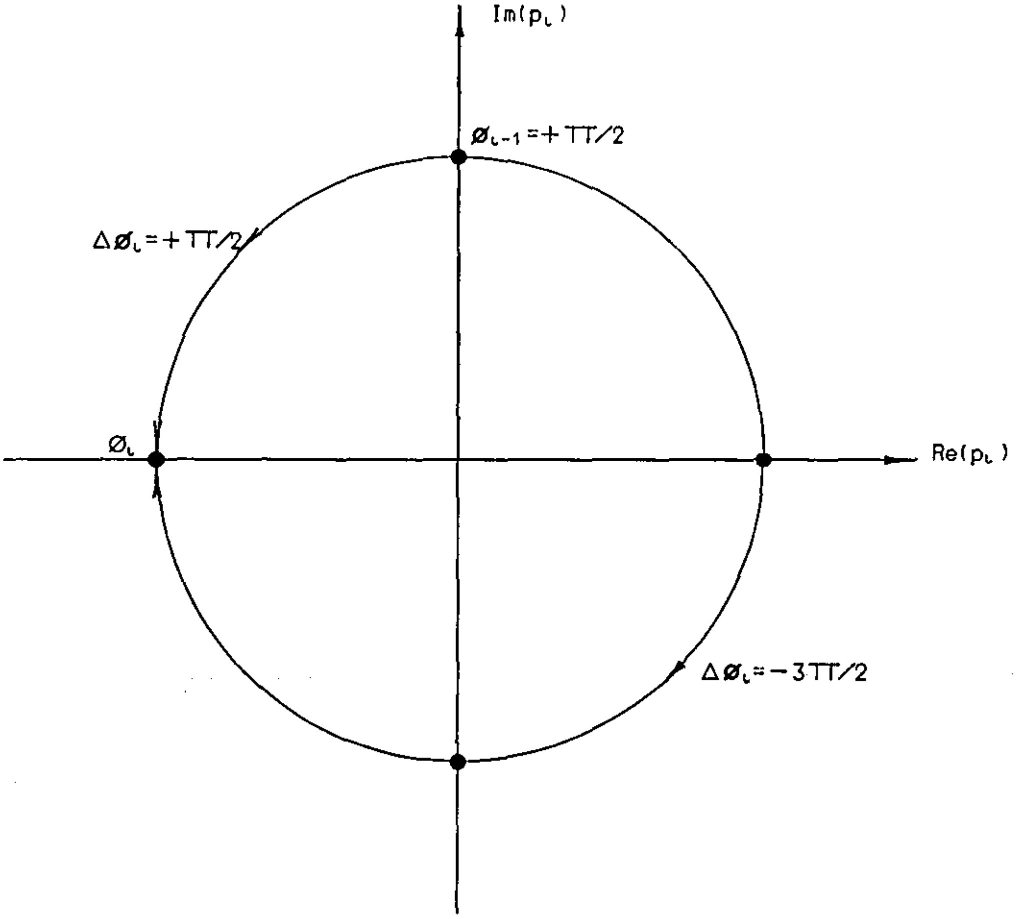


Figure 2.3.1 Example of Non-uniqueness for Differential-Phase CORPSK(4-7,1+D)

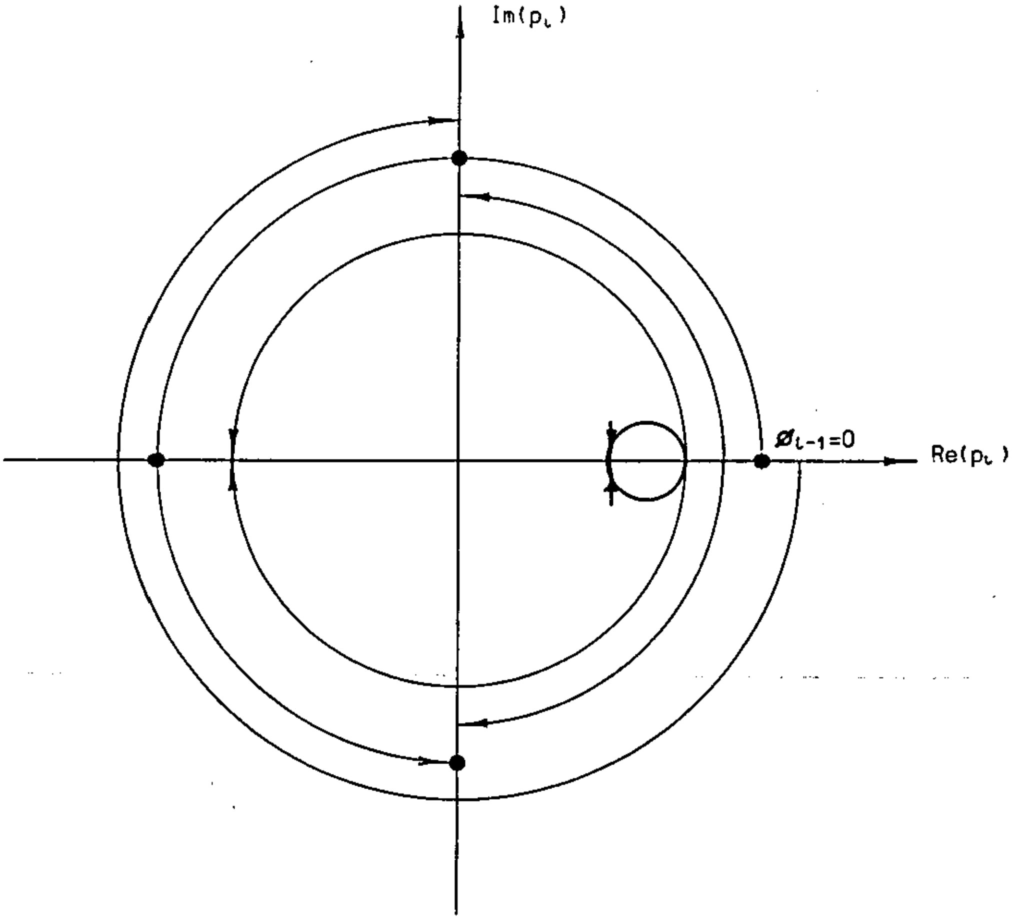


Figure 2.3.2 CORPSK(4-7,1+D) Phase Trajectories

2.4 FILTERED DIFFERENTIAL CORPSK(4-7,1+D) MODEL

The filtered CORPSK(4-7,1+D) model is a less idealised version of the differential-phase scheme described in Section 2.3. In particular the model includes specific premodulation filters to smooth the signal's phase and therefore restrict its bandwidth (Chapter 1), and a wider range of channel impulse responses. This section introduces the characteristics of the premodulation filters, which are required for a full understanding of the computer program based on this model (Appendix B4). In addition this section outlines the method by which knowledge of the impulse response Y is used in the computer program. This technique considerably reduces the execution time of the computer program. Again, precoding is retained as an option. The correlative coding rule is,

$$c_i = q_i + q_{i-1} \qquad (2.4.1)$$

Here q_i has one of the values $-1\frac{1}{2},-\frac{1}{2},+\frac{1}{2}$ and $+1\frac{1}{2}$, so that c_i has one of the values -3,-2,-1,0,1,2,3. This definition of q_i is required to facilitate the description of the premodulation filtering (see Appendix A2). q_i itself has not changed. Only its representation has changed. The modulator is an FM modulator, which includes at its input a premodulation filter with a phase characteristic described by its frequency modulating pulse $\alpha(t)$. (See Appendix A2 for the physical significance of $\alpha(t)$.) Equation 2.4.2 gives the definition of the phase response function of the premodulation filter, $\beta(t)$, derived from $\alpha(t)$.

$$\beta(t) = \int_{-\infty}^{t} \alpha(\tau)d\tau \qquad (2.4.2)$$

Appendix A2 describes $\alpha(t)$ and $\beta(t)$ in more detail.

The input to the premodulation filter is, for convenience, modelled as a sequence of phase shifts $\{\Delta\phi_i\}$ as in Section 2.3, where the mapping of the code symbols $\{c_i\}$ onto the $\{\Delta\phi_i\}$ is given in Table 2.3.1. The phase of the signal at the input to the filter before transmission begins, ϕ_O, is zero radians. The $\{\Delta\phi_i\}$ in conjunction with ϕ_O give the phase of the signal at the input to the premodulation filter at the time instants $\{iT\}$. These are the phase angles of the complex numbers $\{p_i\}$ in Section 2.1. The $\{\Delta\phi_i\}$ are taken to be the inputs to the premodulation filter in place of the $\{p_i\}$, simply because the filter is here described by its phase response function $\beta(t)$, (see Appendix A2). The output of the premodulation filter constitutes the baseband modulating waveform which is fed to the modulator. The samples of the modulating waveform at times $\{iT\}$ are complex numbers of constant magnitude (equal to 2.0 as described in Section 2.1) with phase angles $\{\phi_i\}$. It is convenient here to combine the coding and premodulation filtering in the composite phase response functio $\beta'(t)$ (see Appendix A2).

$$\beta'(t) = \beta(t) + \beta(t-T) \qquad (2.4.3)$$

The phase angles $\{\phi_i\}$ can now be defined in terms of $\beta'(t)$ and the uncoded symbols $\{q_i\}$ as below (from Appendix A2).

$$\phi_i = 2\pi h \sum_{j=1}^{\infty} q_j\beta'_{i-j} \qquad (2.4.4)$$

h is the constant modulation index[34,49] = ½ (Appendix A2) and $\beta'_j = \beta'(jT)$.

A number of premodulation filters have been incorporated in the model. Both time-limited (non-frequency-limited), and frequency-limited (non-time-limited), frequency modulating pulses $\alpha(t)$ have been considered. An example of a time-limited frequency modulating pulse is the 1RC pulse, (100% Roll-Off Raised Cosine pulse of duration T seconds), as defined by Equation 2.4.5.

$$\alpha(t) = \begin{cases} (1/2T)[1-\cos(2\pi t/T)] & , 0\leq t\leq T \\ 0 \text{ elsewhere} & \end{cases} \tag{2.4.5}$$

The stages in deriving $\beta'(t)$ from $\alpha(t)$ are given in Graphs 2.4.1 to 2.4.3. See Appendix B5 for the program which performs this operation. Graph 2.4.1 depicts the frequency modulating pulse $\alpha(t)$. The correlative-level coding is combined with $\alpha(t)$ to give the composite frequency modulating pulse $\alpha'(t)$, in Figure 2.4.2.

$$\alpha'(t) = \alpha(t) + \alpha(t-T) \tag{2.4.6}$$

$\alpha'(t)$ is integrated to give $\beta'(t)$ in Figure 2.4.3.

$$\beta'(t) = \int_{-\infty}^{t} \alpha'(\tau)d\tau \tag{2.4.7}$$

$\beta'(t)$ thus produced satisfies Nyquist's Third Criterion. This ensures that the phase of the transmitted signal with respect to the carrier at the time instants $\{iT\}$ is $\ell h\pi$ where ℓ=0,1,2, or 3 (see Appendix A2).

An example of a frequency modulating pulse which is frequency-limited, and which satisfies Nyquist's Third Criterion, is the Nyquist III-ammended 0% Roll-Off Raised Cosine pulse. For this pulse, the filter's transfer characteristic without Nyquist III ammendment is given by Equation 2.4.8.

$$\lambda_I(f) = \begin{cases} 1 & ; \quad |f|\leq 1/(2T) \\ 0 & ; \quad \text{elsewhere} \end{cases} \tag{2.4.8}$$

where f is frequency in Herz.

From Equation A2.9, the transfer characteristic after Nyquist III ammendment is given by Equation 2.4.9. Again, this ensures that the phase of the transmitted signal at the time instants $\{iT\}$ is $\ell h\pi$, where ℓ=0,1,2, or 3.

$$\lambda_{III}(f) = \begin{cases} \pi fT/\sin(\pi fT) & ; \ |f| \leq 1/(2T) \\ 0 & ; \ \text{elsewhere} \end{cases} \qquad (2.4.9)$$

This transfer characteristic is shown in Graph 2.4.4, and the corresponding frequency modulating pulse $\alpha(t)$ is given in Graph 2.4.5. $\alpha(t)$ is gained by taking the inverse Fourier Transform of the filter's transfer characteristic. Graph 2.4.6 shows the composite frequency modulating pulse $\alpha'(t)$, which includes the coding. Graph 2.4.7 depicts the composite phase response function $\beta'(t)$, produced by integrating $\alpha'(t)$ and shifting the result by +T/2 seconds. Reference (62) develops the relationship between the FM and PM implementations of the system, which necessitates this shift in the composite phase response. Both the above premodulation filters have been incorporated in the model.

Three sets of equipment filters have been incorporated in the model. The first yields the channel described by Equation 2.3.2, (the so-called perfect channel when each received signal element is sampled twice). This produces an indication of the performance degradation due to the realistic premodulation filtering, compared to the idealised implementation of Section 2.3. The second channel utilises filters designed by Mr. M.J. Fairfield of Loughborough University for the UNIVERSE and CERS projects[5-10]. The original filter impulse responses were given in Figure 2.1.2. Because of the double sampling required in the receiver (see Section 2.3), the bandwidth of these filters must be doubled to produce impulse responses with a time duration halved as compared with Figure 2.1.2. Thus for T in Figure 2.1.2 read T/2 for the wideband filters. Both the channel defined by Figure 2.1.2 (Ch=Mn) and the double-bandwidth version (Ch=Mw) have been used. The third channel is the Raised Cosine channel described by Equation 2.1.3.

As in Section 2.3, a channel impulse response, $Y=[y_{-1},\ldots,y_{2g}]$ where $y_j=y(jT/2)$ is defined which includes the effects of all the filtering. Therefore the Nyquist rate[1] (or near Nyquist rate) sampled signal at the detector input at time $t=(i-\frac{1}{2})T$ is given by Equation 2.4.10.

$$r_{i-\frac{1}{2}} = \sum_{h=0}^{g} p_{i-h}\, y_{2h-1} + w_{i-\frac{1}{2}} \qquad (2.4.10)$$

Similarly, at time t=iT, the sample at the input to the detector is given by Equation 2.4.11.

$$r_i = \sum_{h=0}^{g} p_{i-h}\, y_{2h} + w_i \qquad (2.4.11)$$

In the computer program for this model (see Appendix B4) an explicit knowledge of Y is not used in the receiver. (Knowledge of Y is required in order that the detector can form possible values of the received samples in the absence of noise.) Instead, the baseband channel is modelled as a Finite-State Machine as depicted in Figure 2.4.1. A general Finite-State Machine has a finite number of states N_s, an input symbol which can have one of a number of different values, and an output symbol which can have one of a number of different values (which usually differ from the set of possible values of the input symbol). The states may or may not have a physical meaning. In this case they do, to be defined later in this section. The symbol at the output of the machine at time t=iT, and the state of the machine Φ_{i+1} at time t=(i+1)T, are completely defined by the input symbol q_i, and state of the machine Φ_i, at time t=iT.[72] The number of states in the model is a direct function of a number of data symbols $\{q_j\}$ (where j<i), where $\Delta\phi_i$ in the absence of noise is a function of q_i and only these earlier data symbols $\{q_j\}$. The number of data symbols q_j (where j<i) involved in $\Delta\phi_i$ is termed the memory of the channel. In the case of non-time-limited baseband channels, the definition requires some qualification. In such cases the memory of the channel is unlimited, (since $\Delta\phi_i$ is dependent on all data symbols), but it is possible to truncate the channel impulse response such that the discarded components are negligible and N_s is finite. A number of different truncations, and accordingly a number of different Finite-State Machine definitions, are utilised. The state of the machine at time t=iT (an integer value) is given by Equation 2.4.12.

$$\Phi_i = 4^0 q_{i-\ell} + 4^1 q_{i-\ell+1} + \ldots + 4^{\ell-1} q_{i-1} + 4^{\ell}|\phi_{i-1}|\cdot\frac{2}{\pi} \qquad (2.4.12)$$

This is a Finite-State Machine with $4^{\ell+1}$ states where $|\phi_i|$ is the positive value (modulus) of ϕ_i. A number of values of the parameter ℓ are used. The term $4^{\ell}|\phi_{i-1}|\cdot\frac{2}{\pi}$ is included since knowledge of the initial phase, (the phase state), is required in the Finite-State

Machine in order to define the machine's output symbol. Immediately it can be seen that this definition poses problems for premodulation filtering which does not satisfy Nyquist's Third Criterion (see Appendix A2), since ϕ_{i-1} may have greater than four possible values, these values not necessarily being multiples of πh, where h is the modulation index, (see Appendix A2 and Reference 49). The definition is sound for the premodulation filters previously described, with truncation of the phase response function where necessary.

The implementation (in the computer program) of the Finite-State Machine is by way of three look-up tables. All three are addressed by the initial state Φ_i and the input symbol q_i. The first table produces the state Φ_{i+1} at its output. The other two produce possible values of $r_{i-\frac{1}{2}}$ and r_i respectively, in the absence of noise. The look-up tables are produced using a separate program which models the channel in the absence of noise (see Appendix B6).

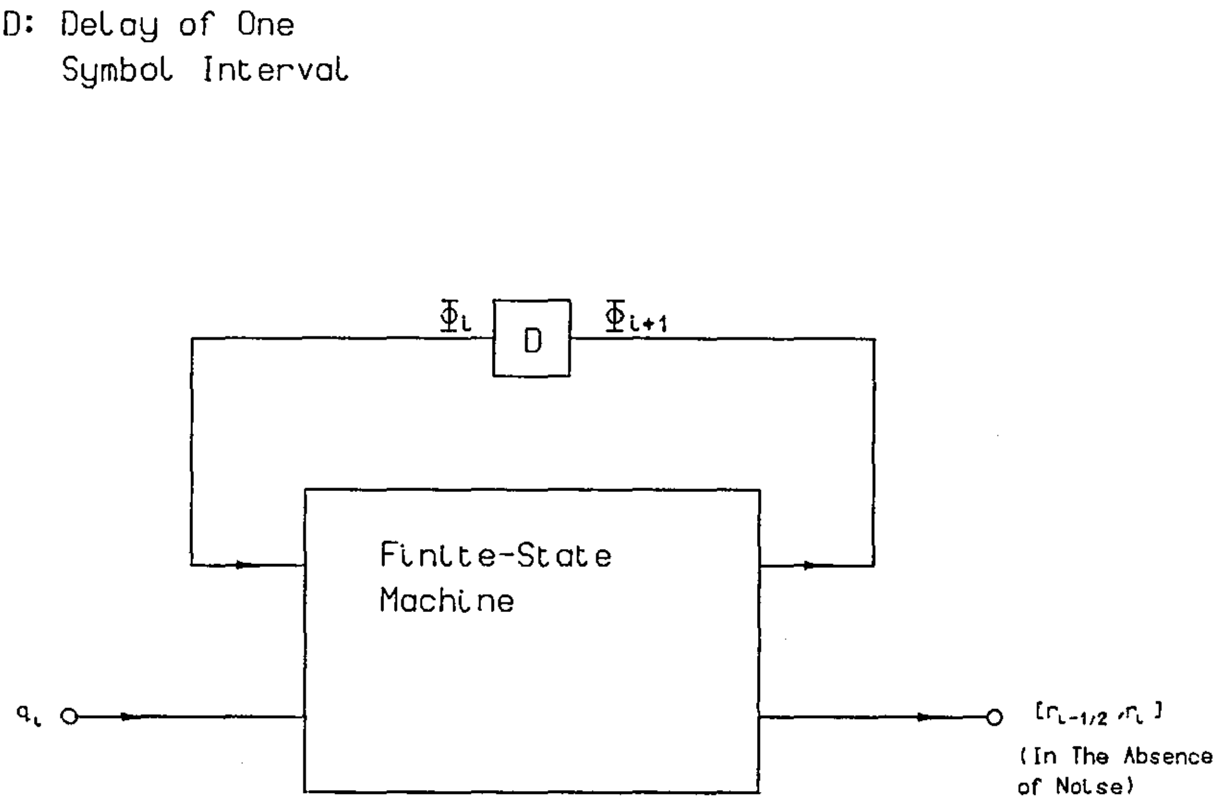
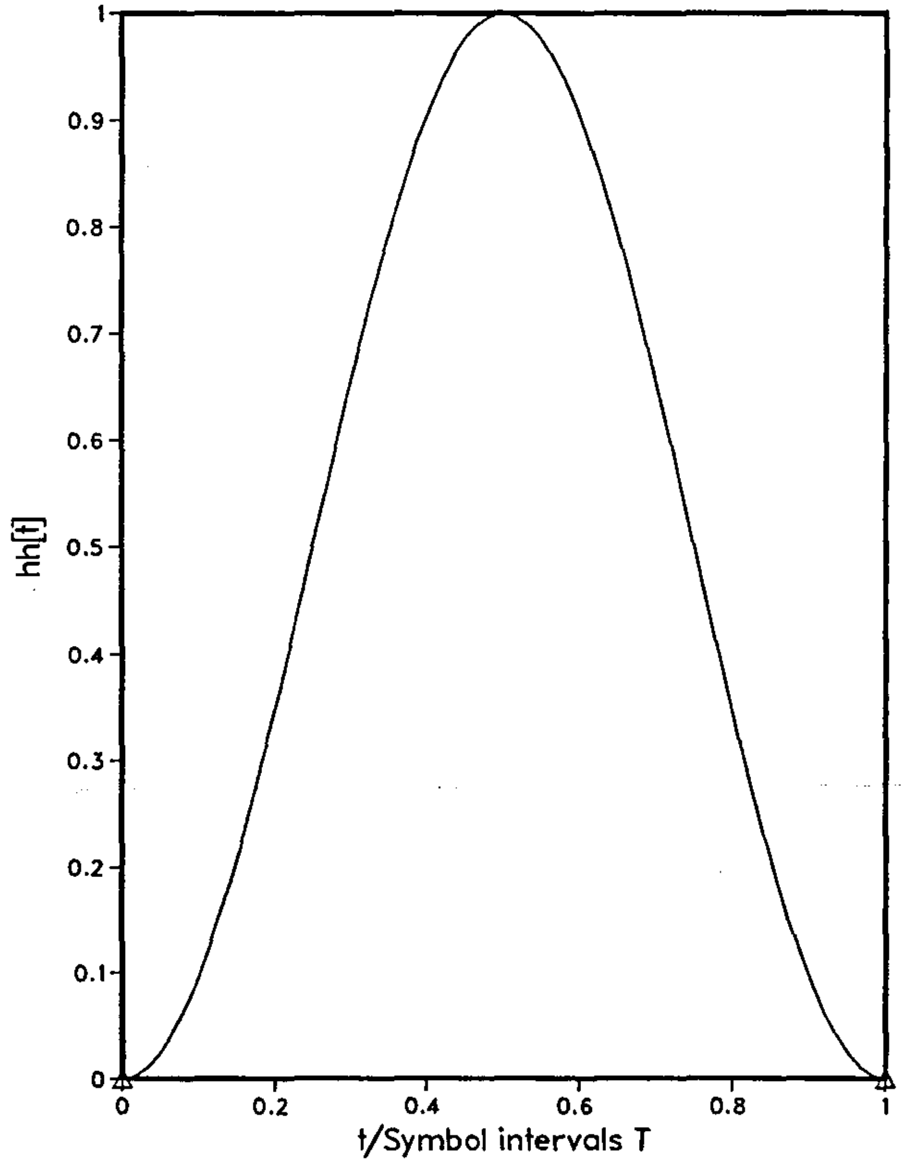
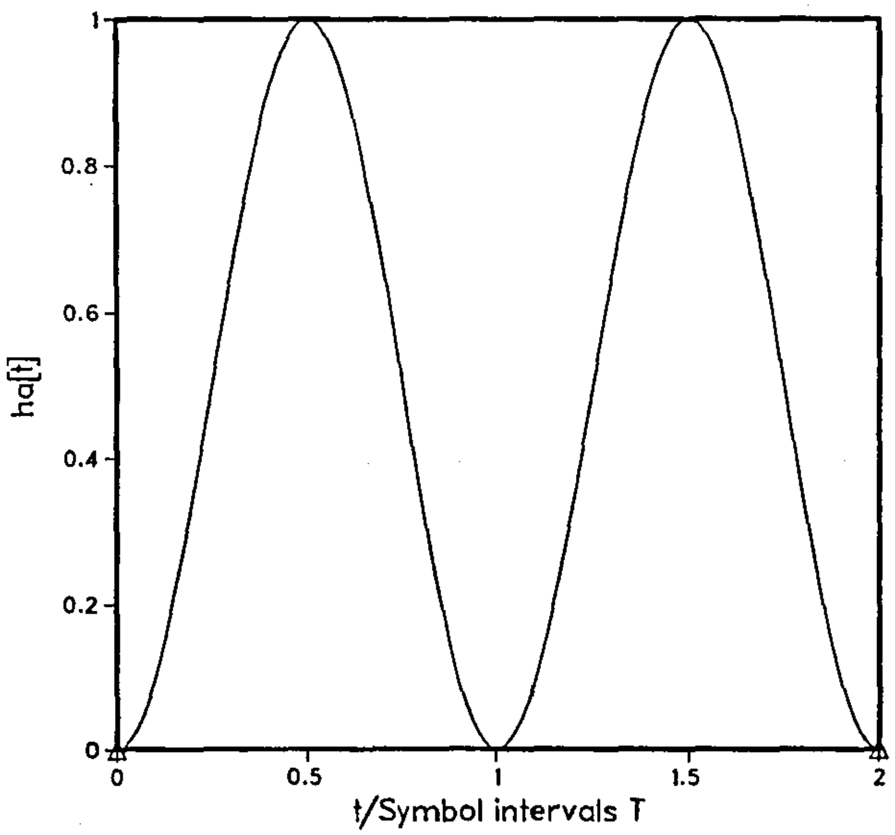


Figure 2.4.1 Finite-State Machine Structure

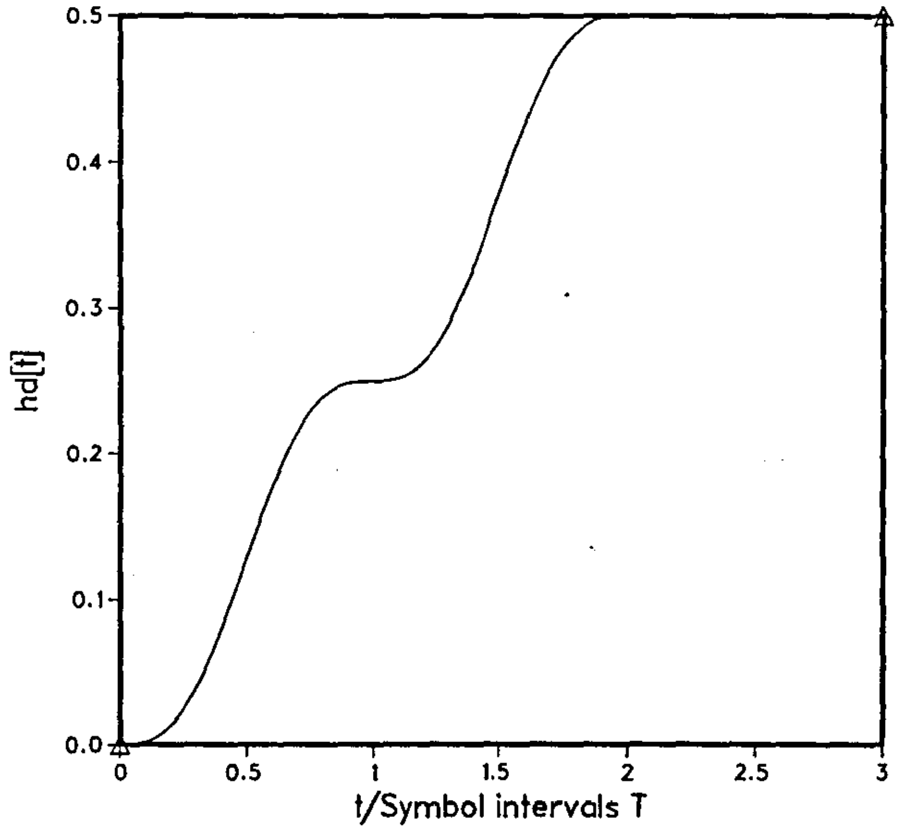
Graph 2.4.1 1RC Frequency Modulating Pulse



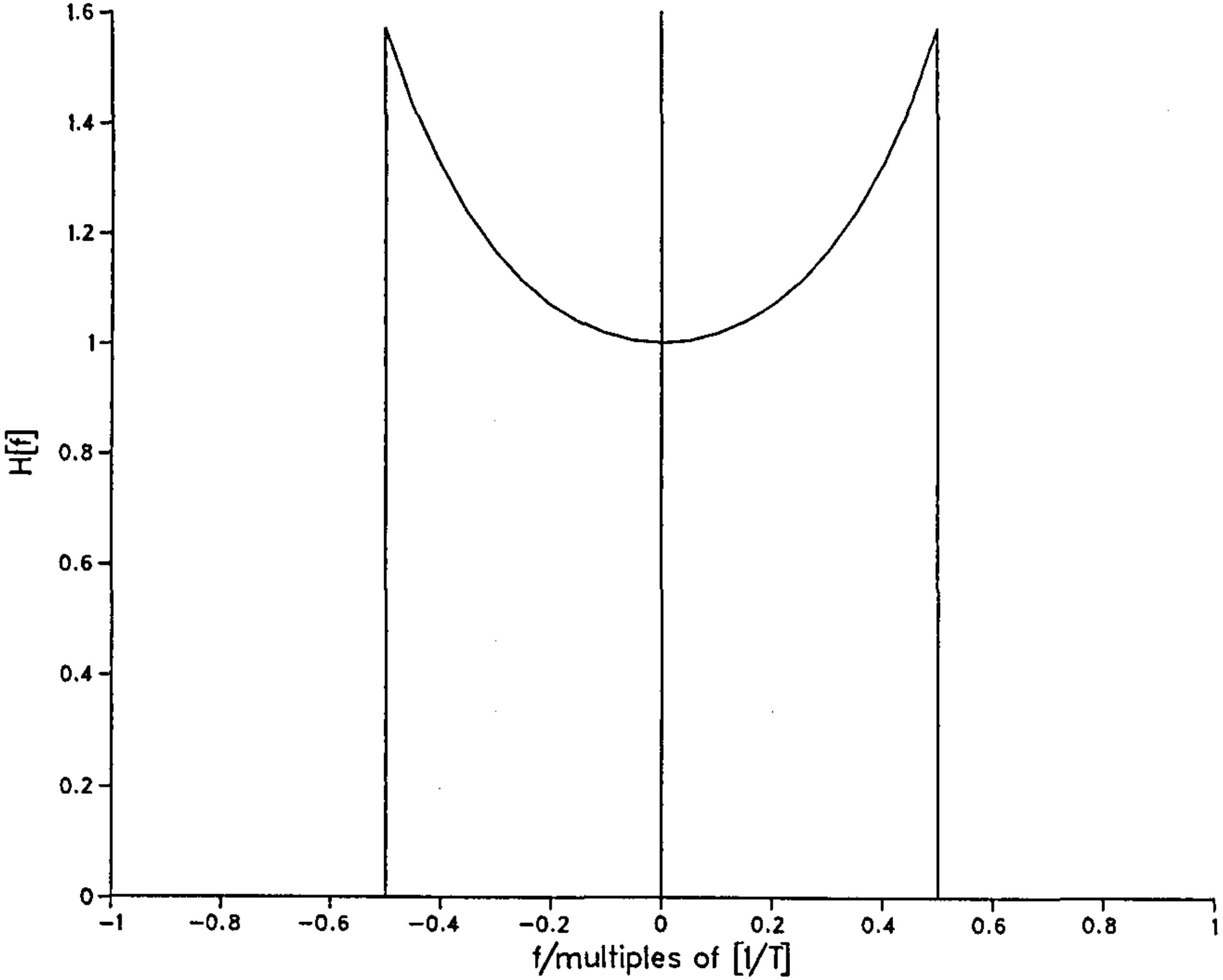
Graph 2.4.2 1RC Composite Frequency Modulating Pulse



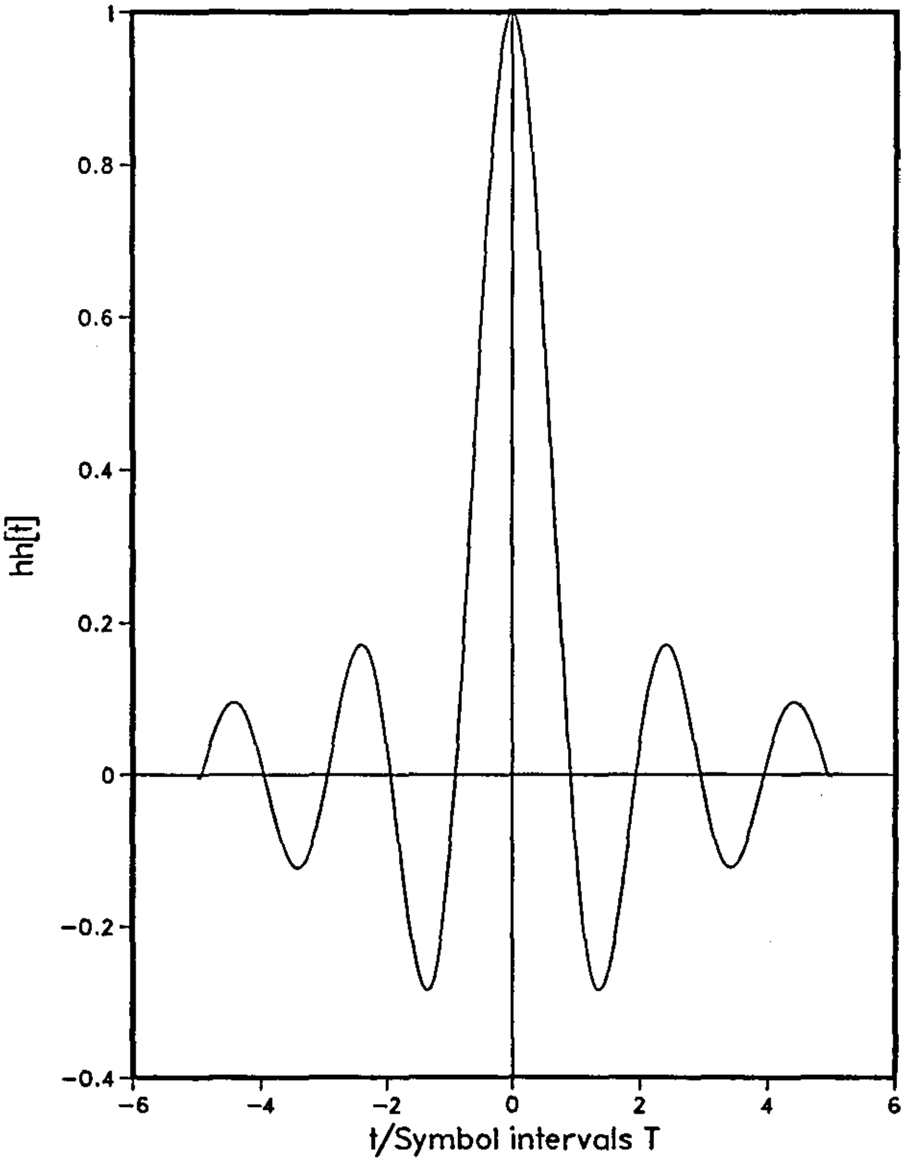
Graph 2.4.3 1RC Composite Phase Response Function



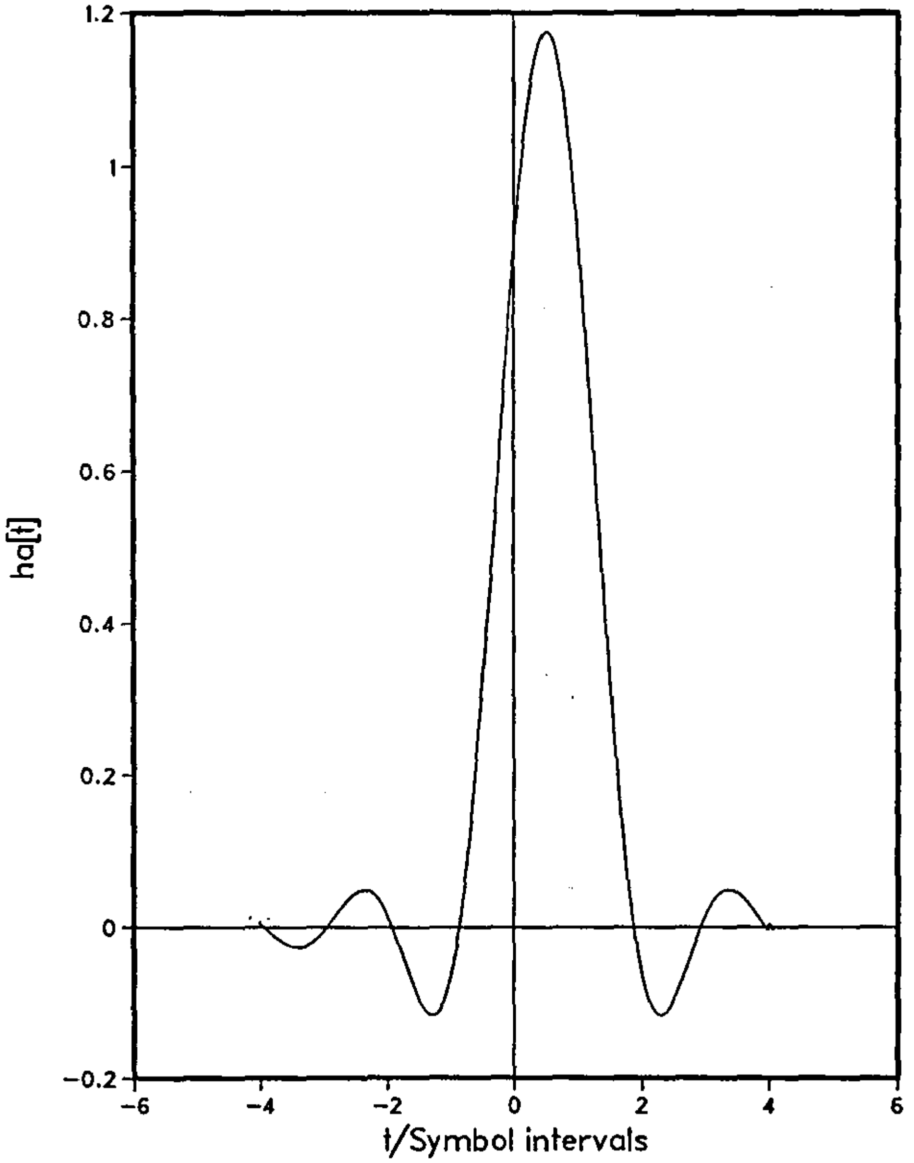
Graph 2.4.4 Nyquist III [N3] Ammended 0% Roll-Off Raised Cosine Premodulation Filter Transfer Characteristic



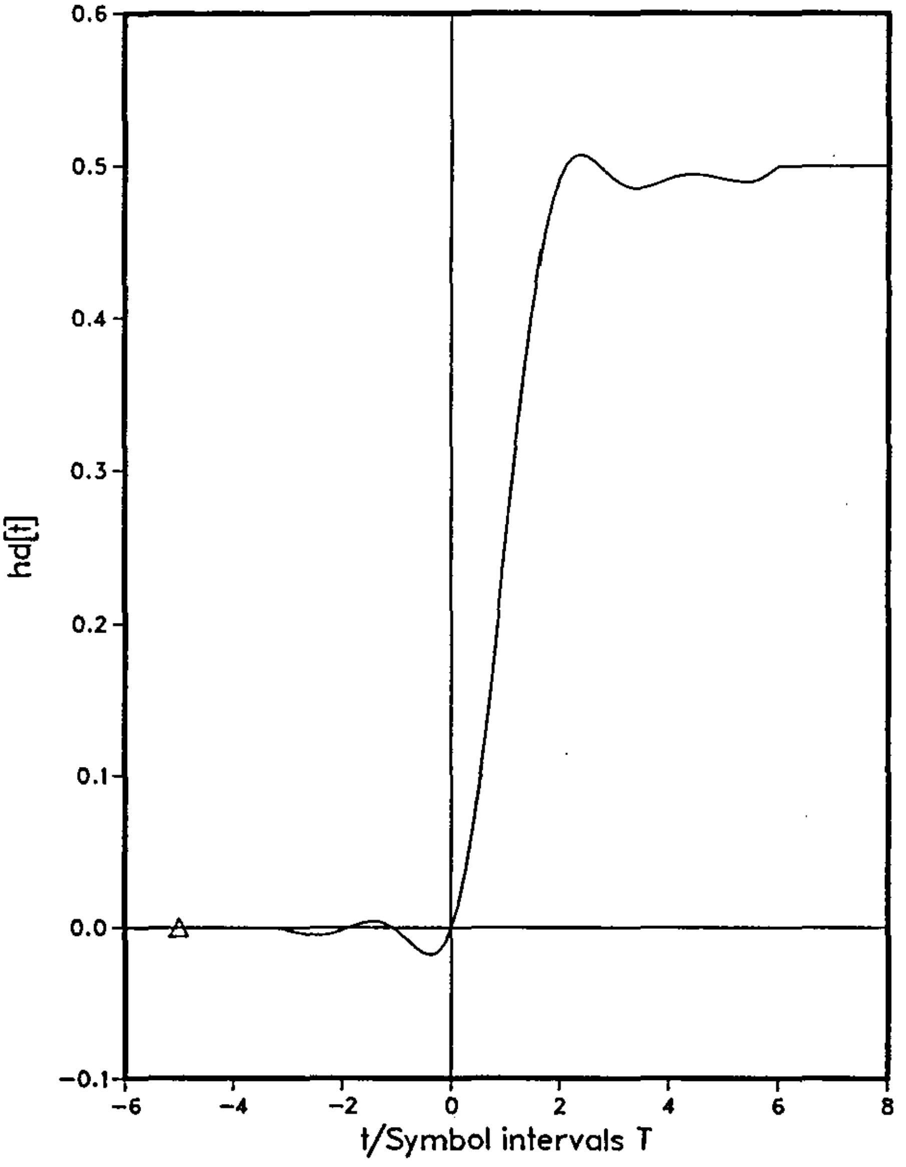
Graph 2.4.5 Nyquist III [N3] Ammended 0% Roll–Off Raised Cosine Frequency Modulating Pulse



Graph 2.4.6 Nyquist III [N3] Ammended 0% Roll–Off Raised Cosine Composite Frequency Modulating Pulse



Graph 2.4.7 Nyquist III [N3] Ammended 0% Roll–Off Raised Cosine Composite Phase Response Function



2.5 CONVOLUTIONALLY ENCODED 8PSK PERFECT CHANNEL MODEL

This model is based on Coded Trellis Modulation (CTM) introduced by Ungerboeck[20], which has since been widely studied.[12,21-26] This section briefly describes the Rate-2/3 convolutional encoder using the concept of code sub-generators defined in Appendix A4. This technique allows a conceptually simple formulation of the coder as a number of digital feedforward filters using MODULO-2 arithmetic, whose outputs are combined to produce a code symbol (see Figure 2.5.2). The coding, mapping, and baseband channel are modelled as a Finite-State Machine (Appendix A4) in order to reduce the execution time of the computer program based upon this model. Appendix A4 describes this modulation method more fully, and explains the relationship between the coding and mapping functions which is a special feature of the scheme.

The "CODER" and "MAPPING FUNCTION" blocks of Figure 2.1.1 are depicted in expanded form in Figure 2.5.1. Precoding is not applied. The Gray Code mapping is given in Table 2.1.1. Here it is used to map the 4-level data symbol q_i onto the two binary symbols $q_i(1)$, $q_i(2)$ which are the inputs to the coder at time t=iT.

The Rate-2/3 (3,2,k) convolutional encoder, where k is the constraint length, is defined by its six code sub-generators g_{ij}; i=1,2, j=1,2,3, given in vector form in Equation 2.5.1. (See Appendix A4 for a more detailed description.) The elements of each vector g_{ij} are binary-valued,

$$g_{ij} = [g_O(i,j),\ldots,g_k(i,j)] \qquad (2.5.1)$$

$$\text{for } i=1,2, \quad j=1,2,3$$

As described in Chapter 1, coding is used to improve the signal's tolerance to noise, compared with the corresponding uncoded scheme. The input to sub-generator g_{ij} at time $t=\ell T$ is the i^{th} input data stream of Figure 2.5.1, $q_m(i)$ where $m \leq \ell$. The output is a term which is one of a number of similar terms which are combined to give the j^{th} binary output symbol of Figure 2.5.1, $c_\ell(j)$. Figure 2.5.2 is the coder's block diagram.

At time t=iT, the output of the encoder is given by Equation 2.5.2 (From Equation A4.4).

$$c_i(j) = \bigoplus_{\ell=1}^{2} \bigoplus_{h=0}^{k-1} q_{i-h}(\ell) g_h(\ell,j) \qquad (2.5.2)$$

for $j=1,2,3$ where $\bigoplus$ denotes MODULO-2 summation

The implementation of the encoder takes two forms in the computer programs based on this model. Early implementations included an explicit coder at both transmitter and receiver (see for example Appendix B7). Figure 2.5.2 depicts the explicit implementation for a general Rate-2/3 code. Later programs use the Finite-State Machine[72] developed in Appendix A4 in order to speed up operation (see Appendix B8 for example). Figure 2.5.3 is a diagram of the Finite-State Machine. Using the notation developed in Appendix A4, the integer Φ_i is the state of the machine at time t=iT. The output symbol expressed as the vector of binary-valued symbols $[c_i(1),c_i(2),c_i(3)]$ at time t=iT, and the state of the machine Φ_{i+1} at time t=(i+1)T, are completely defined by the input symbol expressed as the vector $[q_i(1),q_i(2)]$, and state of the machine Φ_i, at time t=iT. The Finite-State Machine is implemented simply as two look-up tables addressed by q_i and Φ_i, where Equation 2.5.3 defines Φ_i in terms of the data symbols $q_{i-k+1}, q_{i-k+2}, \ldots, q_{i-1}$ resident in the encoder's storage elements

$$\Phi_i = 4^{k-2} q_{i-k+1} + 4^{k-3} q_{i-k+2} + \ldots + 4^0 q_{i-1} \qquad (2.5.3)$$

Clearly, from Figure 2.5.2, knowledge of Φ_i and $[q_i(1), q_i(2)]$ is sufficient to determine $[c_i(1), c_i(2), c_i(3)]$.

The first look-up table yields the vector $[c_i(1), c_i(2), c_i(3)]$ at time t=iT, while the second yields the state Φ_{i+1} at time t=(i+1)T.

The codes used are Codes 1 to 4 as defined by Hui et al.[12] Table 2.5.1 lists the code sub-generators for each code. The following rule is used to yield a single code symbol c_i from the vector of binary values $[c_i(1) c_i(2), c_i(3)]$ where c_i has one of the eight values 0,1,2, 3,4,5,6 or 7.

$$c_i = 2^2 c_i(1) + 2^1 c_i(2) + 2^0 c_i(3) \qquad (2.5.4)$$

The mapping between the 8-level code symbols $\{c_i\}$ and the complex numbers $\{p_i\}$ is defined in Figure 2.5.4. The reason for this particular mapping is discussed in Appendix A4.

The modulator and demodulator are the appropriate 8PSK types. No assumptions are made as to their configurations.

CODE	k	g_{11}	g_{21}	g_{12}	g_{22}	g_{13}	g_{23}
1	3	0 1 0	1 0 1	1 1 1	0 0 1	0 0 0	0 1 0
2	4	0 1 1 0	1 0 1 1	1 1 0 1	1 0 0 1	0 0 0 0	0 1 1 0
3	4	1 0 1 1	1 0 0 1	0 1 0 1	1 0 0 0	0 0 0 0	0 0 1 1
4	4	1 1 1 0	1 0 1 1	0 0 0 1	1 0 1 0	0 0 0 0	0 1 1 0

TABLE 2.5.1: Code Generators

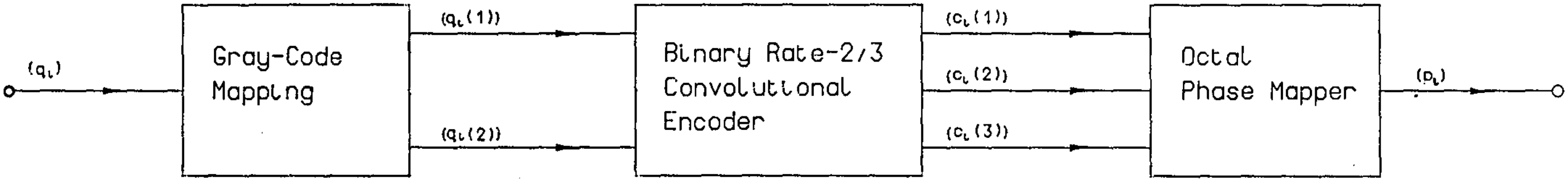


Figure 2.5.1 Coding and Mapping Function for Coded 8PSK

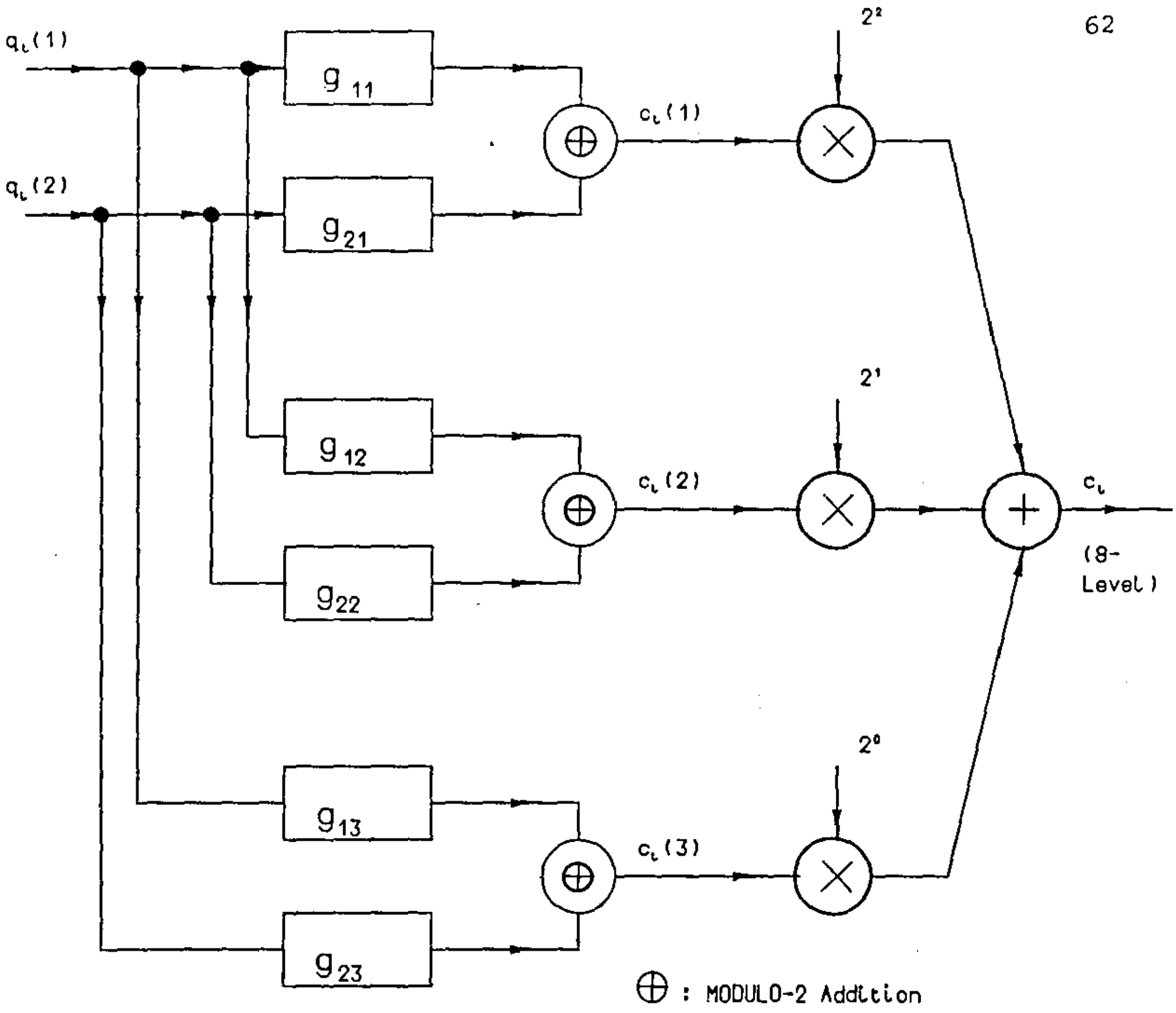


Figure 2.5.2a Rate-2/3 Convolutional Encoder

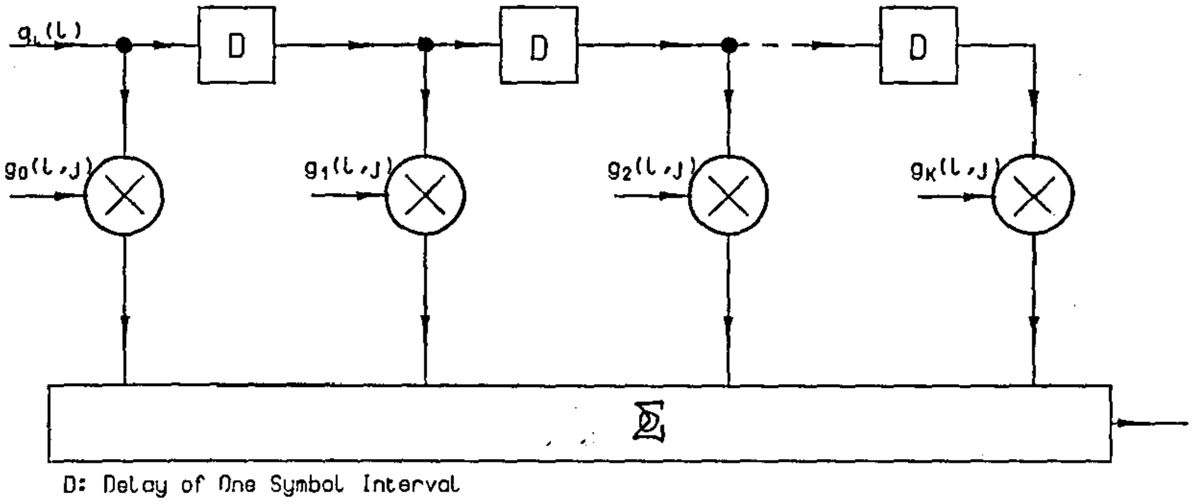


Figure 2.5.2b Implementation of g_{lj}

D: Delay of One
Symbol Interval

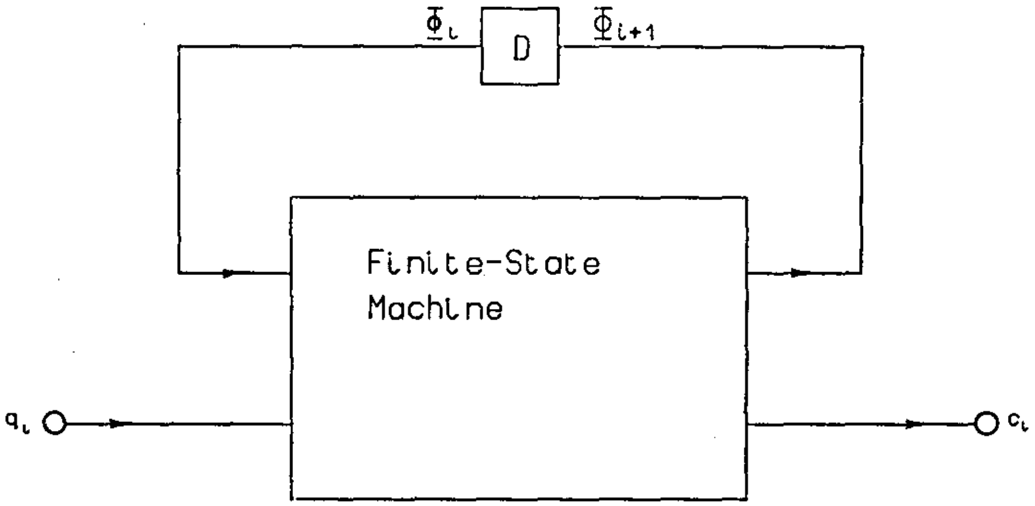


Figure 2.5.3 Finite-State Machine Structure

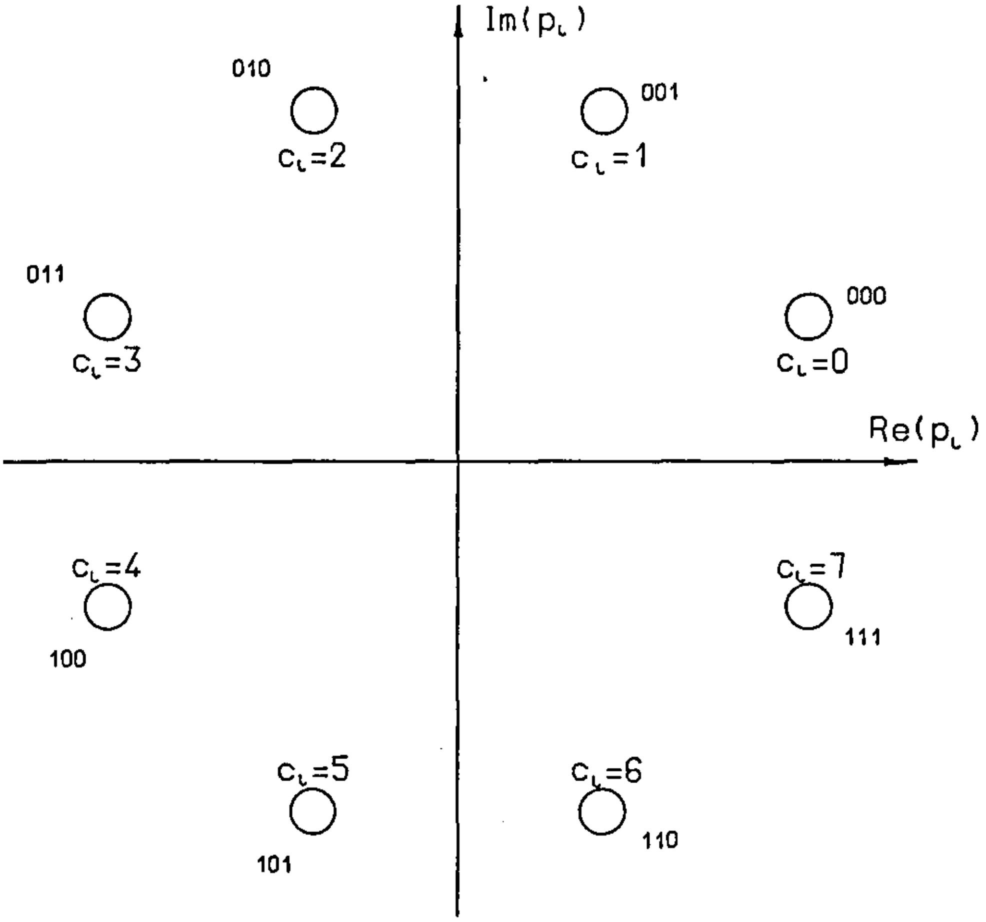


Figure 2.5.4 8PSK Signal Constellation

CHAPTER 3

OPTIMAL AND NEAR-OPTIMAL DETECTION SCHEMES

This chapter describes the detection schemes which achieve the best, or very nearly the best, tolerance to noise for the modulation schemes whose mathematical models were described in Chapter 2.

For the QPSK model of Section 2.2, a received sample r_i is a function of only one data symbol, so that simple threshold-level detection achieves the best tolerance to noise.[1,2] The detector is described in Section 3.1.

The received signal in both the coded schemes is more complex, since each received sample r_i in the absence of noise is a function of a number of data symbols. The optimum detector (achieving the best tolerance to noise) is more complex since it must consider a number of possible data symbols for each received sample r_i. The optimum detector is the Maximum Likelihood detector, implemented using the Viterbi Algorithm (see Appendix A3 and Reference 63). This is described in Section 3.2. The detector stores a number of different vectors of possible data symbols, and uses the algorithm to determine which of these is most likely to contain the values of the data symbols generated at the transmitter, given the received samples $\{r_i\}$. This vector is called the Maximum Likelihood vector, and is defined more fully in Appendix A3. For coded 8PSK modulation the number of stored vectors is a function of the number of data symbols of which each sample r_i is a function, in the absence of noise. The detector ensures that at time t=iT all possible combinations of the values of the data symbols of which r_i is a function in the absence of noise, are contained within the set of stored vectors. This organisation of the stored vectors ensures that the Maximum Likelihood vector is amongst the stored vectors (see Appendix A3, References 19 and 63, and Appendix A4). In the case of CORPSK(4-7,1+D) modulation the stored vectors contain all possible combinations of the data symbols of which the phase change between successive received samples is a function, in the absence of noise. When the Nyquist III-ammended 0% Roll-Off Raised Cosine filter is used, each such phase change is strictly a function of all data symbols (see Section 2.4). In practice only a few data symbols affect the value of the phase change significantly. All combinations of these latter data symbols are held within the set of stored vectors.

Each different combination of the above described values of the data symbols for both modulation methods is called a state. (It is important to distinguish between the meaning of a state, as used here for the stored vectors for CORPSK(4-7,1+D) modulation, and the meaning used in Section 2.4. In the latter case, the states are those of the Finite-State Machine model of the baseband channel. There, the definition of a state includes the phase of the signal at the previous sampling instance, as well as the combination of a number of data symbol values. It will be seen that the number of states in the Finite-State Machine may be varied independently of the number of stored vectors, in order to optimise performance in some way.) The concept of the state of a vector is important in the description of the Viterbi Algorithm. The mathematical definition of such states is given in Section 3.2, for both modulation methods.

Table A8.1 defines the notation which is used to describe the many variants of the schemes for which computer simulation results are presented in this chapter.

3.1 THRESHOLD DETECTION FOR QPSK AND DQPSK

In this investigation, threshold detection is used for both QPSK and DQPSK (precoded QPSK). For all the filtering arrangements (see Section 2.1), the sampled impulse response of the channel from Section 2.1, $Y=[y_0,y_1,\ldots,y_g]$, is such that only y_0 is non-zero, and is equal to one. Therefore a received sample r_i, from Equation 2.1.7, is

$$r_i = p_i + w_i \tag{3.1.1}$$

p_i is a complex number derived from the data symbol q_i using the mapping described in Section 2.2. w_i is a sample value of the Gaussian noise waveform w(t) at the demodulator output. Sections 2.1 and 2.2 describe the model more fully.

The detection process which minimises the probability of error in the detection of the data symbol q_i, from the received sample r_i at time t=iT, selects the value of q_i such that p_i is closest to r_i (see Appendix A3 and References 1 and 2). The value of p_i which is closest to r_i is found by using two thresholds in the complex number plane. The thresholds are shown in Figure 3.1.1, which includes the mapping of the $\{q_i\}$ onto the $\{p_i\}$. For example, if r_i falls into the region between the thresholds where the value of p_i is mapped from the data symbol $q_i=1$, (as shown in Figure 3.1.1), the detected data symbol, q_i', is equal to one.

The binary symbols corresponding to a particular value of q_i' are given by the Gray code mapping of Table 2.1.1. The most likely error in q_i' is that a value of p_i is chosen, which is one of the two possible values closest to the value of p_i generated at the transmitter. Such an error results in only one of the two binary symbols, given by the

Gray code mapping of q_i', being in error. The other possible error is that the chosen value of p_i is that which is furthest from the value of p_i generated at the transmitter. In this case both the binary symbols given by the Gray code mapping of q_i' are in error. This latter case is very unlikely, because a large value of w_i is required to cause it. Therefore at reasonable signal to noise ratios, the bit error rate will be only slightly greater than that for binary antipodal signalling.[1] This is $Q(\frac{d}{\sqrt{2N_O}})$[1,2], where d is the shortest Euclidean distance in the complex number plane between two possible values of p_i. ($d=\sqrt{8}$ since $|p_i|=2$ from Section 2.1.) $N_O/2$ is the two-sided power spectral density of the noise, and Q(x) is

$$Q(x) = \int_{x}^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-\tfrac{1}{2}V^2)dV \qquad (3.1.2)$$

The results are presented as graphs of bit error rate (BER) as the signal to noise ratio is varied. The signal to noise ratio is defined as E_b/N_O where E_b is the energy transmitted per data bit. Appendix A5 defines E_b/N_O for various filtering arrangements, and describes the techniques used in the computer simulations. Appendix A5 also describes a method of determining the accuracy of the results. For the results presented here, this accuracy is ±0.25dB in the range of BER, 1 in 10^3 to 1 in 10^4. Appendix A8 describes the notation used in the graphs.

The results in Graph 3.1.1 include schemes using the filters described by Equations 2.1.3 and 2.1.4, and Graph 2.1.1, of Section 2.1. It is clear that all the filtering arrangements yield very similar results. Since all the filters produce no significant intersymbol interference in the $\{r_i\}$, this is as expected. Graph 3.1.1 also shows that the technique described in Section 2.2, for repeated transmission at lower than nominal data rates, does not degrade the performance of the scheme. The results for QPSK modulation are very close to those predicted theoretically (see earlier).[1,2] The precoding in DQPSK modulation (Equation 2.1.1), usually gives two bit errors in the decoded data $\{s_j'\}$, one at time t=iT and one at time t=(i+1)T, if the complex number p_i' is wrongly chosen as one of the two possible values closest to the value of p_i generated at the transmitter. This is because, from Equation 2.1.8, both s_i' and s_{i+1}' are a function of the detected symbol q_i', (which is itself a function of the chosen value of p_i). Therefore DQPSK modulation gives a BER which is approximately twice that for QPSK modulation, at all values of E_b/N_o.

Graph 3.1.2 gives the results when phase demodulation is assumed, where the received sample is given by the phase angle $\phi(r_i)$. The decision rule is given in Table 3.1.1, where the angle is measured in an anticlockwise direction from the positive real axis. The rule is equivalent to the threshold tests previously described so that no degradation should be apparent as far as the detector is concerned. Graph 3.1.2 shows this to be the case.

RANGE OF $\phi(r_i)$ degrees (MODULO-360°)	q_i'
$315 \leq \phi(r_i) < 45$	0
$45 \leq \phi(r_i) < 135$	1
$135 \leq \phi(r_i) < 225$	2
$225 \leq \phi(r_i) < 315$	3

TABLE 3.1.1: Phase Demodulation Threshold Test Decision Rule

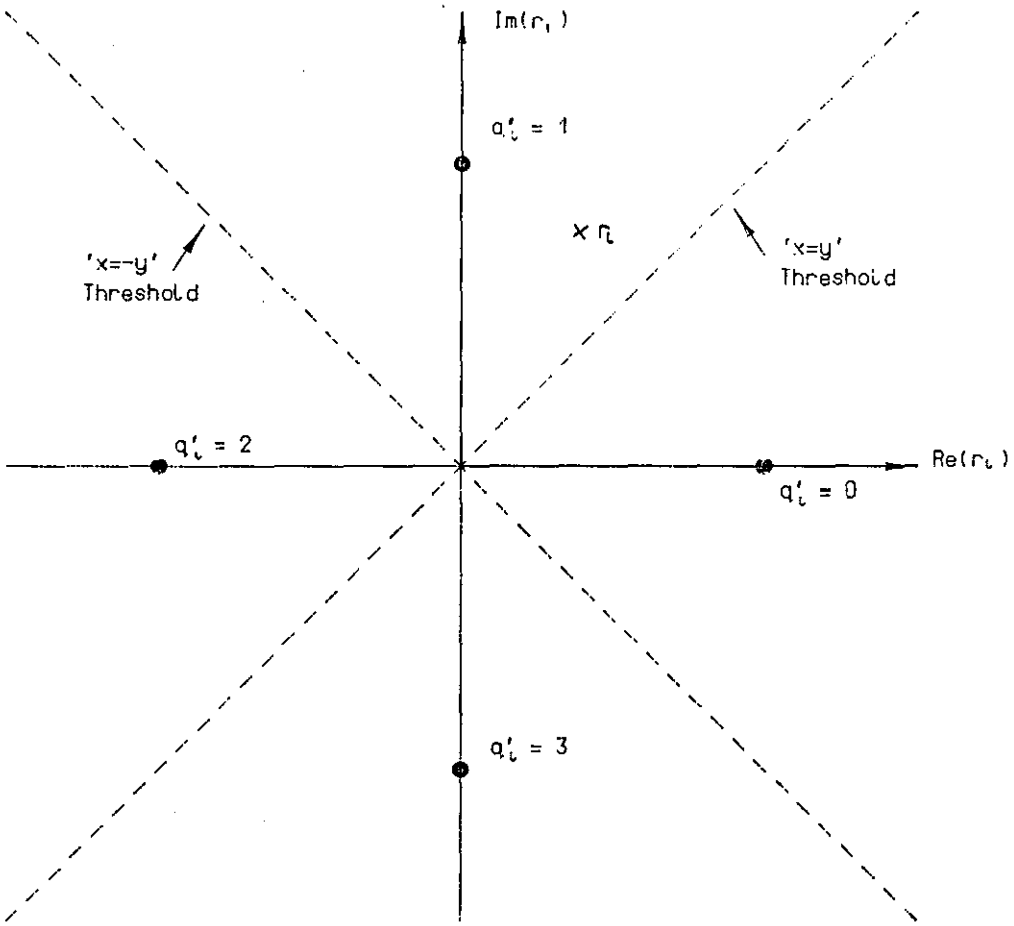
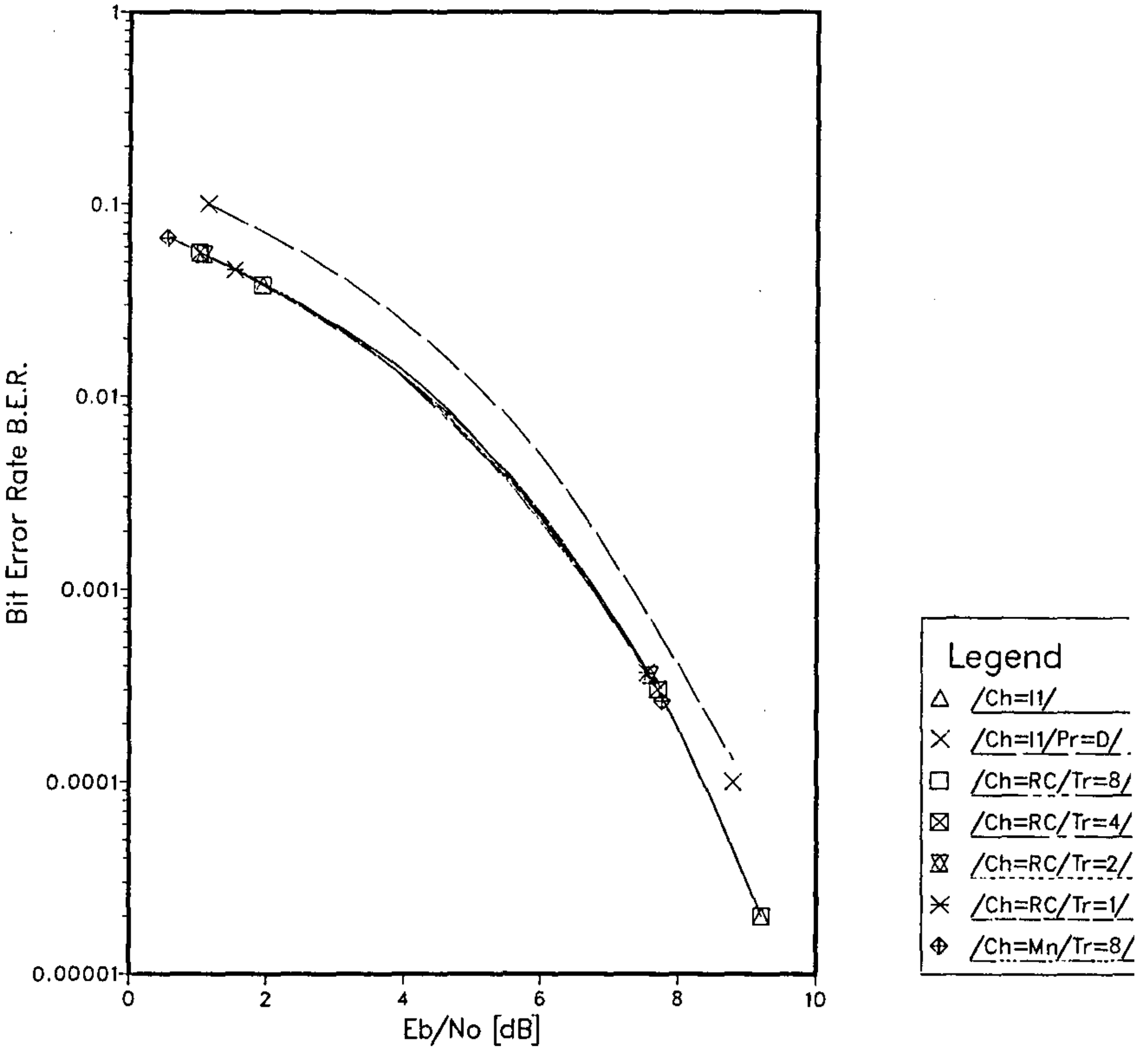


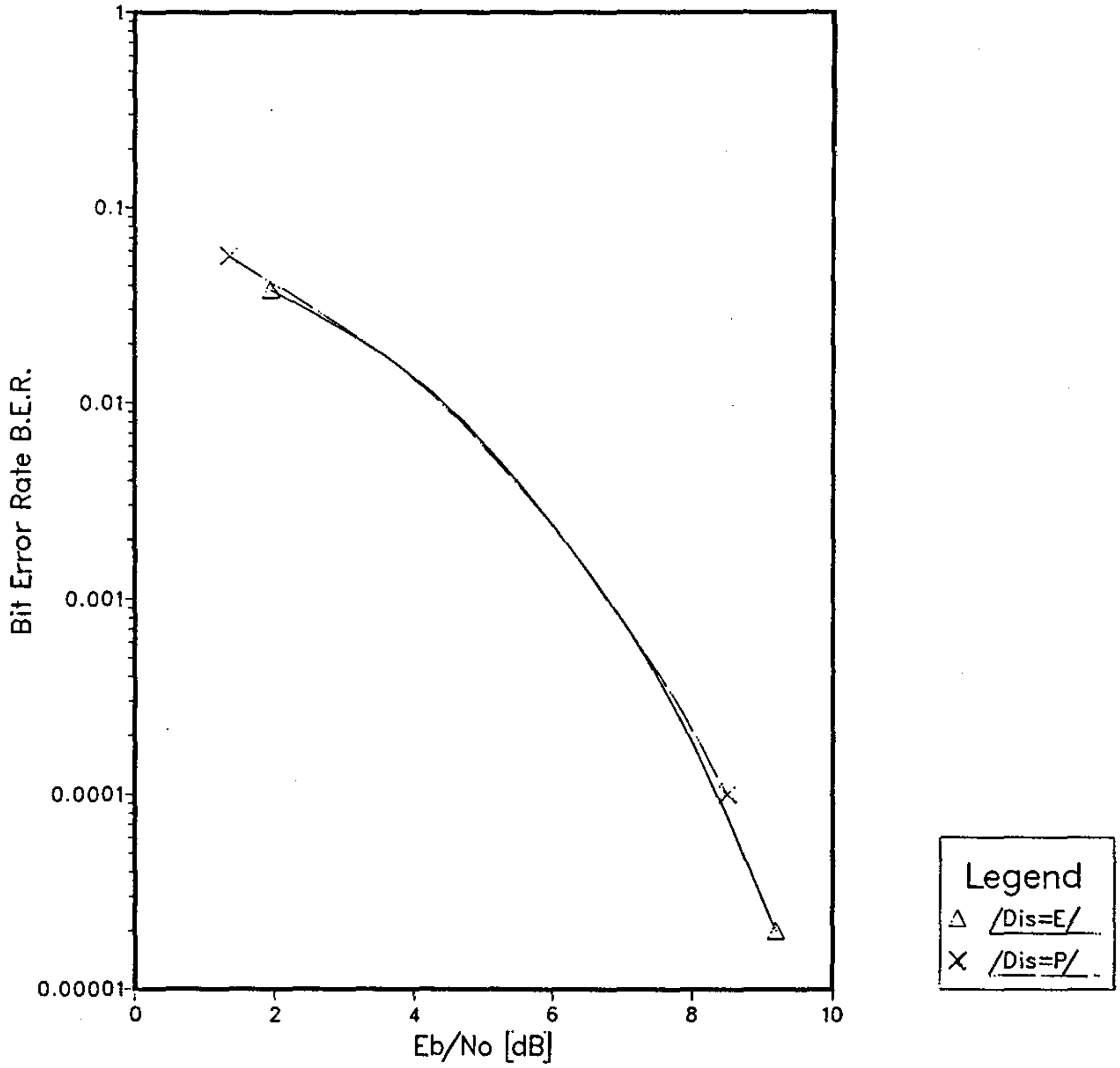
Figure 3.1.1 QPSK Constellation Thresholds

Graph 3.1.1 Uncoded QPSK. Threshold Detection



COMMON ATTRIBUTES
/M=Q/Det=T/

Graph 3.1.2 Uncoded QPSK. Phase–threshold Detection



COMMON ATTRIBUTES
/M=Q/Ch=I1/Det=T/

3.2 VITERBI ALGORITHM DETECTION FOR CODED MODULATIONS

This section describes Maximum Likelihood detection for the coded modulation methods. Initially the detector for coded 8PSK modulation is described, followed by the very similar detector for CORPSK(4-7,1+D) modulation. The mathematical models for these modulation methods were described in Chapter 2. Section 2.3 describes a simplified model for CORPSK(4-7,1+D) modulation and Section 2.4 describes a more realistic model for CORPSK(4-7,1+D) modulation. Appendix A3 gives the theory for Maximum Likelihood detection, and Appendices A2 (for CORPSK(4-7,1+D)) and A4 (for coded 8PSK) describe the modulation schemes more fully.

The description of the detectors begins with a description of the received signals. The detectors are then described in terms of the stored vectors and costs, and the algorithm repeated during every symbol interval, which uses these stored values to yield the detected data $\{q_i'\}$, is defined. In all cases, the unitary distance measure is used, (see Appendix A7). Other distance measures are also used in the computer simulation tests, and Appendix A7 should be consulted for their definitions.

Equation 3.2.1 (from Equation 2.1.7) gives the received sample at the detector input at time t=iT for coded 8PSK modulation. Since the equipment filters introduce no significant intersymbol interference (see Section 2.5), the channel's sampled impulse response is $Y=[y_0,y_1,\ldots,y_g]$ where only y_0 is non-zero, and is equal to one.

$$r_i = p_i + w_i \tag{3.2.1}$$

p_i is the complex number given by the mapping of the code symbol c_i

as described in Figure 2.5.4, and w_i is a sample value of the Gaussian noise waveform w(t) at the demodulator output. See Section 2.5 for more details.

The detector stores 4^{k-1} vectors $\{Q_i'\}$ of possible transmitted four-level data symbols (where k is the constraint length of the code). Each stored vector has a different state Φ_i, where this state is given by the combination of the (k-1) most recent four-level symbols in vector Q_{i-1}', $q_{i-k+1}', q_{i-k+2}', \ldots, q_{i-1}'$. r_i, in the absence of noise, is a function of the data symbols $q_{i-k+1}, q_{i-k+2}, \ldots, q_{i-1}$, (the state), and data symbol q_i. The stored vectors therefore cover all the possible values of the data symbols of which r_i is a function, except for the value of q_i.

Just prior to the receipt of r_i, the set of stored vectors is $\{Q_{i-1}'\}$. On the receipt of r_i the detector forms possible values of r_i in the absence of noise as follows, to be compared with r_i. Each stored vector is expanded four ways to form four expanded vectors, at time t=iT, by appending each of the four possible values of the data symbol q_i; q_i'=0,1,2, or 3. In this way 4^k expanded vectors are produced. The $\{q_i'\}$ are coded using the convolutional code described in Section 2.5, to give the vector of binary code symbols $[c_i'(1), c_i'(2), c_i'(3)]$

$$c_i'(j) = \bigoplus_{\ell=1}^{2} \bigoplus_{h=0}^{k-1} q_{i-h}'(\ell)\, g_h(\ell,j) \qquad (3.2.2)$$

for j=1,2,3.

k is the constraint length of the code and the $\{g_h(\ell,j)\}$ are binary-valued. $\bigoplus$ denotes MODULO-2 summation, and $[q_i'(1), q_i'(2)]$ is derived from the Gray-Code mapping of q_i', (see Table 2.1.1). The vector

$[c_i'(1), c_i'(2), c_i'(3)]$ is mapped onto the eight-level symbol c_i'

$$c_i' = 2^2 c_i'(1) + 2^1 c_i'(2) + 2^0 c_i'(3) \tag{3.2.3}$$

c_i' has the possible values 0,1,...,7.

A possible value of r_i in the absence of noise is given by mapping c_i' onto a complex number p_i'. (The mapping is defined in Figure 2.5.4.)

For each value of p_i' thus produced, the quantity w_i' is calculated which is a possible value of the noise sample w_i

$$r_i = p_i' + w_i' \tag{3.2.4}$$

The expanded vector which has the minimum value of the quantity

$$|W_i'|^2 = |w_1'|^2 + |w_2'|^2 + \dots + |w_i'|^2 \tag{3.2.5}$$

is the Maximum Likelihood vector.

$|W_i'|^2$ is called the cost and uses the unitary distance measure (see Appendix A7). The cost is a measure of how likely it is that a stored vector's element values are the same as those of the transmitted data symbols. A low cost is indicative of high likelihood. $|W_i'|^2$ is calculated for each expanded vector by adding the appropriate value $|w_i'|^2$ (the incremental cost), to the stored cost $|W_{i-1}'|^2$ of the vector Q_{i-1}' from which the expanded vector is derived. Clearly,

$$\begin{aligned} |w_i'|^2 &= |r_i - p_i'|^2 \\ &= [\mathrm{Re}(r_i - p_i')]^2 + [\mathrm{Im}(r_i - p_i')]^2 \end{aligned} \tag{3.2.6}$$

The Viterbi Algorithm selection process follows, which selects $4^{(k-1)}$ vectors $\{Q_i'\}$ from the 4^k expanded vectors. One expanded vector is selected for each particular state Φ_{i+1}, from the four expanded vectors with state Φ_{i+1} (see Appendix A4). (Φ_{i+1} is given by the combination of the values of the (k-1) symbols $q'_{i-k+2}, q'_{i-k+3}, \ldots, q'_i$.) The selection criterion is simply that the chosen expanded vector is that with the lowest value of $|W'_i|^2$. The resultant vectors, $\{Q'_i\}$ are stored along with their costs $\{|W'_i|^2\}$. The Maximum Likelihood vector is, as stated before, that newly stored vector Q'_i with the overall minimum cost $|W'_i|^2$. This minimum cost is subtracted from all $\{|W'_i|^2\}$ to prevent overflow in the stored values.

Ideally, no firm decision as to the value of data symbol q_ℓ is made until the end of transmission, when all the $\{q_\ell\}$ are detected simultaneously. In practice, as large a delay as possible, (N symbol intervals), is inserted before detecting q_ℓ. q_{i-N+1} is detected as the value of q'_{i-N+1} in the Maximum Likelihood vector Q'_i at time t=iT. The chosen delay, N symbol intervals, defines the number of symbols in each stored vector, since the values $q'_{i-N}, q'_{i-N-1}, \ldots$ are not required in the detection of q_{i-N+1} or any subsequent q_ℓ; $\ell = i-N+2, i-N+3, \ldots$. The $\{Q'_i\}$ are therefore of the form given in Equation 3.2.7.

$$Q'_i = [q'_{i-N+1}, q'_{i-N+2}, \ldots, q'_i] \qquad (3.2.7)$$

The process involves 4^k complex squaring operations (Equation 3.2.6), followed by 4^{k-1} cost ranking operations, each involving four costs $|W'_i|^2$. For k=3 this means 64 complex squarings followed by 16 cost rankings, whereas for k=4 there are 256 complex squarings to be undertaken, followed by 64 cost rankings. Clearly this level of complexity is prohibitive. For an example of a simulation program using the Viterbi Algorithm detector for coded 8PSK, see Appendix B9.

For CORPSK(4-7,1+D), Equations 3.2.8 and 3.2.9 give the received samples $r_{i-\frac{1}{2}}$ and r_i at times t=(i-1/2)T and t=iT respectively, where these are repeated from Section 2.3,

$$r_{i-\frac{1}{2}} = \sum_{h=0}^{g} p_{i-h}y_{2h-1} + w_{i-\frac{1}{2}} \tag{3.2.8}$$

$$r_i = \sum_{h=0}^{g} p_{i-h}y_{2h} + w_i \tag{3.2.9}$$

The sampled channel impulse response is $Y=[y_{-1},y_0,y_1,\ldots,y_{2g}]$ where $y_j=y(jT/2)$, (see Sections 2.3 and 2.4).

The detector stores 4^{ℓ} vectors $\{Q_i'\}$ where $\ell\geq 1$, ℓ depends on the coding, and premodulation and channel filtering, and is taken to be a variable quantity in the simulations. Each stored vector has a particular and different state. The state of a vector Q_{i-1}' is given by the combination of the values of the symbols $q_{i-\ell}',q_{i-\ell+1}',\ldots,q_{i-1}'$, in the vector. This meaning of a state should be distinguished from the meaning used in Section 2.4, for the Finite-State Machine model of the baseband channel. The implementation of the Viterbi Algorithm is much the same as for coded 8PSK. The major differences lie in the algorithm which gives the possible values of r_i in the absence of noise, and in the definition of $|W_i'|^2$. The elements of each expanded vector at time t=iT are coded (from Sections 2.3 and 2.4), to give a code symbol c_i'.

$$c_i' = q_i' + q_{i-1}' \tag{3.2.10}$$

c_i' is mapped onto a complex number p_i', where the mapping is given in Section 2.3 for the one differential-phase, and two direct-phase map schemes, described in that section.

As explained in Section 2.4, the coding and channel filtering giving the possible received samples (in the absence of noise), $\sum_{h=0}^{g} p'_{i-h}y_{2h-1}$ and $\sum_{h=0}^{g} p'_{i-h}y_{2h}$ for each expanded vector, is implemented by three look-up tables based on a Finite-State Machine model. The look-up tables are addressed by the appropriate initial state Φ_i (combination of the values of the symbols $q'_{i-\ell}, q'_{i-\ell+1}, \ldots, q'_{i-1}$, and the phase of the signal at time $t=(i-1)T$, ϕ_{i-1}), and symbol q'_i. (Note that the phase ϕ_{i-1} is a function of the particular stored vector, and is therefore known. See Section 2.4 for more details.) The first look-up table provides the state Φ_{i+1}. The other two tables provide possible values of $\sum_{h=0}^{g} p_{i-h}y_{2h-1}$ and $\sum_{h=0}^{g} p_{i-h}y_{2h}$; $\sum_{h=0}^{g} p'_{i-h}y_{2h-1}$ and $\sum_{h=0}^{g} p'_{i-h}y_{2h}$ respectively. For each expanded vector, the values $w'_{i-\frac{1}{2}}$ and w'_i are determined

$$r_{i-\frac{1}{2}} = \sum_{h=0}^{g} p'_{i-h}y_{2h-1} + w'_{i-\frac{1}{2}} \qquad (3.2.11)$$

$$r_i = \sum_{h=0}^{g} p'_{i-h}y_{2h} + w'_i \qquad (3.2.12)$$

w'_i is a possible value of w_i and $w'_{i-\frac{1}{2}}$ is a possible value of $w_{i-\frac{1}{2}}$.

The cost $|W'_i|^2$ is then calculated, by adding the appropriate value $|w'_{i-\frac{1}{2}}|^2 + |w'_i|^2$, (the incremental cost), to the value of $|W'_{i-1}|^2$, of the vector Q'_{i-1} from which the expanded vector is derived.

$$|W'_i|^2 = |W'_{i-1}|^2 + |w'_{i-\frac{1}{2}}|^2 + |w'_i|^2 \qquad (3.2.13)$$

where

$$|w'_{i-\frac{1}{2}}|^2 = [\mathrm{Re}(r_{i-\frac{1}{2}} - \sum_{h=0}^{g} p'_{i-h}y_{2h-1})]^2 + [\mathrm{Im}(r_{i-\frac{1}{2}} - \sum_{h=0}^{g} p'_{i-h}y_{2h-1})]^2 \qquad (3.2.14)$$

and
$$|w_i'|^2 = [\mathrm{Re}(r_i - \sum_{h=0}^{g} p_{i-h}' y_{2h})]^2 + [\mathrm{Im}(r_i - \sum_{h=0}^{g} p_{i-h}' y_{2h})]^2 \qquad (3.2.15)$$

The performance results are given as graphs of bit error rate (BER) against signal to noise ratio, E_b/N_O, where E_b is the average energy transmitted per data bit and $N_O/2$ is the two-sided power spectral density of the additive white Gaussian noise. (See Appendix A5 for more details of the simulation techniques. Appendix A8 describes the notation used in the graphs.)

The results for Viterbi detection of coded 8PSK signals are presented in Graphs 3.2.1 to 3.2.3. Those for CORPSK(4-7,1+D) signals appear in Graphs 3.2.4 to 3.2.8.

Graph 3.2.1 presents the results for all four convolutional codes which were used (see Table 2.5.1), and contrasts these with the curve for threshold detected QPSK. All the coded schemes gain significantly in tolerance to noise over QPSK at a bit error rate of 1 in 10^4. Table 3.2.1 outlines the results at this BER. The accuracy of the curves in the range of BER, 1 in 10^3 to 1 in 10^4, is ±0.25dB (see Appendix A5). Code 1 yields a theoretical asymptotic gain, (that is, the gain at high signal to noise ratios), of 4.1dB, while the remaining codes yield theoretical asymptotic gains of 5dB, over uncoded QPSK. From Table 3.2.1 the shortfall in the actual gain compared to its asymptotic value at a BER of 1 in 10^4 is quite significant, especially for the constraint length k=4 codes (Codes 2 to 4). The result of this is that the much simpler system using Code 1 (k=3) compares very favourably, at practical signal to noise ratios, with the schemes using the longer constraint length (k=4) codes. The detectors for the schemes using the codes with k=4 can be considered to be roughly a factor of four more complex than the scheme using Code 1. The differing performances of the schemes using the codes where k=4, down to a BER of 1 in 10^4, imply that, although the codes have the same asymptotic gain, their distance profiles differ. A code's distance profile is a measure of how quickly the distance between two sequences of code symbols increases.[73] The two code sequences are those, of all possible code sequences, where this distance is a minimum given that the two sequences differ in their first symbol. This measure clearly has a bearing on the performance of coded schemes. For example, if the distance profile increases only slowly, the costs of the two code sequences as defined above, where one is the correct sequence, may remain very similar over quite a long period of time. This affects the probability of discarding the correct sequence over this period of time. An interesting comparison is given by the error burst characteristics of the different schemes (where the definition of an error burst is given in Appendix A5), as outlined in Table 3.2.2. The trend is that schemes using the shortest constraint length code, Code 1 with k=3, produce the lowest number of bit errors per burst overall. The scheme using Code 3, whose performance resembles that of Code 1 most closely, produces the next lowest number of bit errors per burst.

Graph 3.2.2 gives the results for the scheme using Code 3 as the detection delay, N, is reduced. Clearly a reduction in N from 80 to 23 symbol intervals causes the relatively low reduction of 0.25dB in tolerance to noise at a BER of 1 in 10^4. As N is reduced further to 7 symbol intervals, the degradation increases substantially. At a BER of 2 in 10^3 the degradation in tolerance to noise is 1.35dB. The reasons for this are most clearly seen by considering the code trellis diagram for the detector. This diagram is essentially a graph of the state of a stored vector Φ_i (vertical axis), as it varies with time in symbol intervals (horizontal axis). The state Φ_i is an integer which is a function of the data symbols $q'_{i-k+1}, q'_{i-k+2}, \ldots, q'_{i-1}$, (see Section 2.5). The code trellis diagram gives the $\{\Phi_j\}$ for each stored vector, over a period of time up to the current time instant t=iT. Each line in the diagram is for one of the 4^{k-1} stored vectors. More details are given in Appendix A4. Figure 3.2.1 is a code trellis diagram which was produced during a computer simulation test for coded 8PSK using Code 1. When the current received sample is r_i, this diagram shows that all the vectors have the same state at time t=(i-23)T. That is, amongst all the stored vectors at time t=iT, the state Φ_{i-23} is fixed. (In practical terms this means that the contents of all vectors for t<(i-23)T are identical). The convergence of the states is much less marked overall at time t=(i-20)T, so that a small number of different states $\{\Phi_{i-20}\}$, occur among the stored vectors at time t=iT. At time t=(i-7)T, convergence is minimal, so that many of the 4^{k-1} possible states occur among the vectors. Many of the vectors contain different data symbol values at time t=(i-7)T. A detection delay of 7 symbol intervals is too short, since the detector chooses a value of q'_{i-7} before all the vectors contain this value. Therefore, the detector may not be choosing the value of q'_{i-N+1} from the Maximum Likelihood vector. This explains the degradation in performance as N is reduced.

Graph 3.2.3 gives the results when the phase distance measure is used for a scheme using Code 3. This is one of the distance measures proposed in Appendix A7 to reduce the complexity of the detector. It is simply the difference in the phase angles of r_i and a possible received sample in the absence of noise, (where this difference is MODULO-180° so that the maximum difference is 180°). The degradation in tolerance to noise is quite substantial, (0.8dB at a BER of 1 in 10^4), and the squared phase distance measure produces no advantage. The conclusion is that the relative sizes of the vectors' costs are considerably altered by this simple distance measure, compared with the use of the unitary distance measure, and that the Viterbi detector is relatively sensitive to these changes. A comparison in terms of the systems' error burst characteristics is given in Table 3.2.3. It can be seen that there is very little variation in the results, in comparison with the use of the unitary distance measure.

Graphs 3.2.4 to 3.2.6 give the results for the perfect channel CORPSK(4-7,1+D) model described in Section 2.3. The accuracy of the curves is ±0.25dB, for the range of BER, 1 in 10^3 to 1 in 10^4. In all cases four vectors are stored ($\ell=1$), since in this simple model the phase change over the time interval, $(i-1)T\leq t\leq iT$ is a function of only q_{i-1} in addition to q_i.

Graph 3.2.4 gives results for both precoded and non-precoded versions of the three systems; differential-phase, and direct phase-map schemes Ph=Ma and Ph=Mb, with DQPSK as the reference scheme. Table 3.2.4 gives the performance comparisons at a BER of 1 in 10^4. It can be seen that there is no significant difference between the precoded and non-precoded schemes at a BER of 1 in 10^4. At higher error rates the case is somewhat altered. For the differential-phase scheme, a significant advantage is gained by precoding for error rates in excess of 1 in 10^4. This is also true, but even more so, for the Ph=Ma direct map scheme. At a BER of 1 in 10^2, the gain for the precoded version of the latter scheme, in comparison with the non-precoded scheme, is 0.5dB. In all cases, the curves for the precoded and non-precoded schemes are converging as the BER reduces. This phenomenon can be explained by considering the error burst characteristics given in Table 3.2.5. The precoding has consistently reduced the average number of errors per burst for all schemes. This is particularly marked in the case of the Ph=Mb direct map scheme. The mechanism by which precoding achieves this is that the coding in the signal is effectively removed by the precoding, if the coded data is interpreted MODULO-4. Coding increases the number of bit errors per burst because if the detector chooses a wrong value for one code symbol c'_j, this will affect the values of more than one of the detected data symbols $\{q'_i\}$. (See Appendix A1 for a full treatment of precoding as applied to CORPSK(4-7,1+D) modulation.) The improvement shown in Table 3.2.5 for the differential phase scheme is much less marked, (as in Graph 3.2.4). This may be due to the fact that in this case each value of p_j is a function of all previous data symbols $\{q_i\}$, because the coded symbols are mapped onto phase shifts (see Section 2.3).

Graph 3.2.5 presents results for the use of the phase distance measure for the differential-phase and direct phase-map Ph=Mb schemes. Clearly severe degradation is introduced. Table 3.2.6 gives the results at a BER of 1 in 10^4. The error burst characteristics are outlined in Table 3.2.7. It is clear that there is little variation in the error burst characteristics, in comparison with the use of the unitary distance measure.

Graph 3.2.6 shows the degradation which occurs when realistic quantisation of the received signal is assumed. Results for both 3 bit and 4 bit quantisation per in-phase or quadrature component are contrasted with those for infinitely fine quantisation. At a BER of 1 in 10^4, the degradations in tolerance to noise due respectively, to 4 bit and 3 bit quantisation, are 0.25dB and 0.5dB. This could be significant if deep signal fades occur, when the quantisation may effectively fall to one or two bits per component. Automatic gain control is usually needed to prevent this. The error burst characteristics are given in Table 3.2.8. The results show a trend towards an increase in the number of errors per burst as the quantisation becomes coarser.

Graphs 3.2.7 and 3.2.8 give the results for the differential-phase CORPSK(4-7,1+D) model of Section 2.4 incorporating both premodulation and channel filtering. For these graphs the accuracy is ±0.3dB over the range of BER, 1 in 10^3 to 1 in 10^4. The description of the detector earlier in this section noted the use of look-up tables to give the values of possible received samples at times $t=(i-\frac{1}{2})T$ and $t=iT$, in the absence of noise. The number of states in the model for these look-up tables can be varied independently of the number of stored vectors, (see Section 2.4). The results include variations in the number of stored vectors, and in the number of states in the model for the look-up tables. For example, the detector designated /Det=V4,16/, (from Appendix A8), has 4 stored vectors, and look-up Tables based on a model with 16 states.

In Graph 3.2.7 the effect of the premodulation filter is considered using the Ch=I2 channel (Appendix A8). Pf=1RC (100% Roll-Off Raised Cosine), and Pf=N3 (Nyquist III-ammended 0% Roll-Off Raised Cosine) premodulation filtering is used. Both precoded and non-precoded schemes are included. The relative performance of these schemes at a BER of 3 in 10^4 is given in Table 3.2.9. It is clear that precoding has again improved the performance of the schemes under consideration. For the Pf=1RC-filtered schemes the difference is substantial, 0.5dB at a BER of 3 in 10^4. The result of increasing the number of detector-held vectors from 4 to 16, while keeping the number of states in the model of the look-up tables constant at 16, is of interest. For the Pf=1RC-filtered schemes with no precoding, there is a relatively large difference at high error rates, but the curves tend to converge at lower error rates. For the precoded Pf=1RC-filtered schemes there is no apparent difference. The Pf=N3-filtered schemes have a small gain in tolerance to noise when 16 vectors are held compared with 4 vectors, but this amounts to no more than 0.1dB at a BER of 3 in 10^4. It is interesting to note that the Pf=1RC-filtered schemes gain in tolerance to noise compared with the simple model (Pf=0, see Section 2.3), when precoding is applied, whereas the Pf=N3-filtered schemes are considerably degraded. Clearly the implementation of the Pf=1RC filter produces some advantage over the Pf=0 model. It is difficult to pinpoint what this could be. The Pf=1RC-filtering is such that the received sample at time $t=(i-\frac{1}{2})T$ is midway between the received samples at times $t=(i-1)T$ and $t=iT$, on the signal envelope (see Section 2.4). This is also true for the differential-phase scheme of Section 2.3, so one would expect both schemes to produce similar results.

From Graph 3.2.7 the curves for the two schemes are nearly identical down to a BER of 1 in 10^3. At a BER of 3 in 10^4 the difference in tolerance to noise is only 0.2dB which may be due to the accuracy limits (see earlier). The Pf=N3-filtered schemes are considerably degraded in tolerance to noise compared with the Pf=0-filtered scheme, (0.6dB at a BER of 3 in 10^4). This degradation may be due to the smoothing action of the premodulation filter on the phase, which reduces the minimum distance between possible received sequences of samples, in the absence of noise. This is shown by the samples at times $\{(i-1/2)T\}$, which are nearer to one or other of the samples at times $t=(i-1)T$ and $t=iT$, in the absence of noise. (Note that the curve for /Pf=N3/Pr=D/ agrees very closely with Muilwijk's result[62].) In contrast the smoothing action of the Pf=1RC-filtered schemes is not apparent in the samples at times $\{(i-1/2)T\}$, so that the minimum distance remains unaltered. The major difference between the schemes lies in their effective bandwidth. The smoothing action of the Pf=N3-filtered scheme leads to a signal with a much narrower bandwidth compared with the Pf=1RC-filtered scheme.[34,49,62] The error burst characteristics are noted in Table 3.2.10. The average number of bit errors per burst converges, (to 3 approximately), for all the precoded schemes. Clearly precoding is useful.

Graph 3.2.8 gives the results for the cases where channel filtering, differing from the Ch=I2 arrangement, is used. Curves for the scheme where Pf=0 and Ch=I2, and for DQPSK are included. Overall, the degradation in tolerance to noise is severe. Table 3.2.11 lists the degradations in tolerance to noise with respect to the scheme where Pf=0 and Ch=I2 at a BER of 3 in 10^4 at which point the scheme where

Pf=O and Ch=I2 gains 1.6dB in tolerance to noise over DQPSK . Clearly this degradation limits the possible gain in tolerance to noise over DQPSK at a BER of 3 in lO^4, to little over ldB, which is a severe reduction compared with the asymptotic gain of 2dB quoted by Muilwijk.[62] It is interesting to note though, that no significant extra degradation is introduced over the /Ch=I2/-filtered schemes of Graph 3.2.7, in the case of the wideband (/Ch=Mw/) filters (see Section 2.4). Clearly a data rate of 8M bits per second, using the narrower (/Ch=Mn/) filters, produces results which are worse than those for DQPSK below a BER of 1 in lO^3. On the other hand, the test at 8M bits per second over the Raised Cosine channel, (/Ch=Rc/), results in a much smaller degradation in tolerance to noise. This is O.4dB at a BER of 1 in lO^3. An interesting comparison involves the complexity of the Viterbi detector. There is apparently very little to be gained by increasing the number of stored vectors from 4 to 16. Therefore the 4-vector scheme seems adequate. On the other hand, changing the number of states in the model for the look-up tables from 16 to 64, makes a significant difference at bit error rates below 1 in lO^3. At a BER of 3 in lO^4 the gain in tolerance to noise is O.2dB. Since the increased complexity in using 64 states lies mainly in the storage hardware, and not in implementation speed, it may well be an advantage to use look-up Tables based on the model with 64 states. Note though, that the accuracy at a BER of 3 in lO^4 is such that the O.2dB advantage may be inaccurate.

The remaining graphs in this section give an idea of the effect of reducing the detection delay (N), and the effect of phase offsets in the received samples due to incorrect carrier phase tracking at the receiver. Graph 3.2.9 charts the effect of reducing N for coded 8PSK modulation using Code 1, for Viterbi detection using 16 stored vectors. It can be seen that the variation in the BER is negligible as N is reduced to 20 symbol intervals. For N<20 symbol intervals the BER rises rapidly. In contrast, the result of reducing N for the scheme using Code 3 is more serious, see Graph 3.2.10. The BER rises rapidly for N<30 symbol intervals. This is because at time t=iT the contents of the stored vectors for the scheme using Code 3 remain different over a longer span of symbols $\{q_j\}$, where $j \leq i$, than do the contents of the stored vectors for the scheme using Code 1. This is influenced by the distance profiles of the codes, as discussed earlier. In contrast Graph 3.2.11 shows that N can be much smaller in the differential-phase CORPSK(4-7,1+D) scheme. Little degradation occurs for N≥8 symbol intervals. Each code symbol is a function of only two data symbols, and the distance between possible code sequences which differ in the value of their first symbol, increases quickly. The difference in the costs of such code sequences increases quickly, so that one with a high cost is likely to be discarded quickly. Therefore at time t=iT all vectors are probably derived from just one vector at time $t=(i-\ell)T$, where ℓ is relatively small. This explains why such a small delay in detection is sufficient.

The remaining plot, Graph 3.2.12, gives the results for a constant (but unknown) error in the receiver estimate of carrier phase for 4-vector Viterbi detection, and CORPSK(4-7,1+D) modulation. The effect on the BER is both large and reasonably linear, for phase errors greater than 5°.

CODE	GAIN IN TOLERANCE TO NOISE OVER QPSK AT BER = 1 × 10^{-4} (dB)
1	2.8
2	3.25
3	3.1
4	3.15

TABLE 3.2.1: Performance of Coded 8PSK using Codes 1 to 4 for Viterbi Algorithm Detection

CODE	AVERAGE NUMBER OF BIT ERRORS PER BURST AT GIVEN BER (Approximate)				
	3 × 10^{-2}	7 × 10^{-3}	1 × 10^{-3}	5 × 10^{-4}	1 × 10^{-4}
1	17	13	11	10	9
2	23	18	15	12	10
3	20	13	11	10	10
4	23	23	12	8	11

TABLE 3.2.2: Error Burst Characteristics for Coded 8PSK using Codes 1 to 4 for Viterbi Algorithm Detection

SCHEME	AVERAGE NUMBER OF BIT ERRORS PER BURST AT GIVEN BER (Approximate)			
	0.1	3×10^{-2}	6×10^{-3}	1×10^{-3}
/Dis=E/	-	20	13	11
/Dis=P/	26	19	11	10
/Dis=P2/	27	19	11	10

TABLE 3.2.3: Error Burst Characteristics for Coded 8PSK using Code 3, for Viterbi Algorithm Detection using the Phase Distance Measure

SCHEME	GAIN IN TOLERANCE TO NOISE COMPARED WITH DQPSK AT BER $= 1 \times 10^{-4}$ (dB)
/Ph=D/Pr=O/	1.7
/Ph=D/Pr=D/	1.7
/Ph=Ma/Pr=O/	1.6
/Ph=Ma/Pr=D/	1.8
/Ph=Mb/Pr=O/	1.2
/Ph=Mb/Pr=D/	1.2

TABLE 3.2.4: Performance Comparisons for Schemes using CORPSK(4-7,1+D) Modulation, over the Perfect Channel

SCHEME	APPROXIMATE AVERAGE NUMBER OF BIT ERRORS PER BURST AT GIVEN BER			
	0.1	4×10^{-2}	1×10^{-2}	1×10^{-4}
/Ph=D/Pr=O/	8	5.2	4.2	4.3
/Ph=D/Pr=D/	7	5	4	3
/Ph=Ma/Pr=O/	-	4	2	1.6
/Ph=Ma/Pr=D/	-	3	1.5	1.4
/Ph=Mb/Pr=O/	-	3	2	1.6
/Ph=Mb/Pr=D/	-	3	1.3	1.0

TABLE 3.2.5: Error Burst Characteristics for Schemes Using CORPSK(4-7,1+D) Modulation, Over The Perfect Channel

SCHEME	DEGRADATION IN TOLERANCE TO NOISE IN COMPARISON WITH THE EQUIVALENT UNITARY DISTANCE SCHEME AT BER = 1 in 10^4 (dB)
/Ph=D/Dis=P/	0.9
/Ph=Mb/Dis=P/	0.5

TABLE 3.2.6: Performance of Schemes using the Phase Distance Measure, for CORPSK(4-7,1+D) Modulation

SCHEME	APPROXIMATE AVERAGE NUMBER OF BIT ERRORS PER BURST AT GIVEN BER				
	0.1	4.5×10^{-2}	1×10^{-2}	1×10^{-3}	1×10^{-4}
/Ph=D/Dis=E/	8	5.2	4.2	4.1	4.3
/Ph=D/Dis=P/	7	5.1	4.3	4.3	4.0
/Ph=Mb/Dis=E/	-	3	2	1.3	1.6
/Ph=Mb/Dis=P/	-	3.4	1.7	1.6	1.7

TABLE 3.2.7: Error Burst Characteristics for Schemes using CORPSK(4-7,1+D) Modulation, Over the Perfect Channel, when the Phase Distance Measure is Used.

SCHEME	APPROXIMATE AVERAGE NUMBER OF BIT ERRORS PER BURST AT GIVEN BER			
	0.1	1×10^{-2}	1×10^{-3}	1×10^{-4}
/Q=inf/	8	4.2	4.1	4.3
/Q=4/	7.3	4.6	4.8	5.6
/Q=3/	7.6	4.7	4.2	5.2

TABLE 3.2.8: Error Burst Characteristics for Schemes using CORPSK(4-7,1+D) Modulation, Over the Perfect Channel, when the Detector's Input Samples are Realistically Quantised.

SCHEMES	GAIN IN TOLERANCE TO NOISE OVER DQPSK AT BER = 3×10^{-4} (dB)
/Pf=O/Pr=D/Det=V4/	1.6
/Pf=1RC/Pr=O/Det=V4,16/	1.3
/Pf=1RC/Pr=D/Det=V4,16/	1.8
/Pf=1RC/Pr=O/Det=V16,16/	1.45
/Pf=1RC/Pr=D/Det=V16,16/	1.8
/Pf=N3/Pr=D/Det=V4,16/	0.95
/Pf=N3/Pr=D/Det=V16,16/	1.0

TABLE 3.2.9: Performance of the Premodulation Filtered, Perfect Channel, CORPSK(4-7,1+D), Modulation Schemes

SCHEME	APPROXIMATE AVERAGE NUMBER OF BIT ERRORS PER BURST AT GIVEN BER			
	0.1	1×10^{-2}	1×10^{-3}	4×10^{-4}
/Pf=O/Pr=D/Det=V4/	7.3	4	3.6	3.4
/Pf=1RC/Pr=O/Det=V4,16/	8	4.2	4.4	5.6
/Pf=1RC/Pr=D/Det=V4,16/	7.8	3.8	3.5	3.6
/Pf=1RC/Pr=O/Det=V16,16/	7.8	4.1	4.7	5.6
/Pf=1RC/Pr=D/Det=V16,16/	7.7	3.8	3.5	3.6
/Pf=N3/Pr=D/Det=V4,16/	8	3.9	3.4	3.3
/Pf=N3/Pr=D/Det=V16,16/	7.9	3.8	3.5	3.3

TABLE 3.2.10: Error Burst Characteristics for Premodulation Filtered, Perfect Channel, CORPSK(4-7,1+D) Modulation Schemes

SCHEME	DEGRADATION IN TOLERANCE TO NOISE IN COMPARISON WITH THE /Pf=O/Ch=I2/ SCHEME, AT BER = 3×10^{-4} (dB)
/Pf=N3/Ch=RC/Det=V16,16/	1.0 (approx.)
/Pf=N3/Ch=Mw/Det=V4,16/	0.6
/Pf=N3/Ch=Mw/Det=V4,64/	0.4
/Pf=N3/Ch=Mw/Det=V16,16/	0.6
/Pf=N3/Ch=Mw/Det=V16,64/	0.4
/Pf=N3/Ch=Mn/Det=V16,16/	2.0 (approx.)

TABLE 3.2.11: Performance of the Channel-filtered CORPSK(4-7,1+D) Modulation Schemes

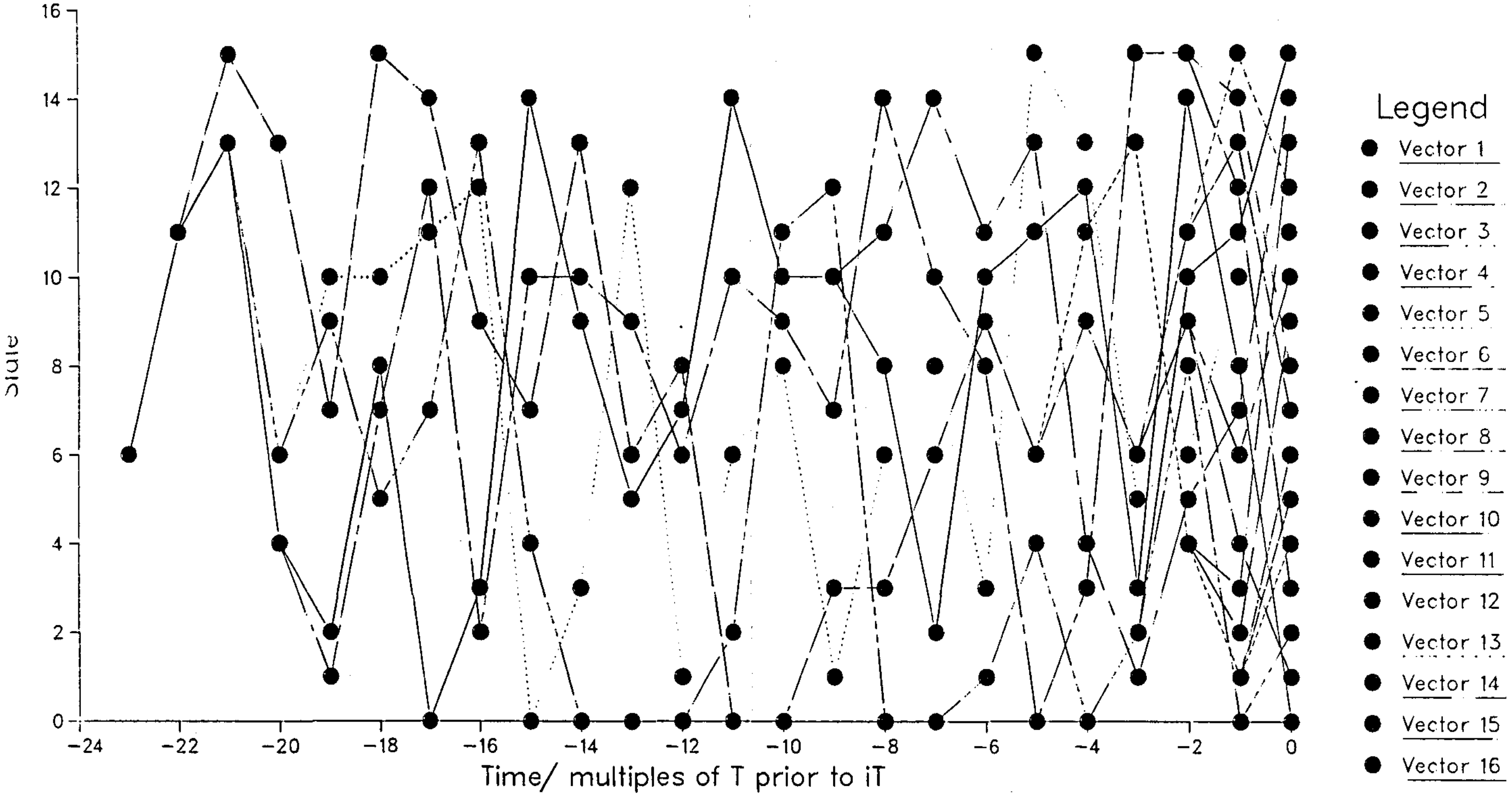
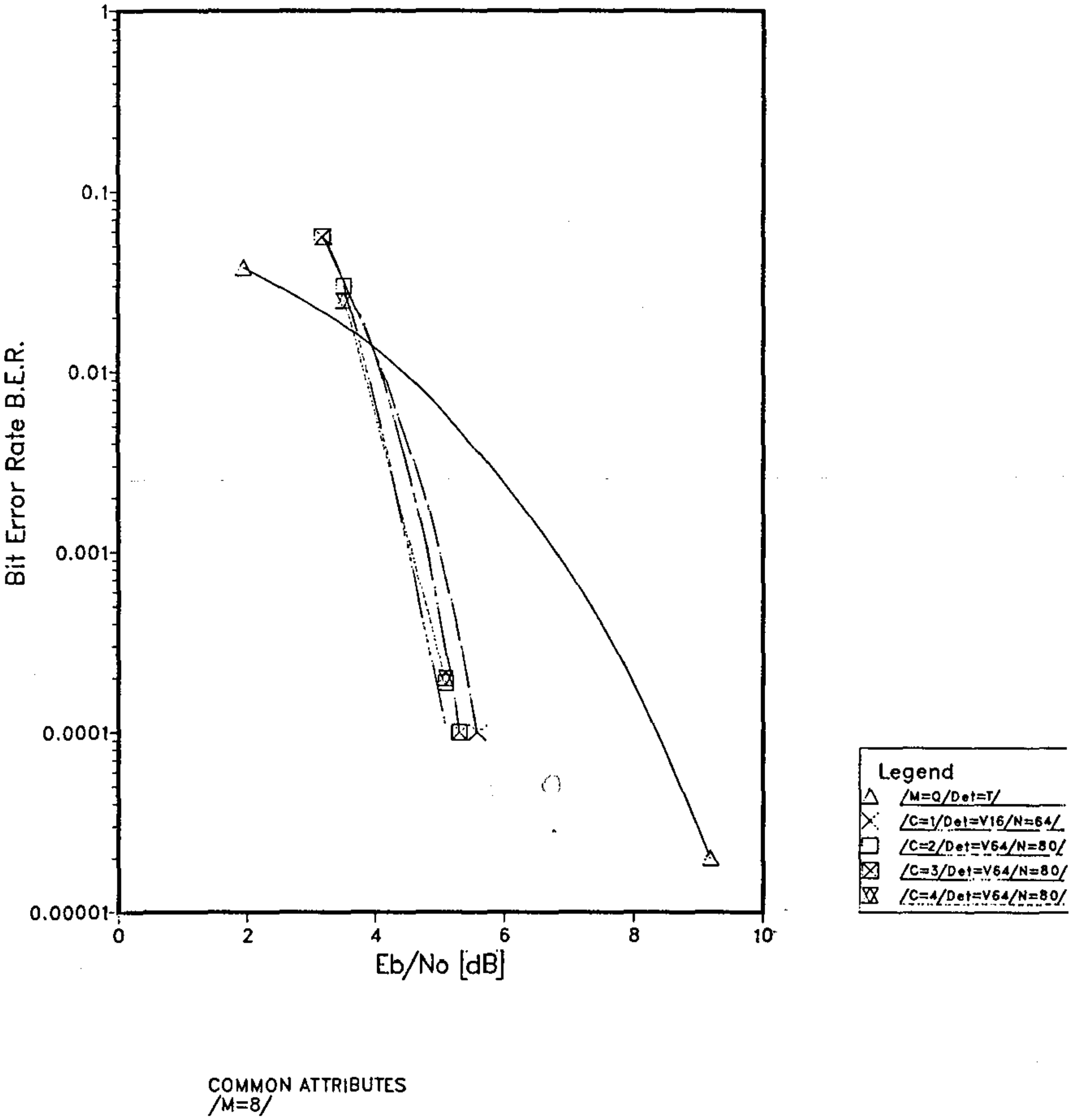
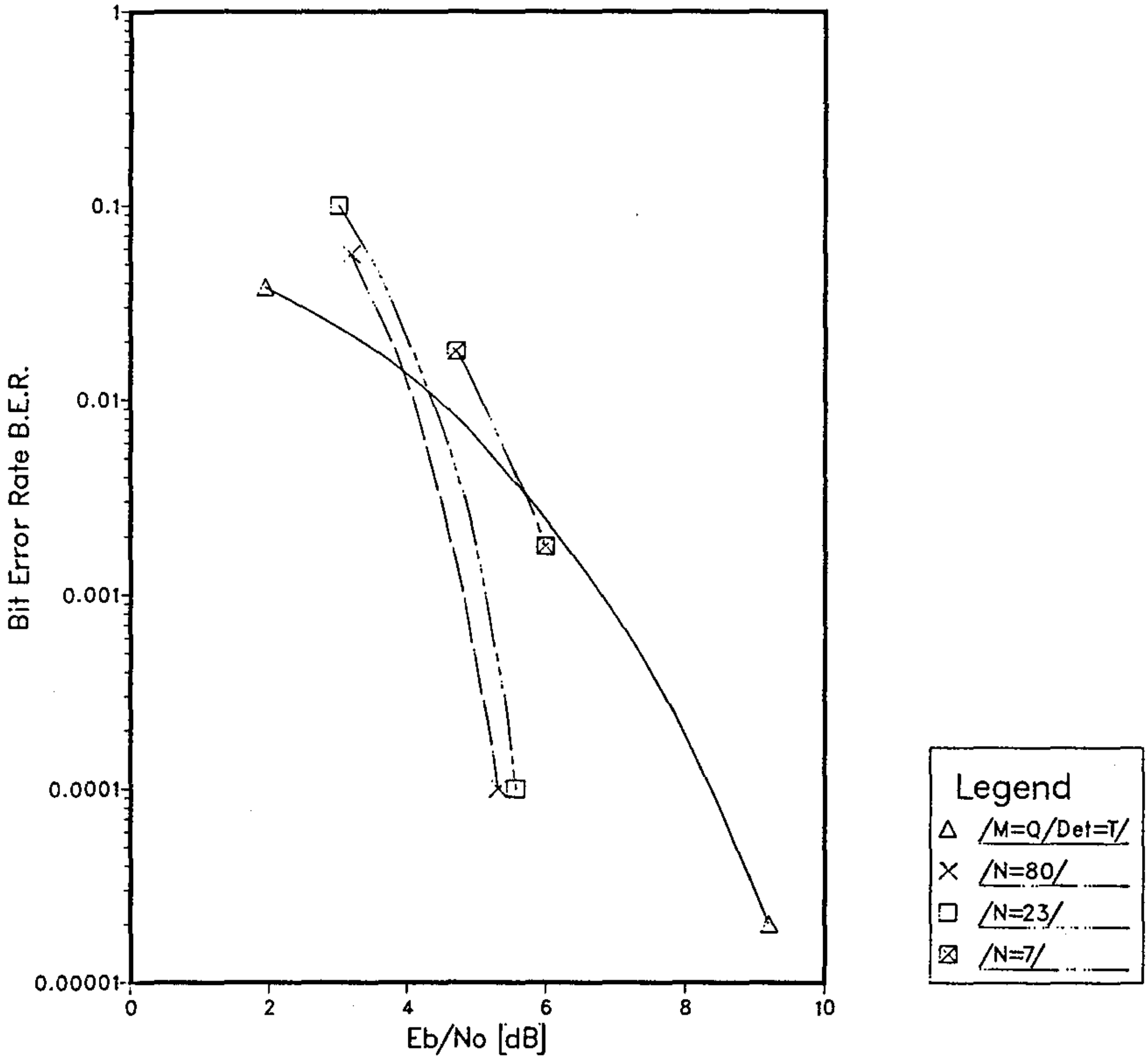


Figure 3.2.1 Code Trellis Diagram for a Scheme using Code 1. Viterbi Detection. BER=0.002

Graph 3.2.1 Coded 8PSK. Viterbi Detection

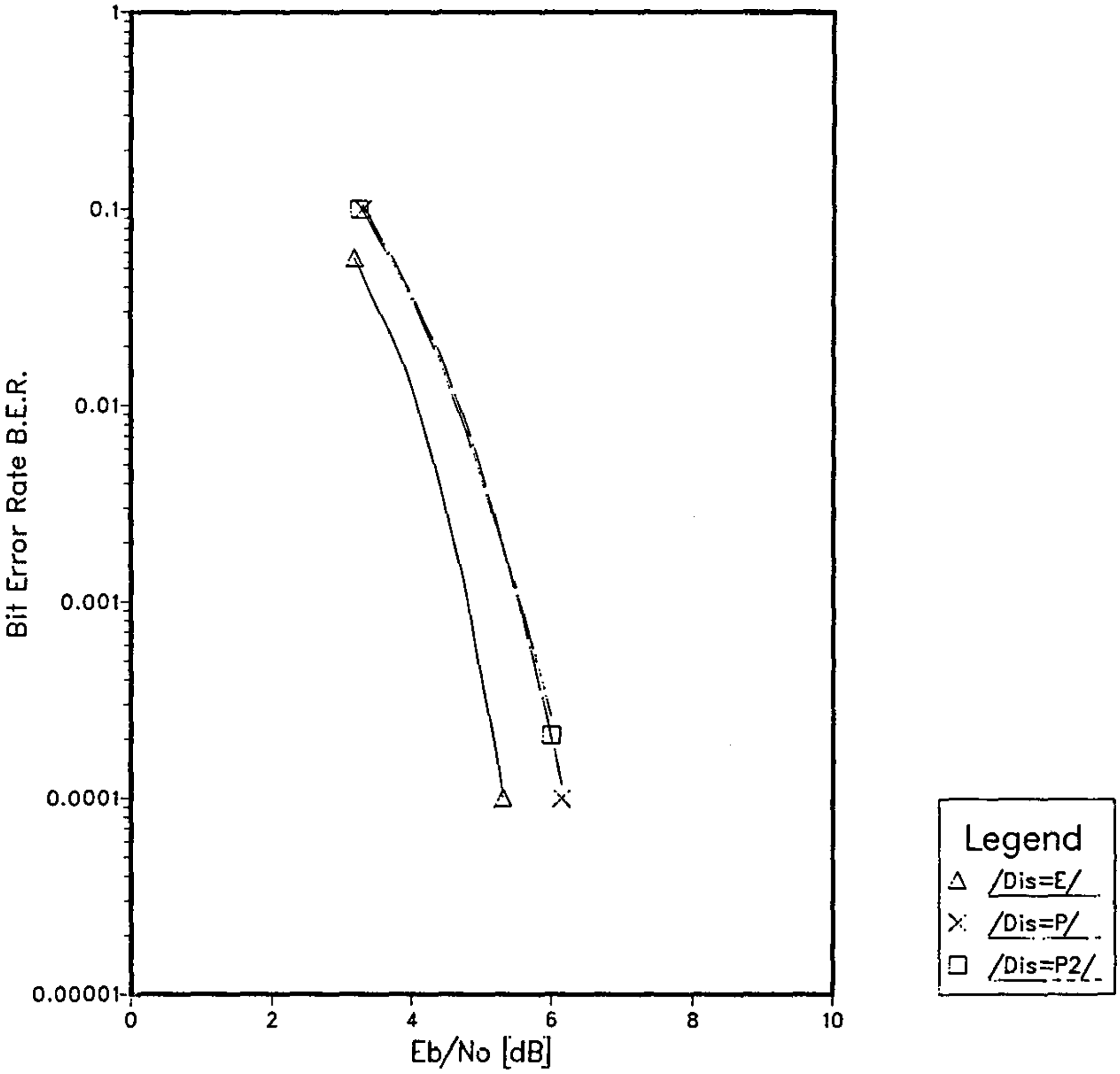


Graph 3.2.2 Coded 8PSK. Reduced Detection Delay



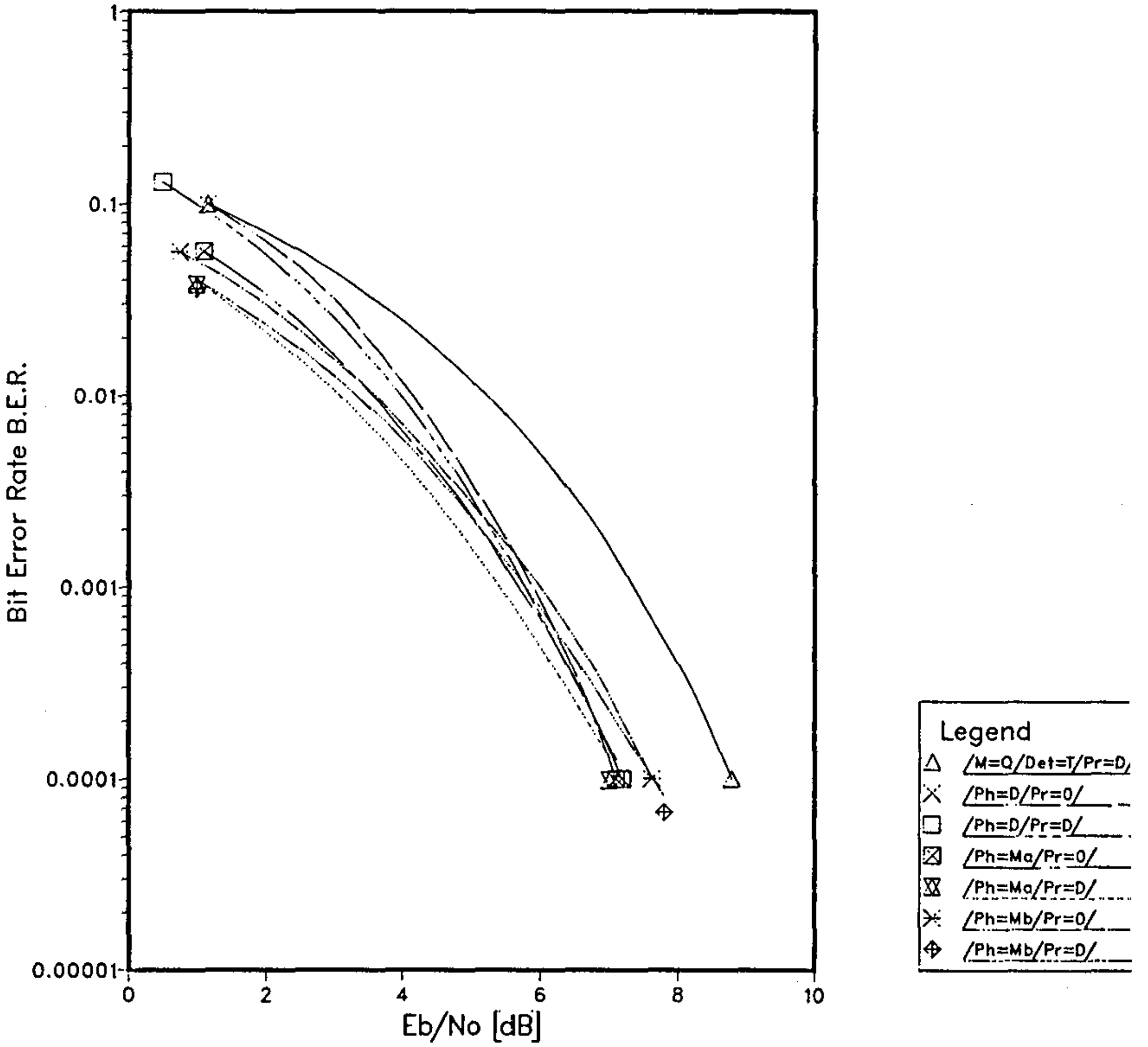
COMMON ATTRIBUTES
/M=8/C=3/Det=V64/

Graph 3.2.3 Coded 8PSK. Phase Distance Measure



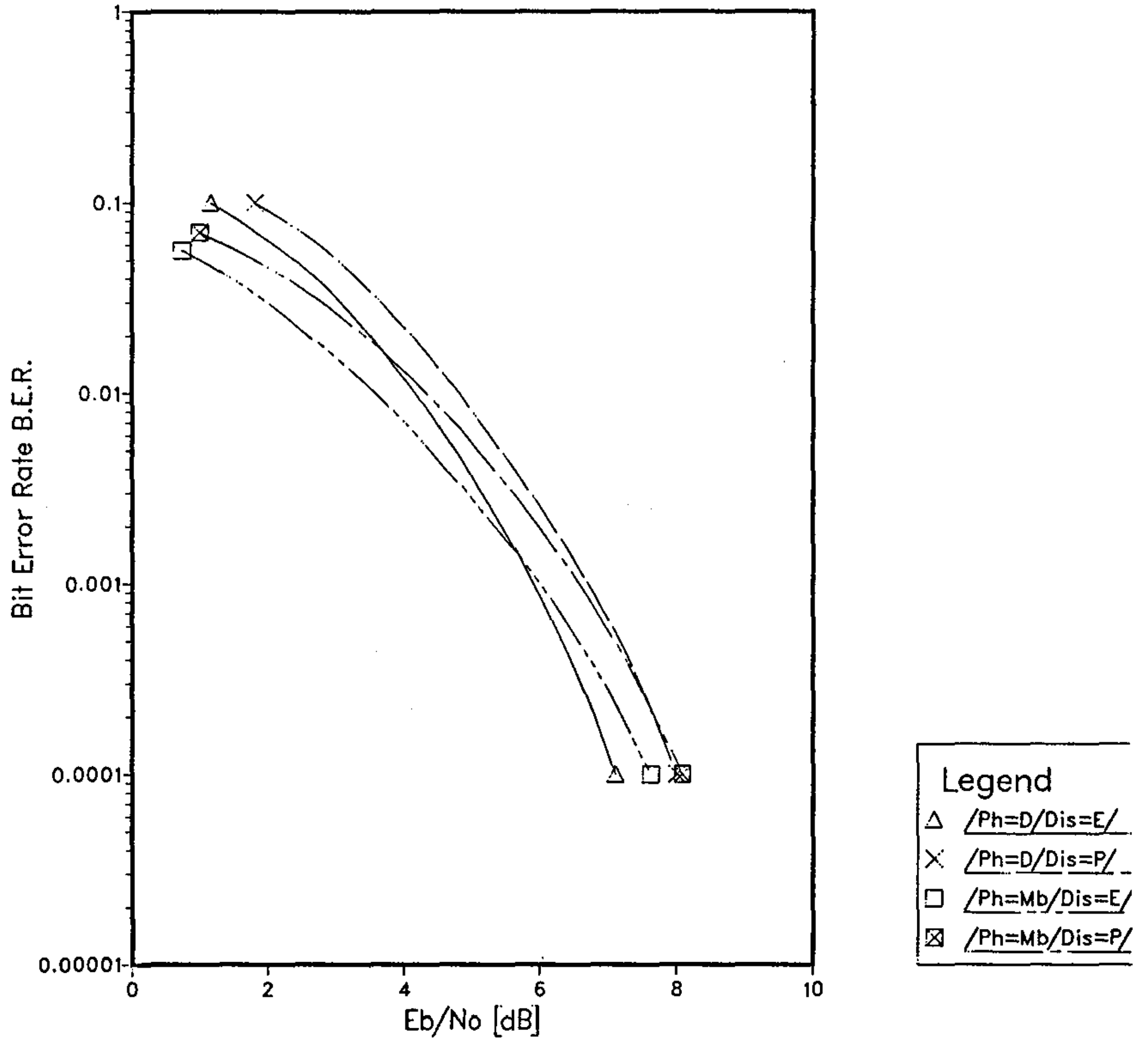
COMMON ATTRIBUTES
/M=8/C=3/Det=V64/N=80/

Graph 3.2.4 CORPSK[4–7,1+D] Perfect Channel Schemes



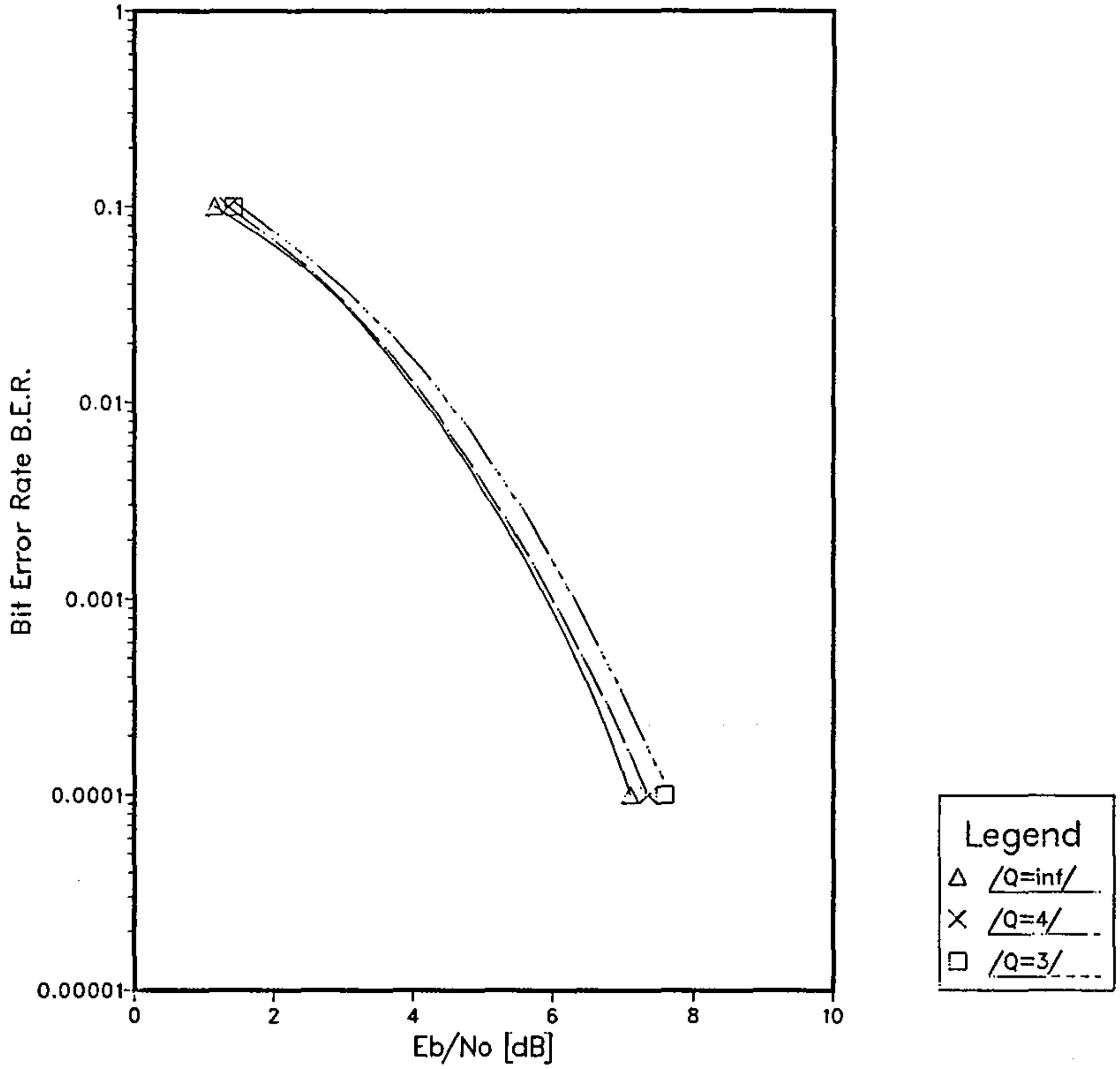
COMMON ATTRIBUTES
/M=C/Ch=I2/Det=V4/N=32/

Graph 3.2.5 CORPSK[4–7,1+D] Perfect Channel Schemes
Phase Distance Measure



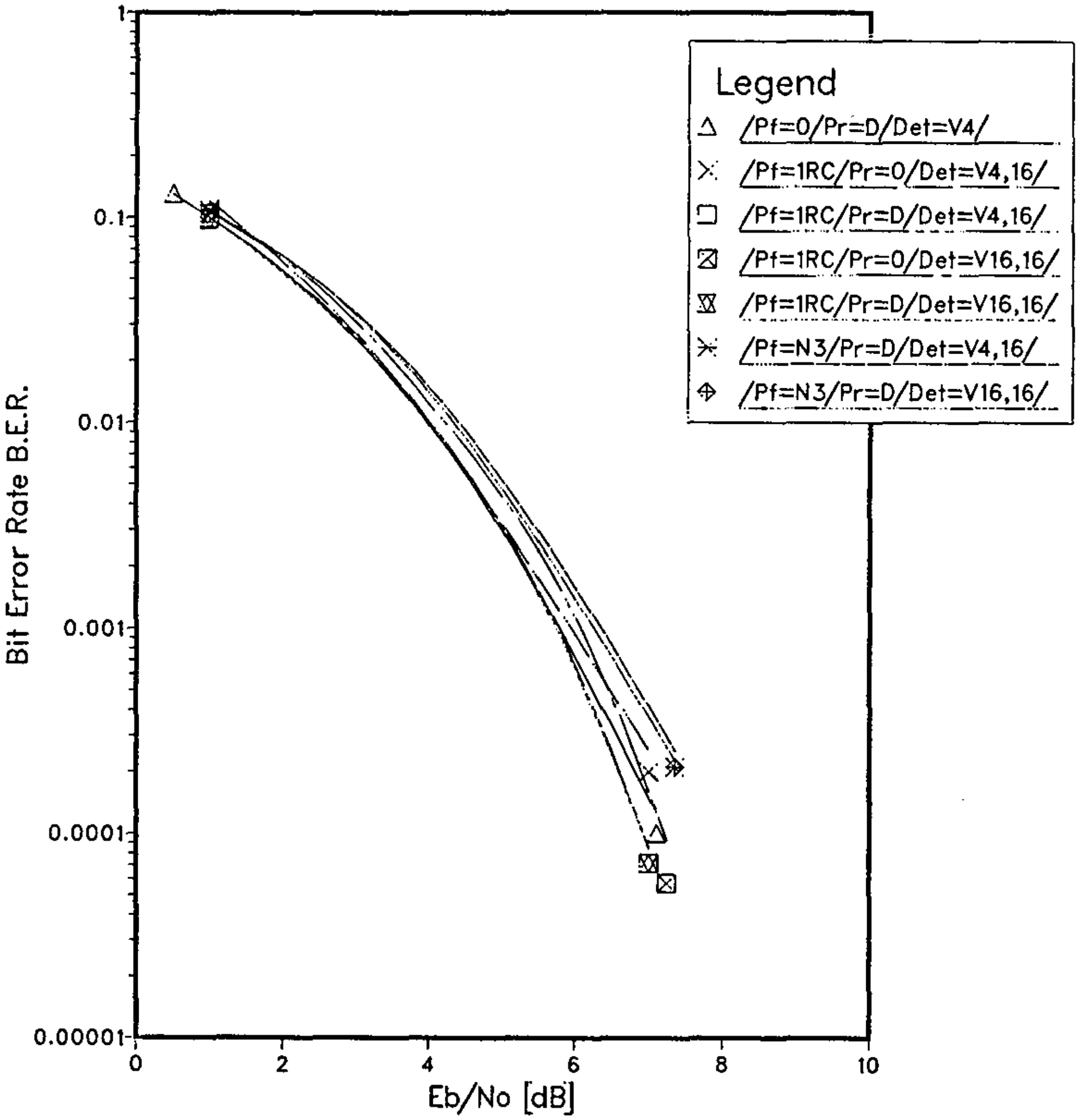
COMMON ATTRIBUTES
/M=C/Ch=I2/Pr=0/Det=V4/N=32/

Graph 3.2.6 Quantised CORPSK[4–7,1+D] Perfect Channel Scheme



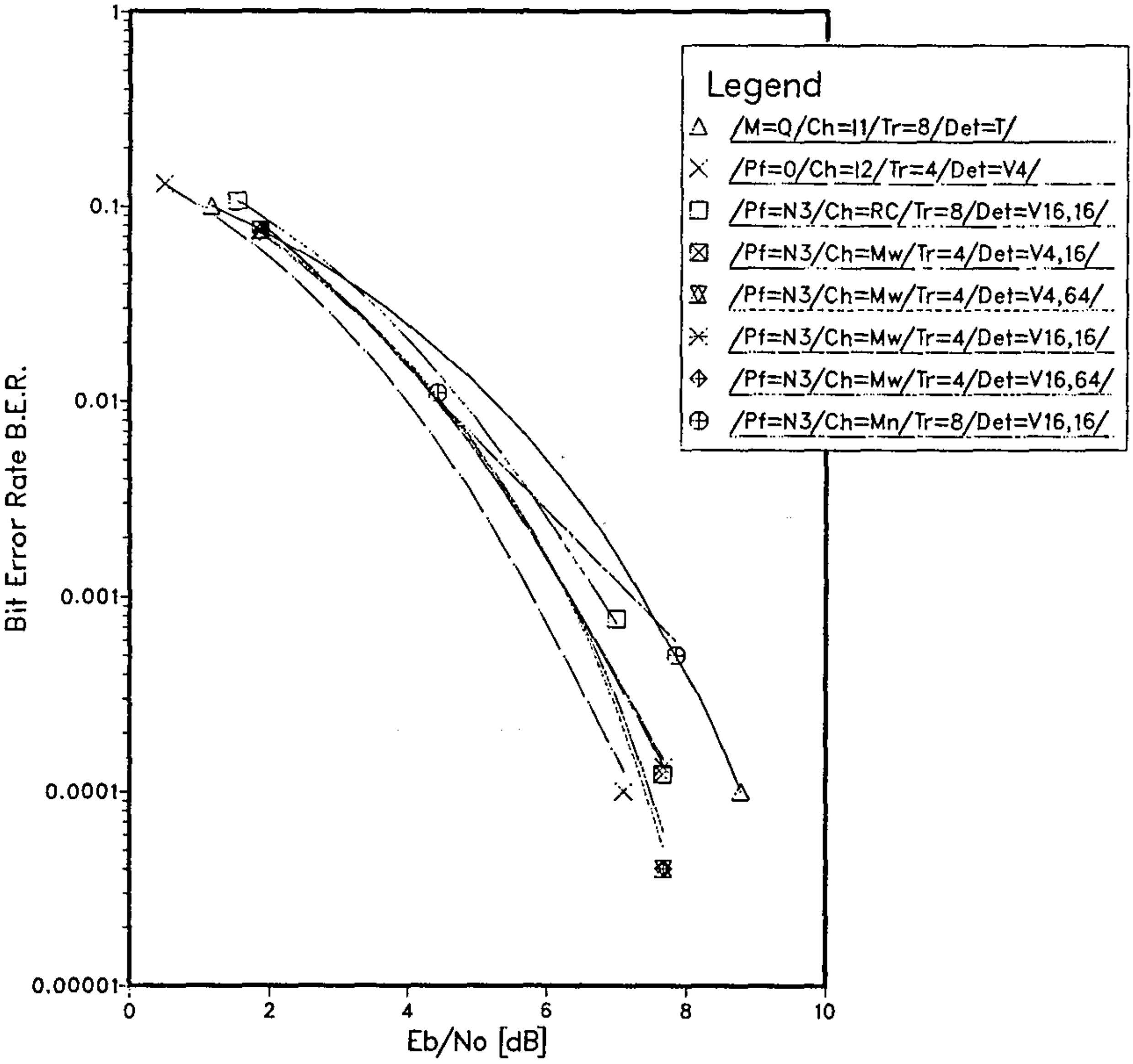
COMMON ATTRIBUTES
/M=C/Ch=I2/Ph=D/Pr=0/Det=V4/N=32/

Graph 3.2.7 Premodulation Filtered CORPSK[4-7,1+D] Perfect Channel Schemes



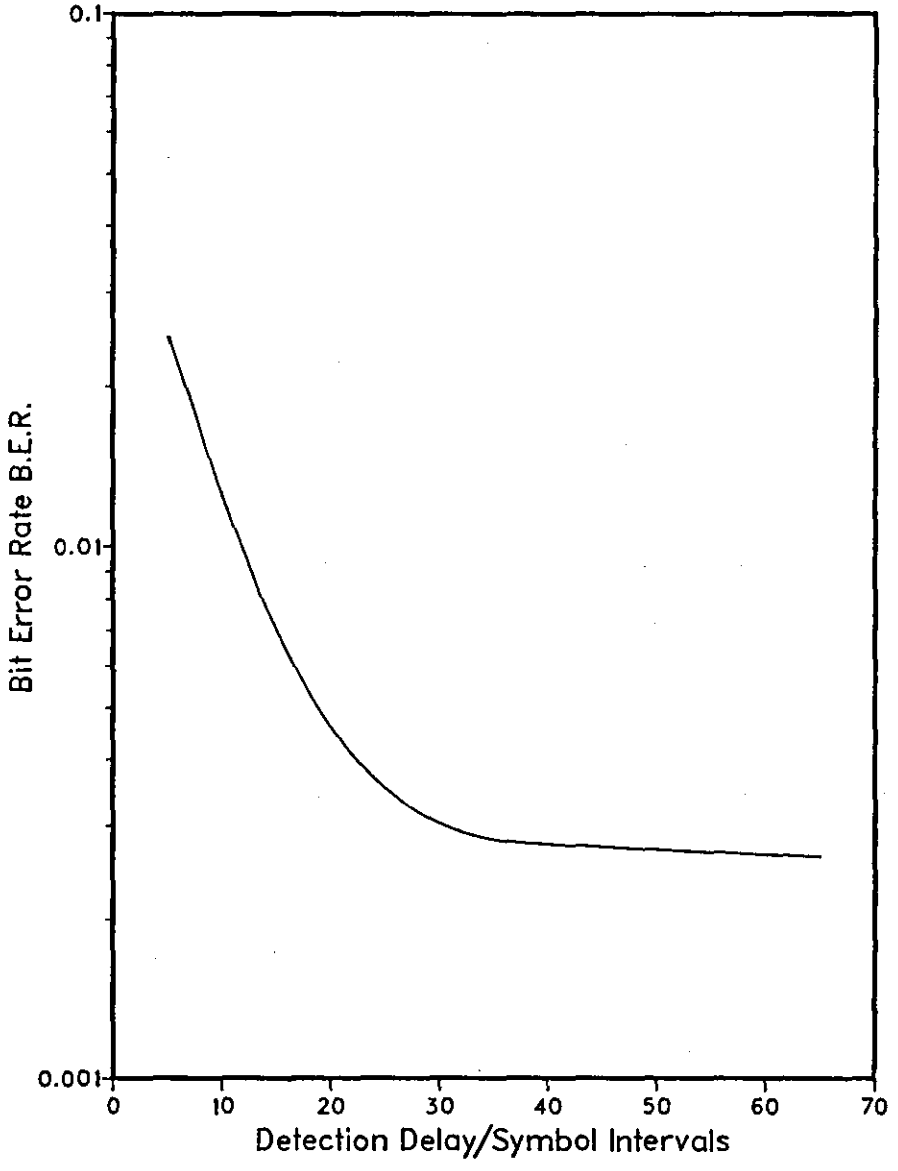
COMMON ATTRIBUTES
/M=C/Ch=I2/Tr=4/Ph=D/N=32/

Graph 3.2.8 Filtered CORPSK[4−7,1+D]



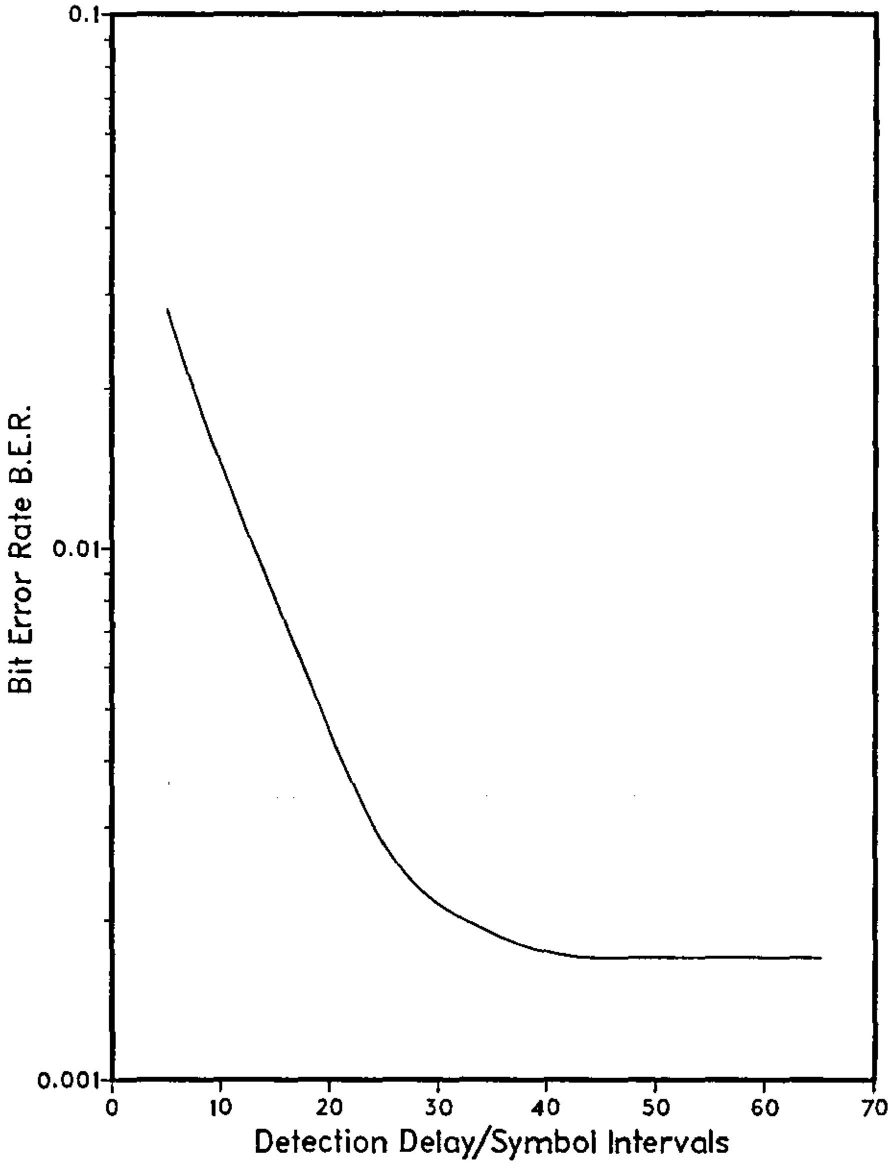
COMMON ATTRIBUTES
/M=C/Ph=D/Pr=D/N=32/

Graph 3.2.9 Variation of B.E.R. with Detection Delay at Eb/No=4.76dB



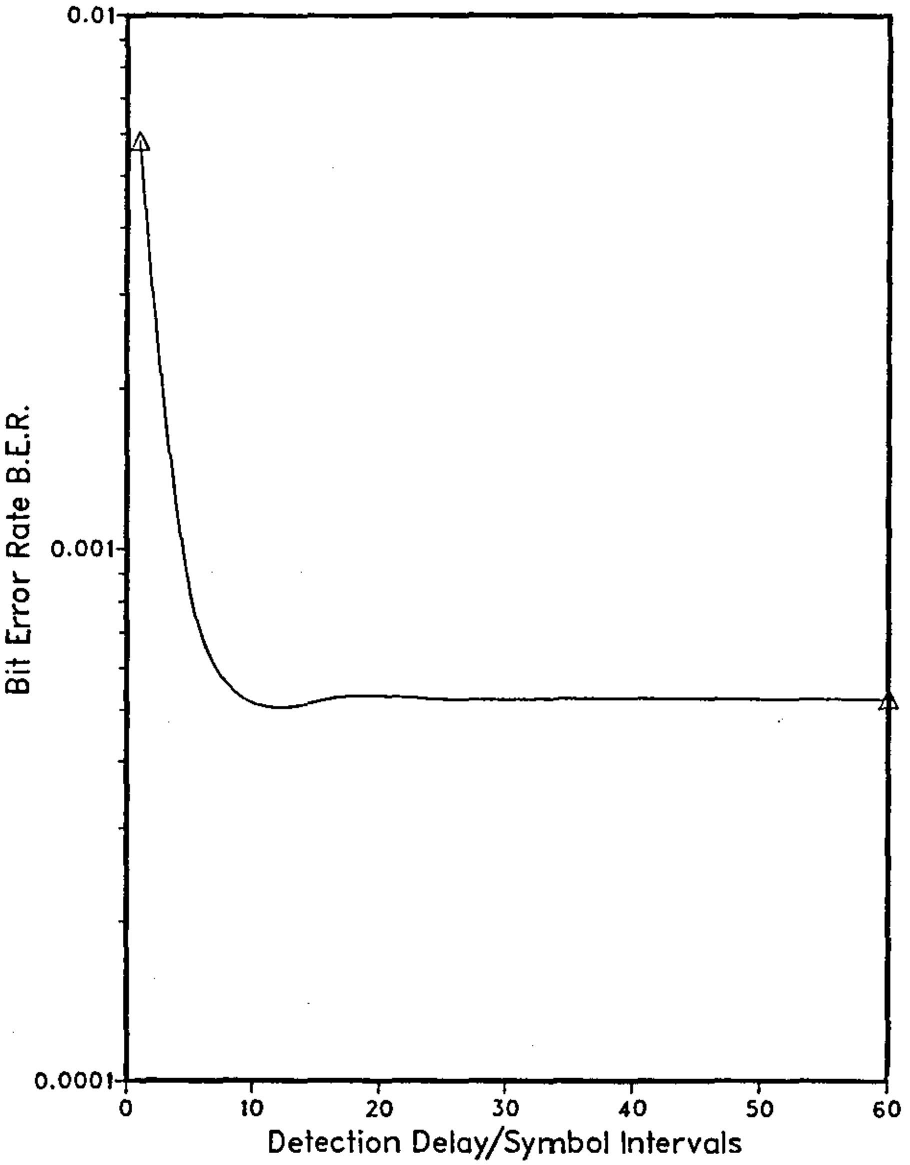
SYSTEM ATTRIBUTES
/M=8/C=1/Det=V16/

Graph 3.2.10 Variation of B.E.R. with Detection Delay at Eb/No=4.76dB



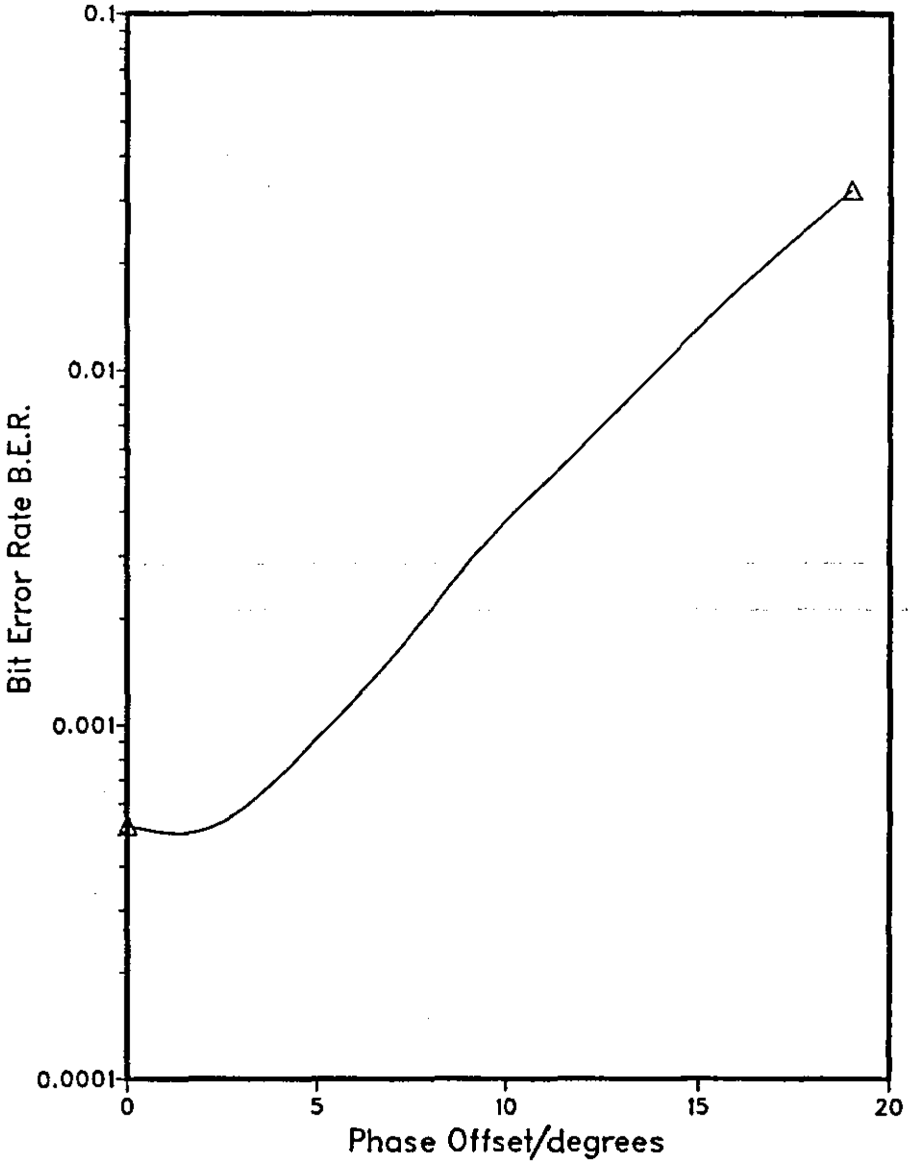
SYSTEM ATTRIBUTES
/M=8/C=3/Det=V64/

Graph 3.2.11 Variation of B.E.R. with Detection Delay at Eb/No=6.3dB



SYSTEM ATTRIBUTES
/M=C/Ch=12/Tr=4/Ph=D/Det=V4/

Graph 3.2.12 Variation of B.E.R. with Received Constant Carrier Phase Offset at Eb/No=6.3dB



SYSTEM ATTRIBUTES
/M=C/Ch=12/Tr=4/Ph=D/Det=V4/

CHAPTER 4

NEAR-MAXIMUM LIKELIHOOD DETECTION SCHEMES FOR CODED 8PSK

This chapter describes a number of detectors which are derived from the Viterbi detector of Chapter 3. In all cases the detectors store a number of vectors of possible data sequences and their associated costs. These detectors differ from the Viterbi detector in the algorithms which use these stored vectors and costs to produce detected data symbols $\{q_i'\}$. The aim is to develop detectors which are considerably less complex than the Viterbi detector, without a significant degradation in tolerance to noise.

These detection techniques were not applied to CORPSK(4-7,1+D) modulation, because Viterbi detection is relatively simple in this case, so that these techniques cannot provide a significant reduction in complexity.

Table A8.1 defines the notation which is used to describe the many variants of the schemes which are tested by computer simulation.

4.1 SYSTEM 1 WITH ANTI-MERGING

This detector is one of a number of Viterbi-type schemes initially investigated by A.P. Clark et al.[64,65] The family of detectors has since been investigated in a number of different applications, and with a number of modifications.[27,28,66-71] The description of the detector begins with a description of the received signals. The detector is then described in terms of its stored vectors and costs. The algorithm, repeated during every symbol interval, which uses these stored values to produce the detected data symbols, is described. The unitary distance measure (Appendix A7) is used for the stored costs. Other distance measures which are used are defined in Appendix A7.

Equation 4.1.1 (from Equation 2.1.7) gives the received sample at the detector input at time t=iT. Since the equipment filters introduce no significant intersymbol interference (see Section 2.5), the sampled impulse response of the channel is taken to be $Y=[y_0,y_1,\ldots,y_g]$, where only y_0 is non-zero, and is equal to one.

$$r_i = p_i + w_i \qquad (4.1.1)$$

p_i is the complex number given by the mapping of the code symbol c_i, described in Figure 2.5.4. w_i is a sample value of the Gaussian noise waveform w(t) at the demodulator output. See Section 2.5 for more details.

The detector stores k_1 vectors $\{Q'_i\}$ of possible four-level data symbol values. At time t=(i-1)T these vectors have the form

$$Q'_{i-1} = [q'_{i-N+1},q'_{i-N+2},\ldots,q'_{i-1}] \qquad (4.1.2)$$

where q'_ℓ is a possible value of the transmitted data symbol q_ℓ. On the receipt of r_i the detector forms possible values of r_i in the absence of noise as follows, to be compared with r_i. Each vector Q'_{i-1} is expanded four ways to form four expanded vectors at time t=iT, by appending one of the four possible data symbol values; q'_i = 0,1,2, or 3. The $\{q'_i\}$ in an expanded vector are then coded using the convolutional code described in Section 2.5 to give the vector of binary code symbols $[c'_i(1),c'_i(2),c'_i(3)]$

$$c'_i(j) = \bigoplus_{\ell=1}^{2} \bigoplus_{h=0}^{k-1} q'_{i-h}(\ell)\, g_h(\ell,j) \qquad (4.1.3)$$

for j=1,2,3,

The $\{g_h(\ell,j)\}$ are binary-valued, and k is the code constraint length. $\bigoplus$ denotes MODULO-2 summation. $[q'_i(1),q'_i(2)]$ is a two-component vector that is uniquely related to q'_i according to Table 2.1.1.

$[c_i'(1), c_i'(2), c_i'(3)]$ is now mapped onto the 8-level symbol c_i'.

$$c_i' = 2^2 c_i'(1) + 2^1 c_i'(2) + 2^0 c_i'(3) \qquad (4.1.4)$$

Since $c_i'(1), c_i'(2)$ and $c_i'(3)$ each have the two possible values 0 or 1, c_i' takes on one of the eight possible values 0,1,2,...,7. A possible value of r_i in the absence of noise is given by mapping c_i' onto a complex number p_i', where the mapping is defined in Figure 2.5.4. For each value of p_i', the quantity w_i' is determined, which is a possible value of the noise component, w_i.

$$r_i = p_i' + w_i' \qquad (4.1.5)$$

Each vector Q_{i-1}' at time t=(i-1)T has a cost $|W_{i-1}'|^2$, which is a function of the $\{w_i'\}$, i=1,2,...,(i-1). The cost is a measure of how likely it is that a vector contains data symbol values which are the same as those of the transmitted data. A low cost implies high likelihood. At time t=iT the cost $|W_i'|^2$ of an expanded vector is given by calculating the appropriate value $|w_i'|^2$, (the incremental cost), and adding this to the value of $|W_{i-1}'|^2$ of the vector Q_{i-1}' from which the expanded vector is derived. Clearly

$$|W_i'|^2 = |W_{i-1}'|^2 + |w_i'|^2 \qquad (4.1.6)$$

Also,

$$\begin{aligned} |w_i'|^2 &= |r_i - p_i'|^2 \\ &= [\mathrm{Re}(r_i - p_i')]^2 + [\mathrm{Im}(r_i - p_i')]^2 \qquad (4.1.7) \end{aligned}$$

The distance measure used in the calculation of $|W_i'|^2$ is the unitary distance measure (see Appendix A7).

Up to this point the Viterbi and System 1 detectors are identical except that the number of stored vectors may be different in the two cases. The System 1 procedure for selecting k_1 vectors $\{Q_i'\}$ from the $4k_1$ expanded vectors, differs from the procedure for the Viterbi algorithm. Initially the detector finds the expanded vector with the lowest cost $|W_i'|^2$. This vector is stored along with its cost, and the value of q_{i-N+1}' contained within this vector is taken to be the detected value of q_{i-N+1}. At this point all other expanded vectors, whose values of q_{i-N+1}' are not the same as that in the vector with the lowest cost, are discarded from all future detection processes. This prevents the merging (becoming the same) of the stored vectors, since it ensures that if they are all different at the start of transmission, no two of them can subsequently become the same. This procedure is called anti-merging. Finally, the detector selects from the remaining expanded vectors, the (k_1-1) with the lowest costs $|W_i'|^2$. Each selected vector Q_i', and associated cost $|W_i'|^2$, are stored. No expanded vector may be selected more than once, so that after being selected the chosen expanded vector is excluded from further selection processes. As in the case of the Viterbi detector, the lowest value of $|W_i'|^2$ is subtracted from all costs $|W_i'|^2$ to prevent overflow in their stored values.

A simple procedure at the start of transmission is to begin with k_1 stored vectors that are all the same and if possible, (but not necessarily), all correct. A cost of zero is associated with one of these vectors, all other vectors having very high costs. In this way, after only a few symbol intervals, all the vectors will be derived from the original vector with zero cost, and will all be different.

Graphs 4.1.1 to 4.1.13 give the results of the computer simulation tests for System 1 detection of coded 8PSK, under various conditions. For Graphs 4.1.1 to 4.1.8, which are graphs of bit error rate (BER) as the signal to noise ratio, E_b/N_0, varies, the accuracy of the results is of the order of ±0.25dB within the range of BER, 1 in 10^3 to 1 in 10^4. E_b is the average energy transmitted per data bit. $N_0/2$ is the two-sided power spectral density of the additive white Gaussian noise. (See Appendix A5 for more details of the simulation techniques. Appendix A8 gives the notation used to describe the variants of System 1 which were tested by computer simulation.)

Graph 4.1.1 gives the results for System 1 detection of schemes using Code 1, where k_1 is either 4 or 8. Graph 4.1.2 is the equivalent for Code 3, where k_1 is 4,8, or 16. In both cases the reference curves are for threshold-detected QPSK, and for the appropriate Viterbi Algorithm-detected scheme. Graph 4.1.1 indicates that schemes using Code 1 and System 1 detection suffer an appreciable degradation in tolerance to noise, compared with the corresponding scheme using Viterbi detection. At a BER of 1 in 10^4 the scheme where k_1=8, (/Det=1N8/), loses approximately 0.9dB in tolerance to noise, while the scheme where k_1=4, (/Det=1N4/) loses approximately 2.2dB in tolerance to noise, compared with Viterbi detection. The scheme with k_1=4 only gains 0.65dB in tolerance to noise with respect to QPSK at this BER. This degradation is all the more severe when the relative complexities of the schemes are considered. The algorithm for System 1 is considerably more complex than the Viterbi algorithm, for the same number of stored vectors. This is because, whereas the Viterbi detector conducts 16 separate cost-comparison operations, each such operation involving the costs of 4 expanded vectors, the detector for System 1 must conduct k_1 separate cost comparisons through all $4k_1$ expanded vectors. Clearly the latter process is considerably more complex than the former, for the same number of stored vectors. Therefore, depending on the detailed method of implementation, it is quite probable that the schemes using System 1 detection are of the same order of complexity as the Viterbi detector. A comparison of the schemes' error burst characteristics is given in Table 4.1.1. The definition of an error burst is given in Appendix A5. It is evident that System 1 detection increases the average number of bit errors per burst. For $k_1=4$, (/Det=1N4/), the increase is severe, since it tends to increase as the BER decreases. For $k_1=8$, (/Det=1N8/), number of bit errors per burst is still more than twice that for the Viterbi detector, at a BER of 5 in 10^4. This is clearly significant. An analysis has been carried out of the typical state of the detector's code trellis diagram for a scheme using Code 1 and System 1 detection, as in Section 3.2 (see Figure 3.2.1). This diagram is essentially a graph of the state of a vector (vertical axis), as it varies with time in symbol intervals (horizontal axis). The state of a vector, as described in Section 2.5, is given by the combination of the values of the vector elements $q'_{i-k+1},q'_{i-k+2},\ldots,q'_{i-1}$ at time t=iT. An integer value is given to each possible state, as described in Section 2.5. The code trellis diagram gives the states for each of the stored vectors over a period of time up to the current time t=iT. Each line in the diagram is for one of the k_1 stored vectors. More details are given in Appendix A4. Figures 4.1.1 to 4.1.3 are typical code trellis diagrams during computer simulation tests for a scheme using

Code 1, for k_1=16,8, and 4, respectively. The code trellis diagrams should be contrasted with Figure 3.2.1. Figure 4.1.1 is of particular interest since k_1 is equal to the number of stored vectors for the Viterbi detector (16). Note that the BER values for Figures 3.2.1 and 4.1.1 are very similar. Whereas in Figure 3.2.1, every stored vector has the same state for time $t \leq (i-22)T$, every vector in Figure 4.1.1 has the same state for time $t \leq (i-10)T$. Clearly the variety in the stored vectors is much less marked for System 1 detection than for Viterbi detection, even when both store the same number of vectors. The shorter the period of time, $(i-j)T<t<iT$, for which the stored vectors' element values differ, the smaller will be the cost differences between the vectors in the absence of noise. In such a case it is more likely that the algorithm could discard the correct vector in the presence of noise, than in the case where the stored vectors have different element values over a longer period of time. Therefore final decisions as to the transmitted data are taken too early with System 1 detection. Another interesting point which emerges from Figure 4.1.1 is that certain vectors have the same element values over periods of time. For example from time $t=(i-3)T$ onwards, vectors 13 and 15 are the same. These vectors at time $t=iT$ have the same state, (combination of the values of q'_{i-2} and q'_{i-1} since k=3), but have different costs. The reason for their different costs is that they have different element values for $t<(i-3)T$. Clearly the System 1 algorithm may discard one or both of these vectors before long, but it is possible that they may remain for a long period. (Clearly the anti-merging rule will prevent merging.) From the theory of Maximum Likelihood detection, (Appendices

A3 and A4), the existence of more than one vector with a given state at time t=iT is superfluous. This is because once the state of two vectors becomes the same at time t=iT, as far as the calculation of the costs is concerned, the vectors are identical. Therefore the difference in the costs of the two vectors remains the same for t>iT. This effectively reduces the number of stored vectors by one.

Clearly this problem is increased as k_1 is reduced since if the number of stored vectors is effectively reduced in this way, there are correspondingly fewer remaining vectors. For example in Figure 4.1.2 for k_1=8, vectors 4 and 6 have had the same state since time t=(i-16)T. The points made about Figure 4.1.1 apply equally to Figures 4.1.2 and 4.1.3, but even more so. For k_1=4, (all the tests providing these diagrams having been conducted at the same value of E_b/N_0=4.76dB, and with the same noise and random data sequences), only two vectors with different element values exist for time t≤(i-3)T in Figure 4.1.3.

Graph 4.1.2 indicates, overall, that the same is true for the schemes using the longer constraint length (k=4) code, Code 3. The comparison is with threshold-detected QPSK and Viterbi detection for a scheme using Code 3 (64 vectors). Schemes using System 1 with k_1=16,8 and 4 have been tested. Table 4.1.2 lists the degradations in tolerance to noise, compared with Viterbi detection, at a BER of 1 in 10^4. Clearly the degradation is severe. It is interesting to note that the degradation caused by reducing the delay in detection for k_1=16 from 64 to 32 symbol intervals is negligible. This should be considered in the light of the code trellis diagram for a scheme using Code 1, where k_1=16 (Figure 4.1.1). Here all the stored vectors have the same element values for t≤(i-10)T. Therefore, a reduction of this sort in the detection delay should cause little degradation. In contrast, for $k_1=4$, reducing the delay in detection from 32 to 16 symbol intervals is significant. Therefore it can be concluded that usually more than one vector with different element values exists for $t \leq (i-16)T$, whereas for $t \leq (i-32)T$ all the vectors usually have the same element values. Note also that the curve for $k_1=8$ compares rather favourably with that for $k_1=16$. In the light of the accuracy of ±0.25dB stated above, this may not be surprising. Table 4.1.3 gives the error burst characteristics for System 1 detection of coded 8PSK using Code 3. Again, System 1 detection significantly increases the average number of bit errors per burst, especially for the lower values of k_1. It is interesting to note though, that bursts of nearly 200 errors which occurred for Code 1, do not occur. This is probably a function of the codes themselves, and their relative suitability for System 1 detection. This is corroborated by Graph 4.1.3 which contrasts System 1 detection for schemes using Codes 1 to 4. Table 4.1.4 lists the degradations, where these can be accurately ascertained, in tolerance to noise compared with Viterbi detection for the scheme using Code 1, at a BER of 1 in 10^4. Clearly, the performance for schemes using the k=4 constraint length codes under System 1 detection, varies widely. Table 4.1.5 gives the error burst characteristics. Clearly, the schemes using Codes 2 and 4 are not suited to System 1 detection, in that large error bursts occur. In Chapter 3, it was suggested that the differing performances of schemes using the three codes with constraint length k=4, is probably due to differing distance profiles. The distance profile is a measure of how quickly the distance between two code sequences increases.[73] The two code sequences are those, of all possible code sequences, where this distance is a minimum given that the two sequences differ in the value of their first symbol. If this distance increases only slowly with time, the costs of the two sequences, where one is the correct sequence, may be very similar over quite a long period of time, even in the absence of noise. This affects the probability of discarding the correct sequence over this period in the presence of noise. Clearly the distance profile will also affect the length of time required to resume correct detection, once the lowest-cost vector contains wrong element values. This affects the number of errors per burst. Another factor in this is the number of other code sequences which have very similar costs to the two minimum-distance code sequences defined above, (the near-minimum distance code sequences). The greater the number of such sequences, the more likely it is under noisy conditions, that the lowest-cost sequence will be one of these, rather than the correct one. For $k_1=4$, the significant increase in the number of errors per burst as the BER reduces, signifies a situation where correct detection only resumes due to further noise-induced errors. This gives an increase in the number of errors per burst as the BER decreases since, as the noise level decreases, such noise-induced errors become fewer.

Graph 4.1.4 contrasts System 1 detection for schemes using Codes 1 and 3. Clearly, for the same value of k_1, Code 3 is preferable. This highlights, as for Viterbi detection, the larger error-correcting capability of the longer constraint length code.

Graphs 4.1.5 to 4.1.8 illustrate the effects of using suboptimal distance measures for the costs and the effect of realistic quantisation of the received samples $\{r_i\}$. Appendix A7 describes these distance measures, which are used to reduce the complexity of the detectors.

Graph 4.1.5 contrasts both the phase distance and the magnitude-sum distance measures with the unitary distance measure, for a scheme using Code 3 under System 1 detection. The phase distance is simply the difference in the phase angles of the received sample r_i and a possible received sample in the absence of noise. The magnitude-sum distance measure is given by calculating the magnitudes of the differences between the real and imaginary parts of r_i and a possible received sample in the absence of noise. These are summed to give the distance measure. These measures involve no squaring operations and are therefore simpler to implement than the unitary distance measure. Results for k_1=16,8, and 4 are presented. Table 4.1.6 lists the degradations in tolerance to noise for schemes using the suboptimal distance measures compared with the equivalent schemes using the unitary distance measure at a BER of 1 in 10^4. Clearly the degradations are quite severe, although in the case of the phase distance measure, not as severe as for Viterbi detection, (Graph 3.2.3). It can be seen that the magnitude-sum measure leads to a consistently larger degradation than does the phase distance measure, particularly for k_1=4. Therefore the phase distance measure may have a particular advantage for constant envelope-type schemes, or more specifically, schemes where all the values of p_i lie on a circle in the complex number plane, (see Appendix A7). Table 4.1.7 gives the error burst characteristics for the schemes. Clearly the error burst characteristics are very similar in all cases.

A true comparison between the schemes of Graph 4.1.5 is not possible, simply because a measure of their relative complexities is not available, (since this is implementation-dependent). A practical advantage can be gained if the received sample is simply the phase angle of r_i, (that is, phase demodulation is used). This is because the bits available in the receiver to represent the received sample can all be used to represent the phase angle of r_i, $\phi(r_i)$, rather than splitting these bits into two equal parts to represent the real and imaginary bits of r_i separately. $\phi(r_i)$ is used to address a look-up table of unitary distances. The incremental costs at the output of the look-up table are those for a received sample with phase angle $\phi(r_i)$ which lies on the circle in the complex number plane upon which the $\{p_i\}$ lie (see Figure 2.5.4). In other words, knowledge of the magnitude of r_i is not used. To test this supposition, this system is compared with a system where r_i is received, and the true unitary distance measure is used.

The total number of available bits is set at 8 in both cases. For the scheme using the true unitary distance, r_i is quantised into 4 bits per real component and 4 bits per imaginary component. For the case where the received sample is the phase angle of r_i, $\phi(r_i)$ is quantised into 8 bits. In both cases the distances are calculated by means of a look-up table. In the unitary distance case, the real and imaginary parts of r_i are each separately quantised, where the outermost quantisation levels are set at $\pm 1.2|p_i|$ where $|p_i|$ is the magnitude of p_i (which is 2.0, see Section 2.1). The quantisation levels are uniformly spaced. In the case where $\phi(r_i)$ is received, the complex number plane is divided into 2^8 equal sectors about the origin. The quantised value of $\phi(r_i)$ is the average phase angle, of all possible phase angles, in the sector in which r_i lies. The cost-calculation block diagrams for both schemes are given in Figure 4.1.4.

Graph 4.1.6 presents the results for the quantised scheme where r_i is received, which uses Code 3. The degradations in tolerance to noise at various BERs are given in Table 4.1.8 compared with the scheme using infinitely fine quantisation. Clearly the degradations are negligible. The error burst characteristics are very similar to those of the infinitely-finely quantised scheme.

Graph 4.1.7 gives the results for the (phase-quantisation) scheme where $\phi(r_i)$ is received for a scheme using Code 3. The comparison is with schemes using the phase distance measure where infinitely fine quantisation is assumed. Despite the use of unitary distance, all the quantised schemes are degraded in tolerance to noise, compared with their infinitely-finely quantised phase distance measure equivalents. The results at a BER of 1 in 10^3 are outlined in Table 4.1.9. It can be seen that the degradations in tolerance to noise are negligible, although it may have been expected that the degradation due to 8-bit quantisation as compared with infinitely-fine quantisation, may have been more than offset by the use of the unitary distance in the look-up tables. Clearly the main reason for the degradation for schemes where the phase distance measure is used, is not the use of phase as the distance measure, but the loss of information about the magnitude of r_i. Again, the error burst characteristics are very similar to those of the infinitely-finely quantised schemes (see Table 4.1.7).

Graph 4.1.8 contrasts the results of the two quantised schemes which, as noted earlier, are of a similar level of complexity as far as the quantisation is concerned. Clearly, the supposed advantage of representing $\phi(r_i)$ in 8 bits compared with 4 bits per component for the scheme using the true unitary distance, does not lead to an improved performance for the phase-quantisation scheme. In all cases the latter scheme has a lower tolerance to noise. The situation at a BER of 3 in 10^4 is outlined in Table 4.1.10.

Graphs 4.1.9 to 4.1.11 give some idea of the effect of reducing the detection delay, at a BER of approximately 1 in 10^3 for long detection delays. System 1 detection is used for schemes incorporating Code 3. This question was touched upon concerning Figures 4.1.1 to 4.1.3. Of interest in these graphs is the point at which the detection delay becomes too short. As a measure of this, Table 4.1.11 notes for each system, the value of the detection delay at which the BER begins to rise substantially, and the value of the detection delay at which the BER is ten times that for long detection delays. It can be seen that the scheme with $k_1=16$ suffers most due to reducing the detection delay, although the difference compared with the scheme where $k_1=8$ is not very pronounced. The results of Table 4.1.11 and Graphs 4.1.9 to 4.1.11 are to be compared with those of Table 4.1.2, which in part outlines the degradation in tolerance to noise due to a reduction in detection delay. In agreement with Graph 4.1.9, Table 4.1.2 indicates that for $k_1=16$, reducing the delay from 64 to 32 symbol intervals has a negligible effect. Also in agreement with Figure 4.1.11, Table 4.1.2 shows that the effect of reducing the delay for $k_1=4$, from 32 to 16, is more appreciable.

Graphs 4.1.12 to 4.1.14 present the results when constant phase offsets are introduced, (constant phase errors in the receiver estimate of carrier phase), for System 1 detection. In all cases the effects are quite severe, and relatively linear for phase offsets in excess of a few degrees, and less than 15 degrees. Table 4.1.12 gives the phase offsets for which the BER is both 10 times and 100 times the BER for no phase offset. It is evident that the effects of the constant phase offsets are very similar for all the schemes, especially considering the fact that the BER with no phase offset is not exactly the same in all cases. Comparing these results with Graph 3.2.12 for CORPSK(4-7,1+D), which is a 4-phase scheme, the effects of phase offsets for 8-phase modulation are seen to be much more serious.

SCHEME	APPROXIMATE AVERAGE NUMBER OF BIT ERRORS PER BURST, AT GIVEN BER		
	2×10^{-2}	1×10^{-3}	5×10^{-4}
/Det=V16/	17	11	10
/Det=1N8/	36	40	25
/Det=1N4/	142	120	300

TABLE 4.1.1: Error Burst Characteristics for System 1 Detection of Coded 8PSK, Using Code 1

SCHEME	DEGRADATION IN TOLERANCE TO NOISE IN COMPARISON WITH VITERBI DETECTION AT BER = 1×10^{-4} (dB)
/Det=1N16/N=64/	0.7
/Det=1N16/N=32/	0.8
/Det=1N8/N=32/	0.9
/Det=1N4/N=32/	1.65
/Det=1N4/N=16/	2.0 (approx.)

TABLE 4.1.2: Performance of System 1 Detection for Coded 8PSK Using Code 3

SCHEME	APPROXIMATE AVERAGE NUMBER OF BIT ERRORS PER BURST AT GIVEN BER		
	3×10^{-2}	7×10^{-3}	1×10^{-3}
/Det=V64/N=80/	20	13	11
/Det=1N16/N=32/	25	22	14
/Det=1N8/N=32/	30	23	18
/Det=1N4/N=32/	55	53	27
/Det=1N4/N=16/	60	60	52

TABLE 4.1.3: Error Burst Characteristics for System 1 Detection of Coded 8PSK using Code 3

SCHEME	DEGRADATION IN TOLERANCE TO NOISE IN COMPARISON WITH VITERBI DETECTION (CODE 1) AT BER = 1 in 10^4 (dB)
/C=2/Det=1N16/N=64/	0
/C=3/Det=1N16/N=64/	0.4
/C=4/Det=1N16/N=64/	0.65
/C=1/Det=1N8/N=32/	0.85
/C=2/Det=1N8/N=32/	0.7
/C=3/Det=1N8/N=32/	0.6
/C=4/Det=1N8/N=32/	0.6 (approx.)
/C=1/Det=1N4/N=32/	2.2
/C=2/Det=1N4/N=32/	-
/C=3/Det=1N4/N=32/	1.2
/C=4/Det=1N4/N=32/	-

TABLE 4.1.4: Performance of System 1 Detection for Coded 8PSK, for Codes 1 to 4

SCHEME	APPROXIMATE AVERAGE NUMBER OF BIT ERRORS PER BURST, AT GIVEN BER	
	3×10^{-2}	1×10^{-3}
/C=2/Det=1N16/N=64/	93	75
/C=3/Det=1N16/N=64/	25	14
/C=4/Det=1N16/N=64/	121	164
/C=1/Det=1N8/N=32/	36	40
/C=2/Det=1N8/N=32/	-	200
/C=2/Det=1N8/N=32/	30	18
/C=4/Det=1N8/N=32/	344	202
/C=1/Det=1N4/N=32/	140	120
/C=2/Det=1N4/N=32/	850	910
/C=3/Det=1N4/N=32/	55	27
/C=4/Det=1N4/N=32/	>1000	>1000

TABLE 4.1.5: Error Burst Characteristics for System 1 Detection for Coded 8PSK, for Codes 1 to 4

SCHEME	DEGRADATION IN TOLERANCE TO NOISE IN COMPARISON WITH EQUIVALENT UNITARY DISTANCE MEASURE SCHEME AT BER = 1×10^{-4} (dB)
/Det=1N16/N=64/Dis=P/	0.3
/Det=1N16/N=64/Dis=MS/	0.4
/Det=1N8/N=32/Dis=P/	0.5
/Det=1N8/N=32/Dis=MS/	0.65
/Det=1N4/N=32/Dis=P/	0.25
/Det=1N4/N=32/Dis=MS/	0.7

TABLE 4.1.6: Performance of System 1 Detection for Coded 8PSK, for Code 3, Using Suboptimal Distance Measures

SCHEME	APPROXIMATE AVERAGE NUMBER OF BIT ERRORS PER BURST, AT GIVEN BER		
	3×10^{-2}	7×10^{-3}	1×10^{-3}
/Det=1N16/N=64/Dis=E/	25	22	14
/Det=1N16/N=64/Dis=P/	23	18	15
/Det=1N16/N=64/Dis=MS/	24	19	13
/Det=1N8/N=32/Dis=E/	30	23	18
/Det=1N8/N=32/Dis=P/	37	22	13
/Det=1N8/N=32/Dis=MS/	31	26	19
/Det=1N4/N=32/Dis=E/	55	53	27
/Det=1N4/N=32/Dis=P/	50	40	24
/Det=1N4/N=32/Dis=MS/	45	40	30

TABLE 4.1.7: Error Burst Characteristics for System 1 Detection for Coded 8PSK, for Code 3, Using Suboptimal Distance Measures

SCHEME	DEGRADATION IN TOLERANCE TO NOISE IN COMPARISON WITH EQUIVALENT INFINITELY-FINELY QUANTISED SCHEME, AT GIVEN BER (dB)		
	3×10^{-2}	1×10^{-3}	1×10^{-4}
/Det=1N16/N=64/	0.2	0.15	<0.05
/Det=1N8/N=32/	0.2	0.2	0.1
/Det=1N4/N=32/	0.3	0.25	0.1

TABLE 4.1.8: Performance of System 1 Detection for Coded 8PSK, for Code 3, with 4-Bits Per Component Quantisation

SCHEME	DEGRADATION IN TOLERANCE TO NOISE IN COMPARISON WITH THE EQUIVALENT INFINITELY-FINELY QUANTISED, PHASE-DISTANCE SCHEME, AT BER = 1×10^{-3} (dB)
/Det=1N16/N=64/	0.1
/Det=1N8/N=32/	0.1
/Det=1N4/N=32/	0.25

TABLE 4.1.9: Performance of System 1 Detection for Coded 8PSK, for Code 3, for 8-Bit Phase-Quantised Received Samples $\phi(r_i)$

SCHEME	DEGRADATION IN TOLERANCE TO NOISE IN COMPARISON WITH THE EQUIVALENT TRUE EUCLIDEAN COST SCHEME, AT BER = 3×10^{-4} (dB)
/Det=1N16/N=64/	0.2
/Det=1N8/N=32/	0.4
/Det=1N4/N=32/	0.35

TABLE 4.1.10: Performance Comparison for System 1 Detection of Coded 8PSK for Code 3, for the 8-Bit Phase-Quantised Scheme and Equivalent Schemes using the Unitary Distance Measure using 4-Bits Per Component Quantisation

SCHEME	DETECTION DELAY AT WHICH SIGNIFICANT DEGRADATION BEGINS (Symbol Intervals)	DETECTION DELAY AT WHICH BER IS 10 × BER FOR LONG DETECTION DELAYS (Symbol Intervals)
/Det=1N16/	34	19
/Det=1N8/	34	14
/Det=1N4/	27	8

TABLE 4.1.11: Measures of the Effect of Reducing the Detection Delay for System 1 Detection of Coded 8PSK, for Code 3

SCHEME	PHASE OFFSET AT WHICH BER IS 10 × BER FOR ZERO PHASE OFFSET (degrees)	PHASE OFFSET AT WHICH BER IS 100 × BER FOR ZERO PHASE OFFSET (degrees)
/Det=1N16/N=64/	8.5	12.5
/Det=1N8/N=32/	6.5	11.5
/Det=1N4/N=32/	7.0	12.5

TABLE 4.1.12: Measures of the Effects of Constant Phase Offsets for System 1 Detection of Coded 8PSK, for Code 3

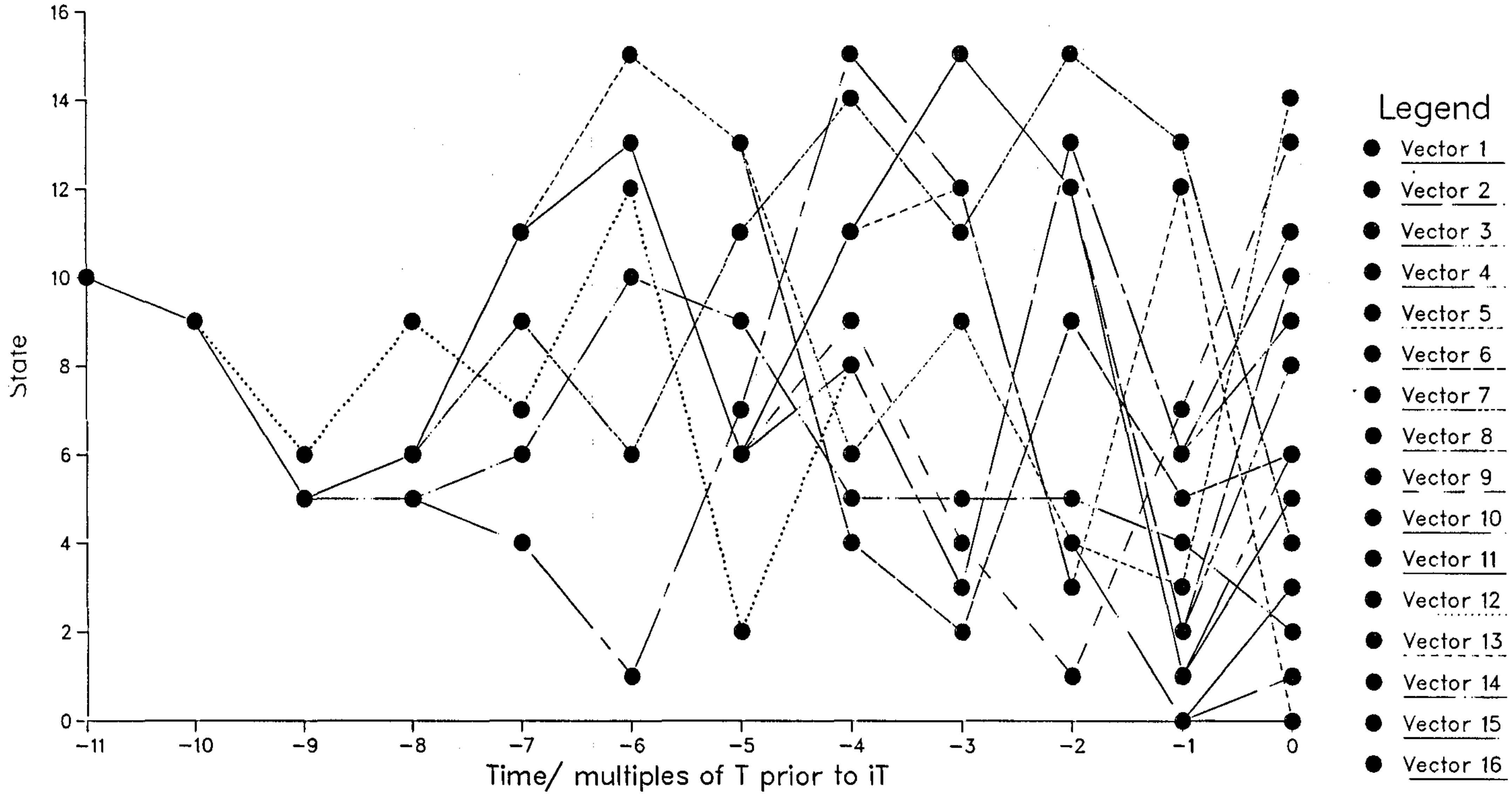


Figure 4.1.1 Code Trellis Diagram for a Scheme using Code 1
System 1 Detection. 16 Stored Vectors. BER=0.003

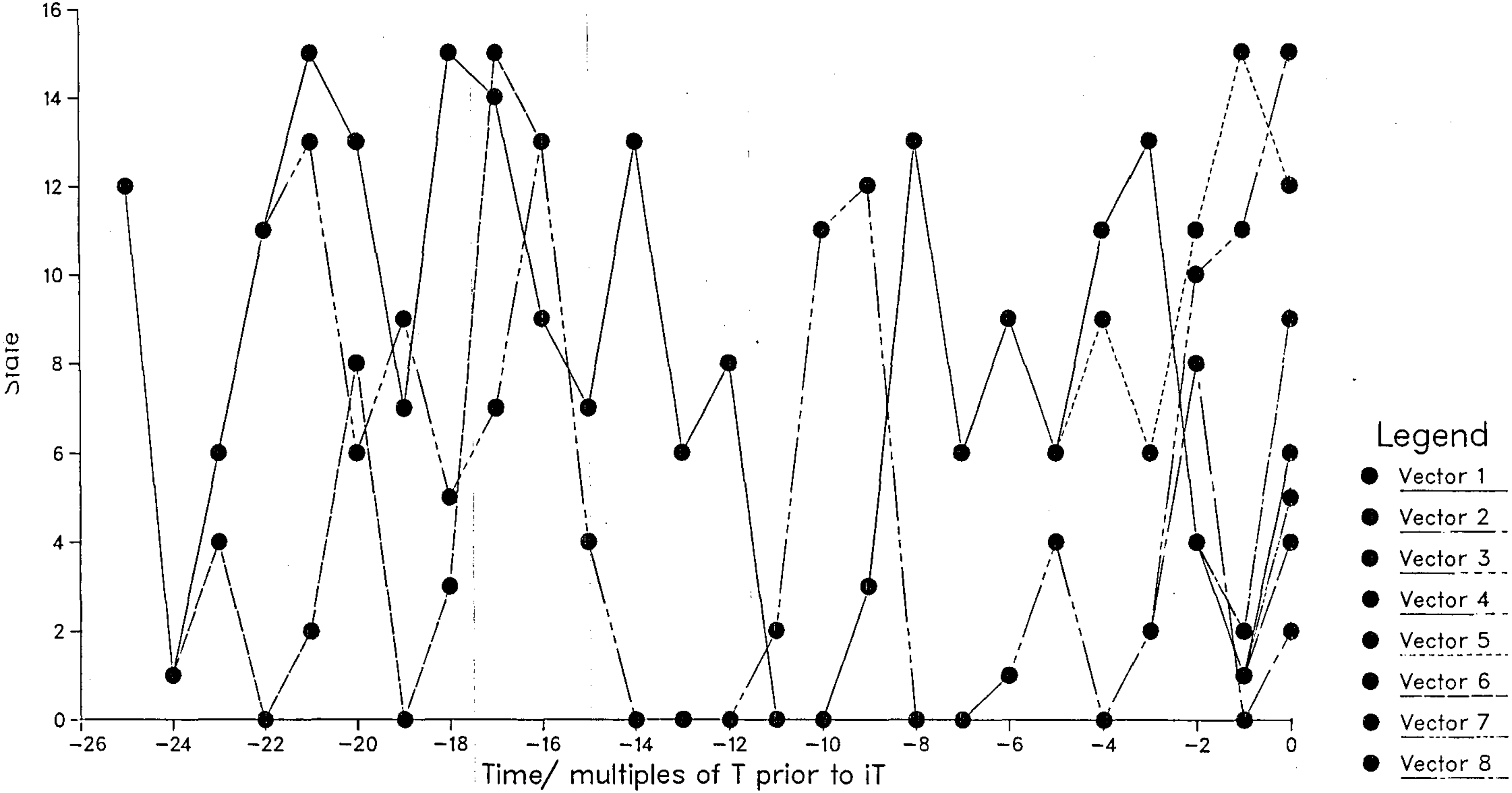


Figure 4.1.2 Code Trellis Diagram for a Scheme using Code 1
System 1 Detection. 8 Stored Vectors. BER=0.01

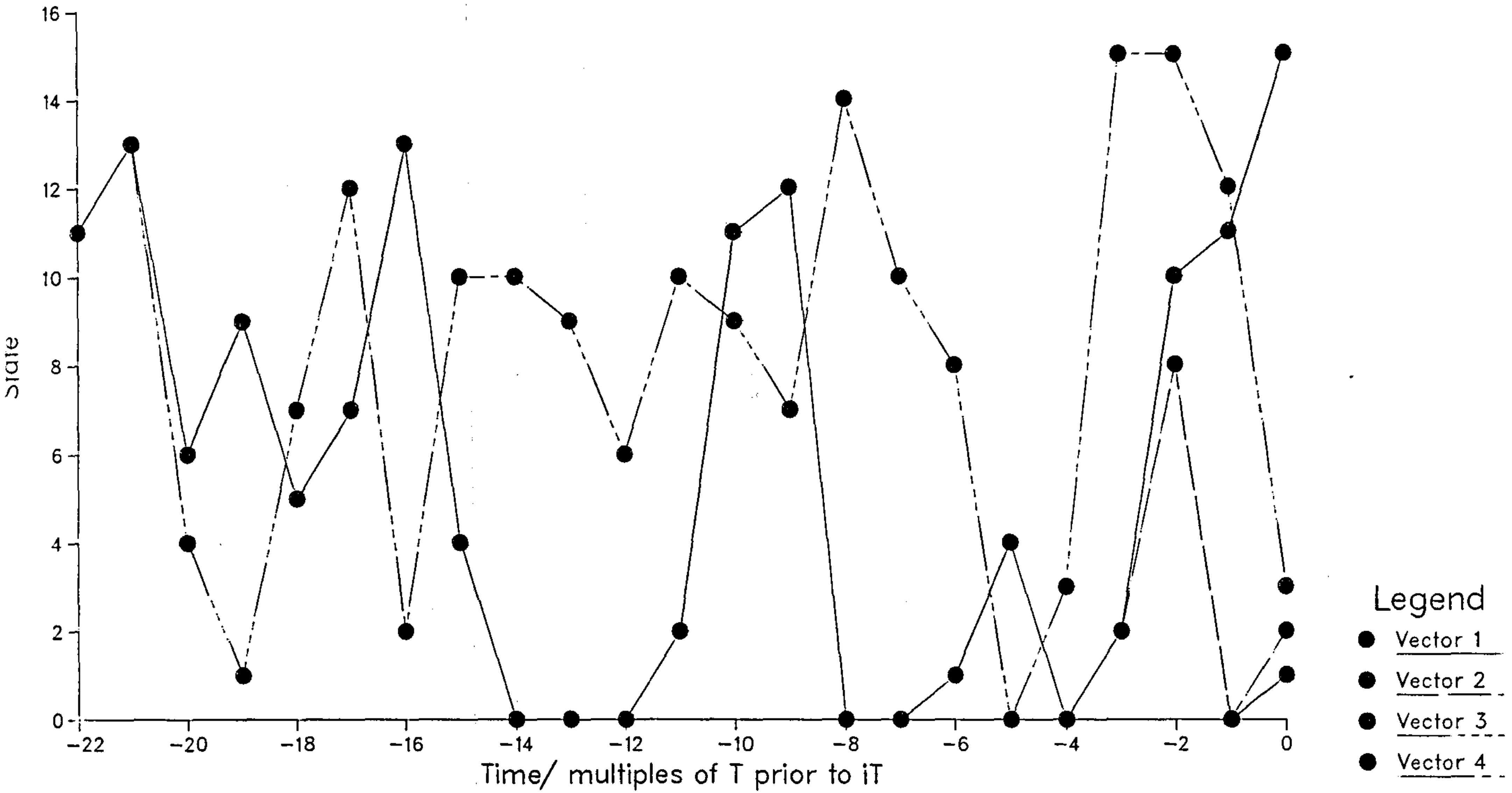
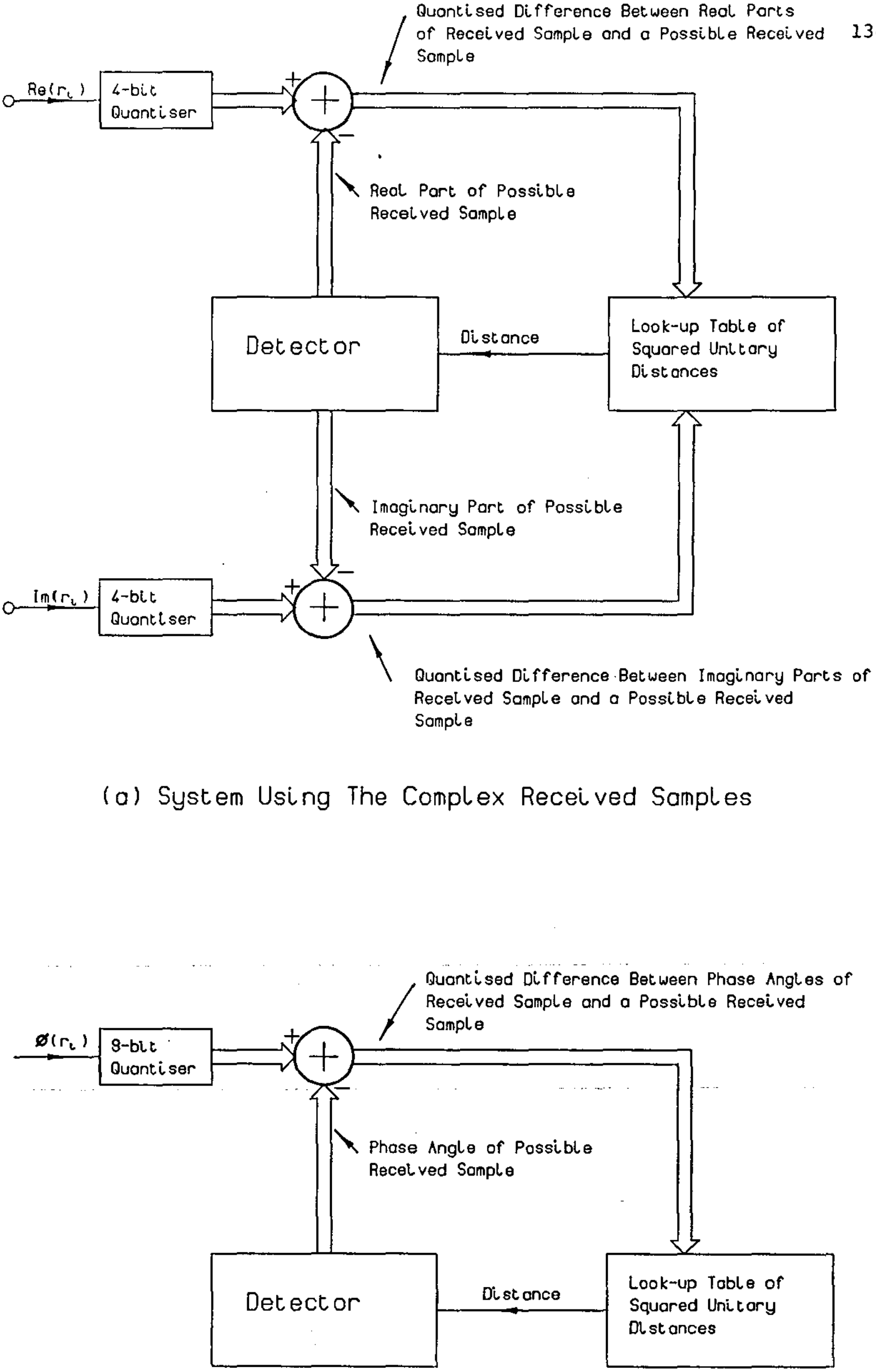


Figure 4.1.3 Code Trellis Diagram for a Scheme using Code 1
System 1 Detection. 4 Stored Vectors. BER=0.07

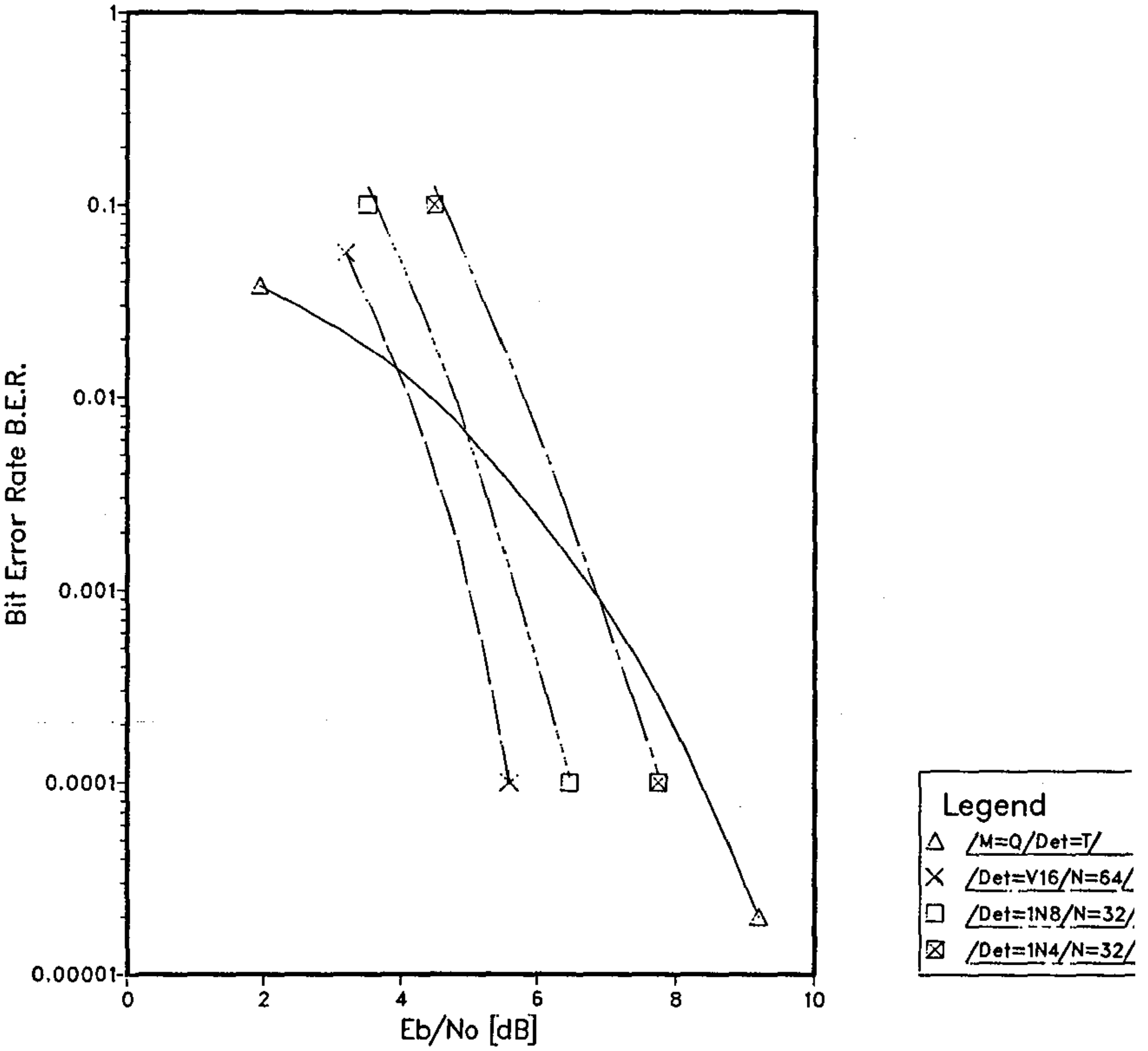

Quantised Difference Between Real Parts of Received Sample and a Possible Received Sample
Re(r_i)
4-bit Quantiser
+
−
Real Part of Possible Received Sample
Detector
Distance
Look-up Table of Squared Unitary Distances
Imaginary Part of Possible Received Sample
Im(r_i)
4-bit Quantiser
+
−
Quantised Difference Between Imaginary Parts of Received Sample and a Possible Received Sample
Quantised Difference Between Phase Angles of Received Sample and a Possible Received Sample
Ø(r_i)
8-bit Quantiser
+
−
Phase Angle of Possible Received Sample
Detector
Distance
Look-up Table of Squared Unitary Distances

(a) System Using The Complex Received Samples

(b) System Incorporating Phase Quantisation

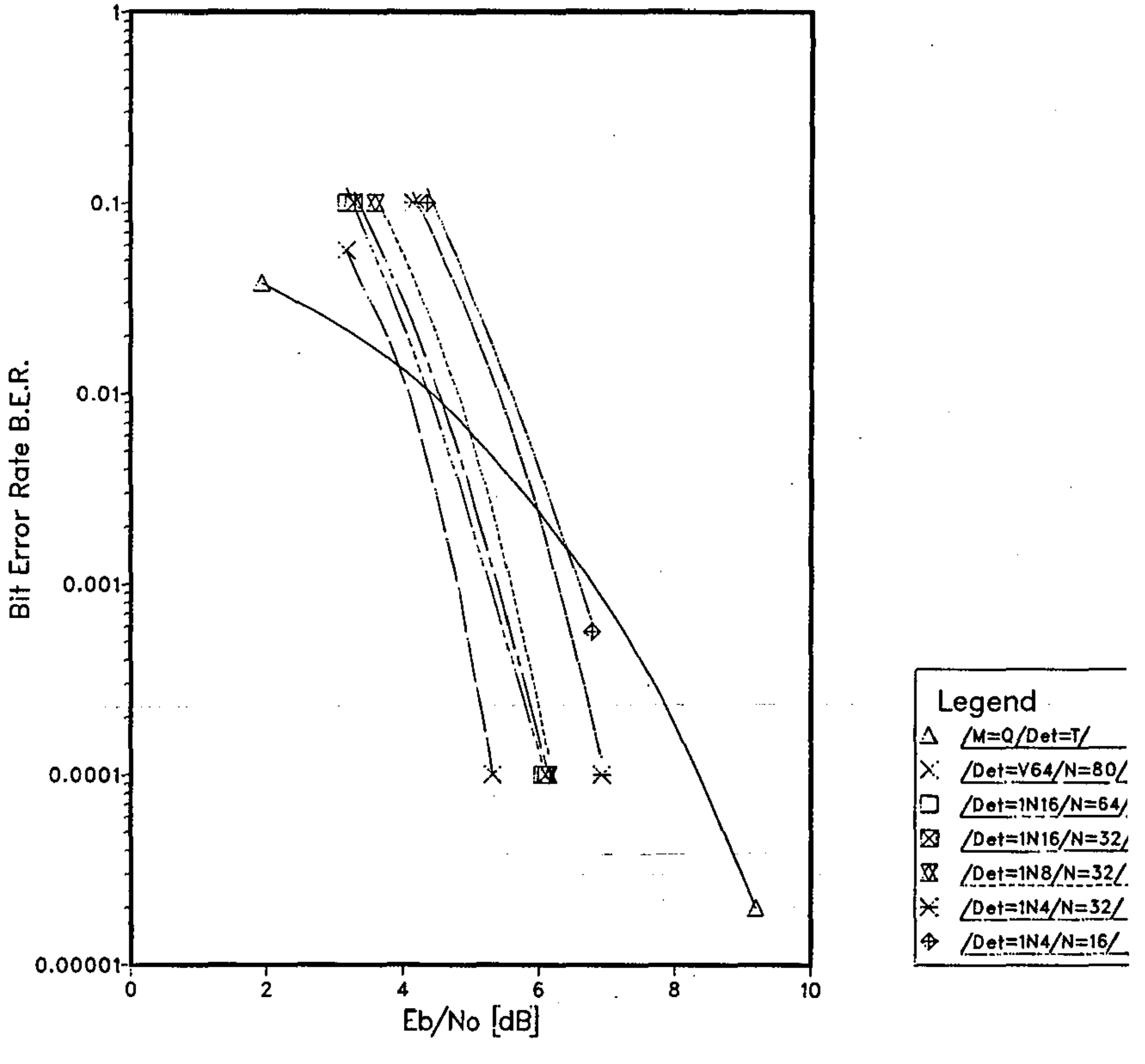
Figure 4.1.4 Block Diagrams of Incremental Cost Determination for The Euclidean Distance and Phase Quantisation Schemes

Graph 4.1.1 Code 1. Near Maximum Likelihood System 1 Detection



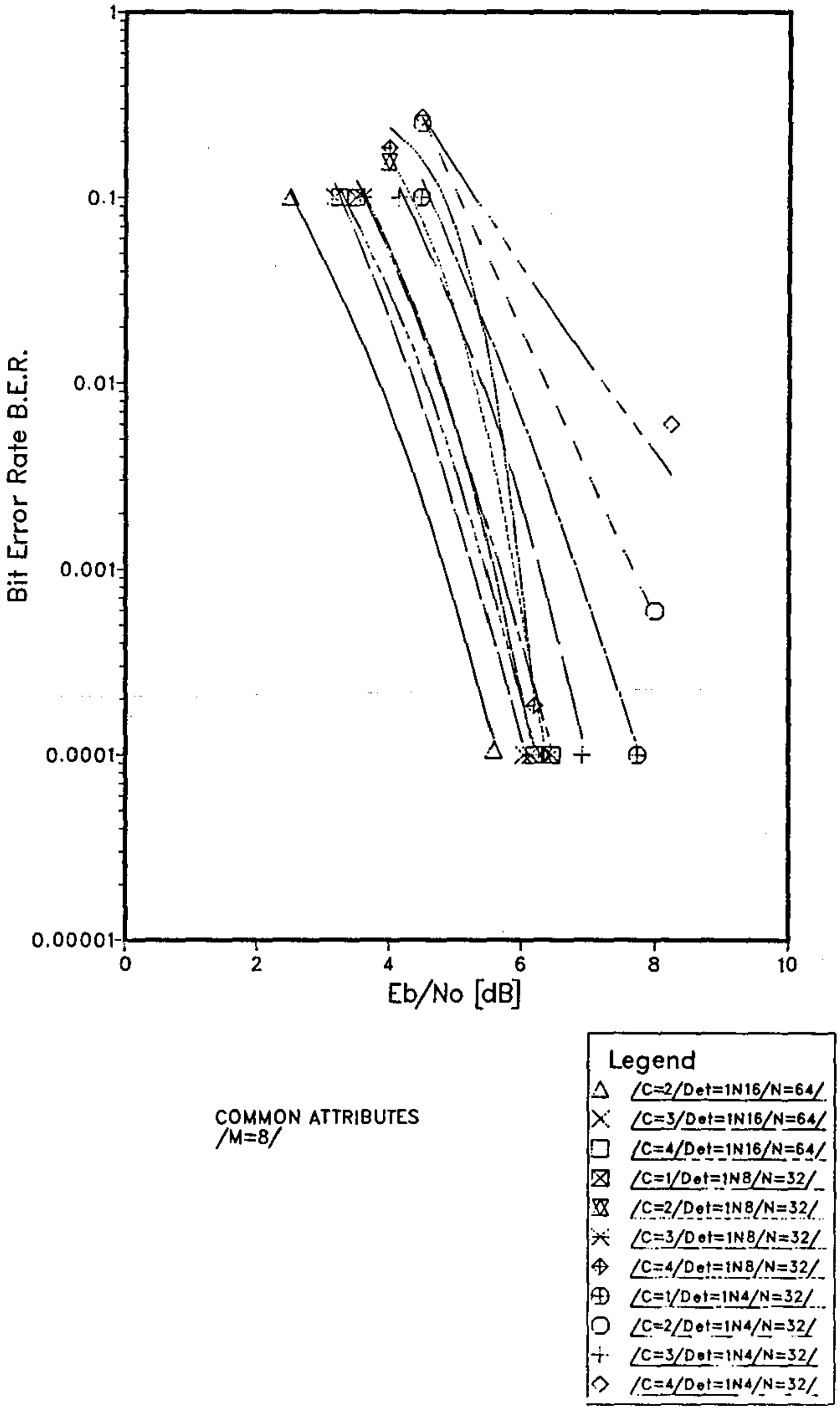
COMMON ATTRIBUTES
/M=8/C=1/

Graph 4.1.2 Code 3. Near Maximum Likelihood System 1 Detection

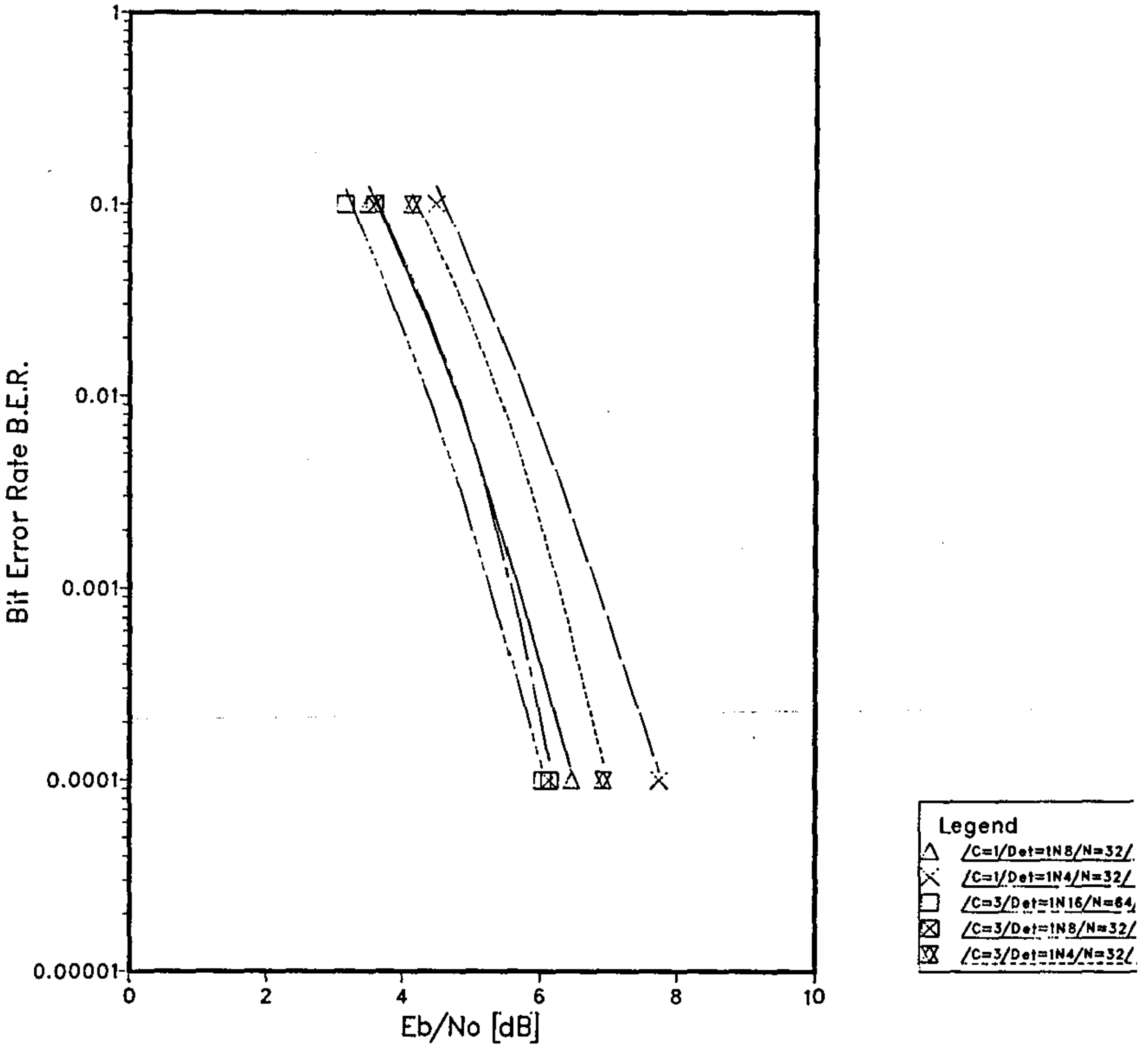


COMMON ATTRIBUTES
/M=8/C=3/

Graph 4.1.3 All Codes. System 1 Detection

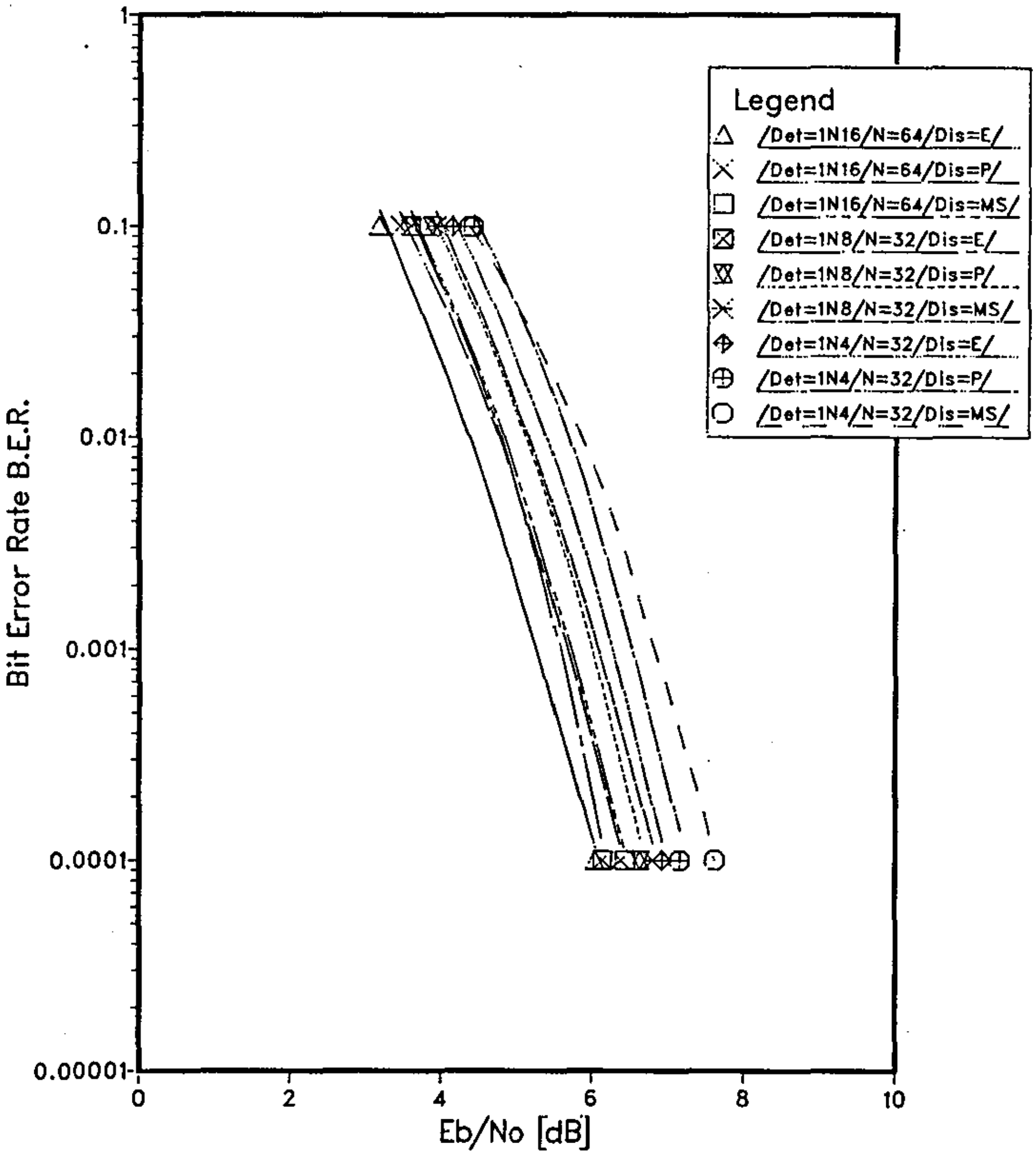


Graph 4.1.4 Comparison of Codes 1 & 3. System 1 Detection



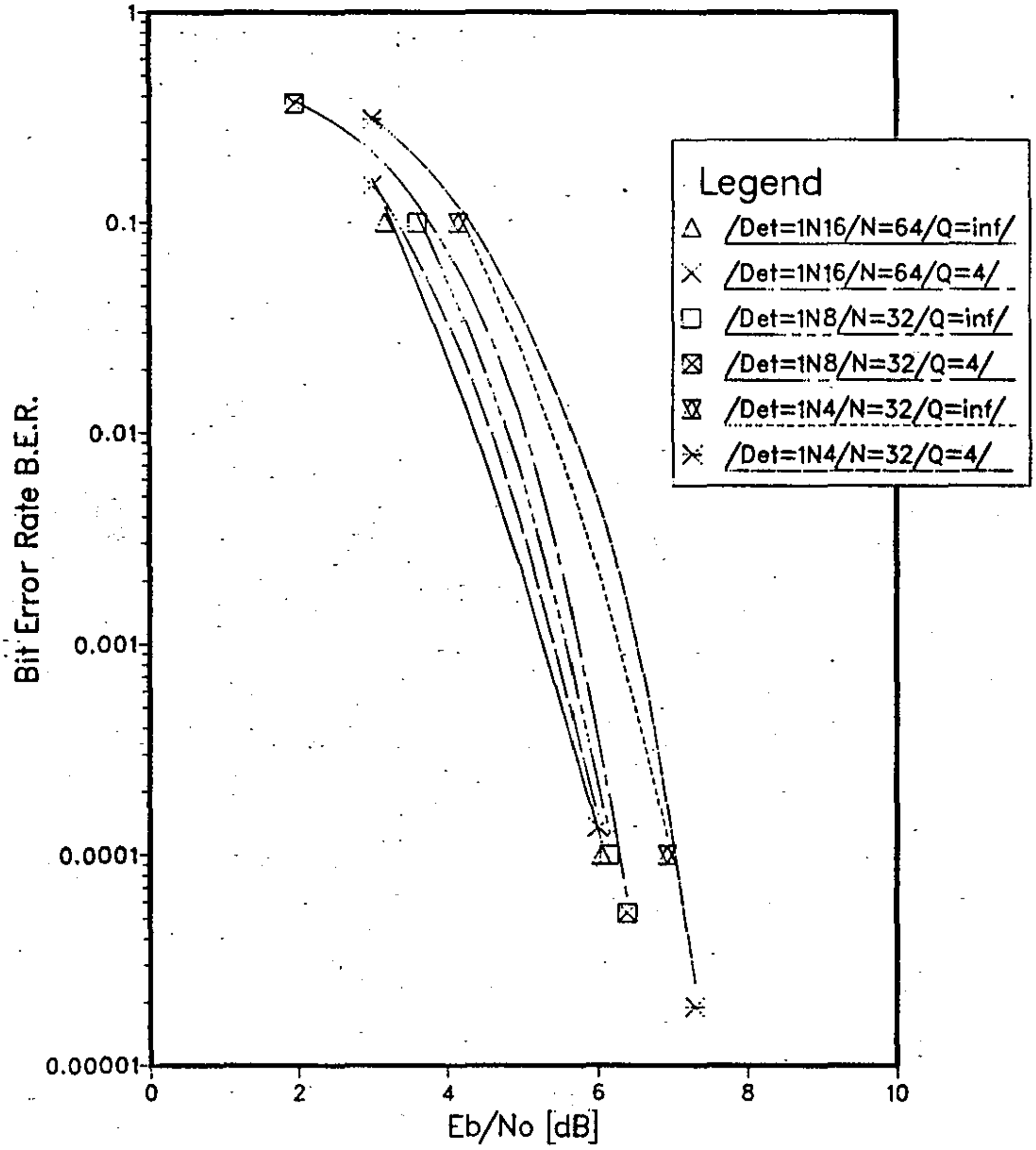
COMMON ATTRIBUTES
/M=8/

Graph 4.1.5 System 1 Detection. Various Distance Measures



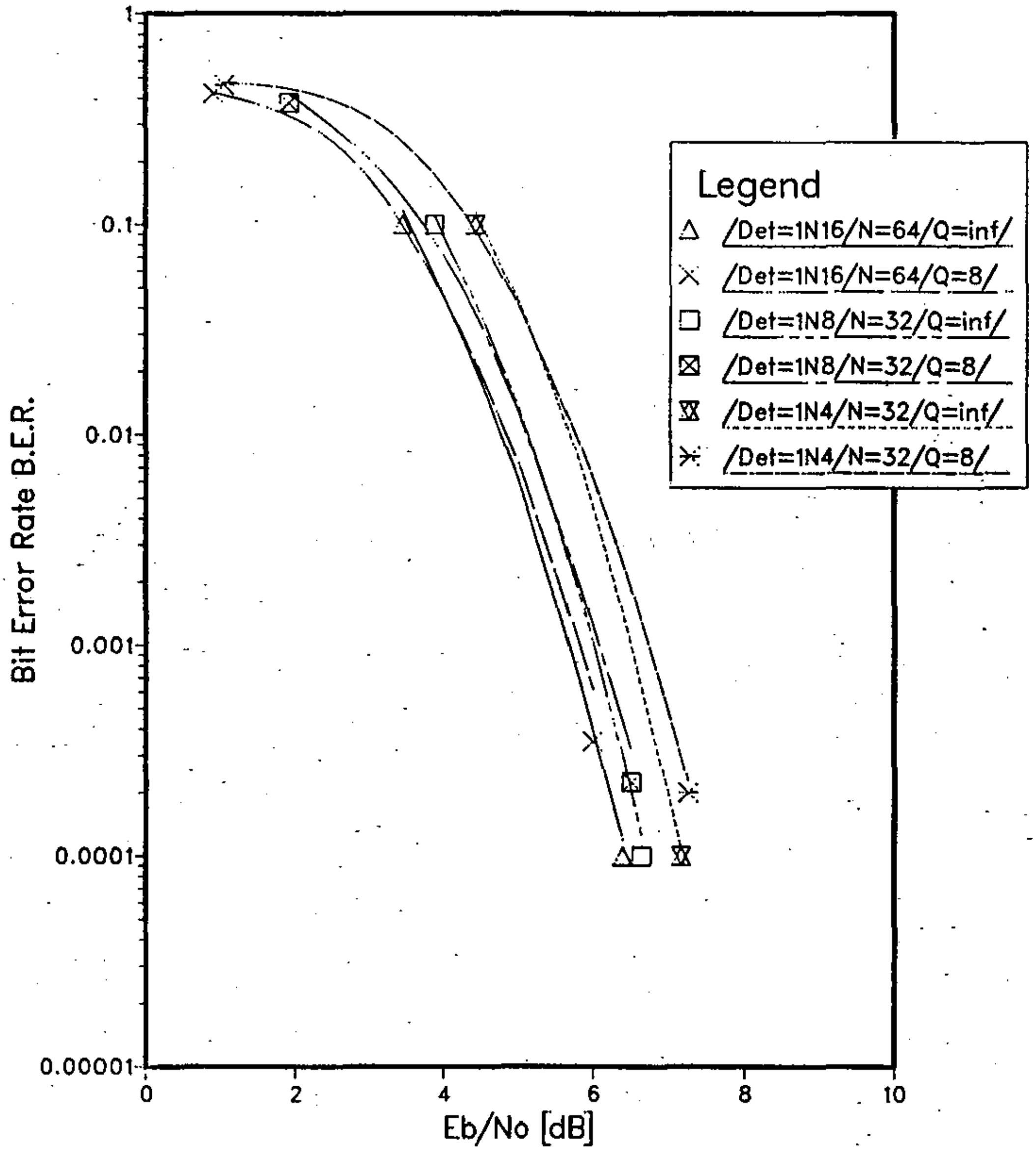
COMMON ATTRIBUTES
/M=8/C=3/

Graph 4.1.6 System 1 Detection. Two Dimensional Quantisation



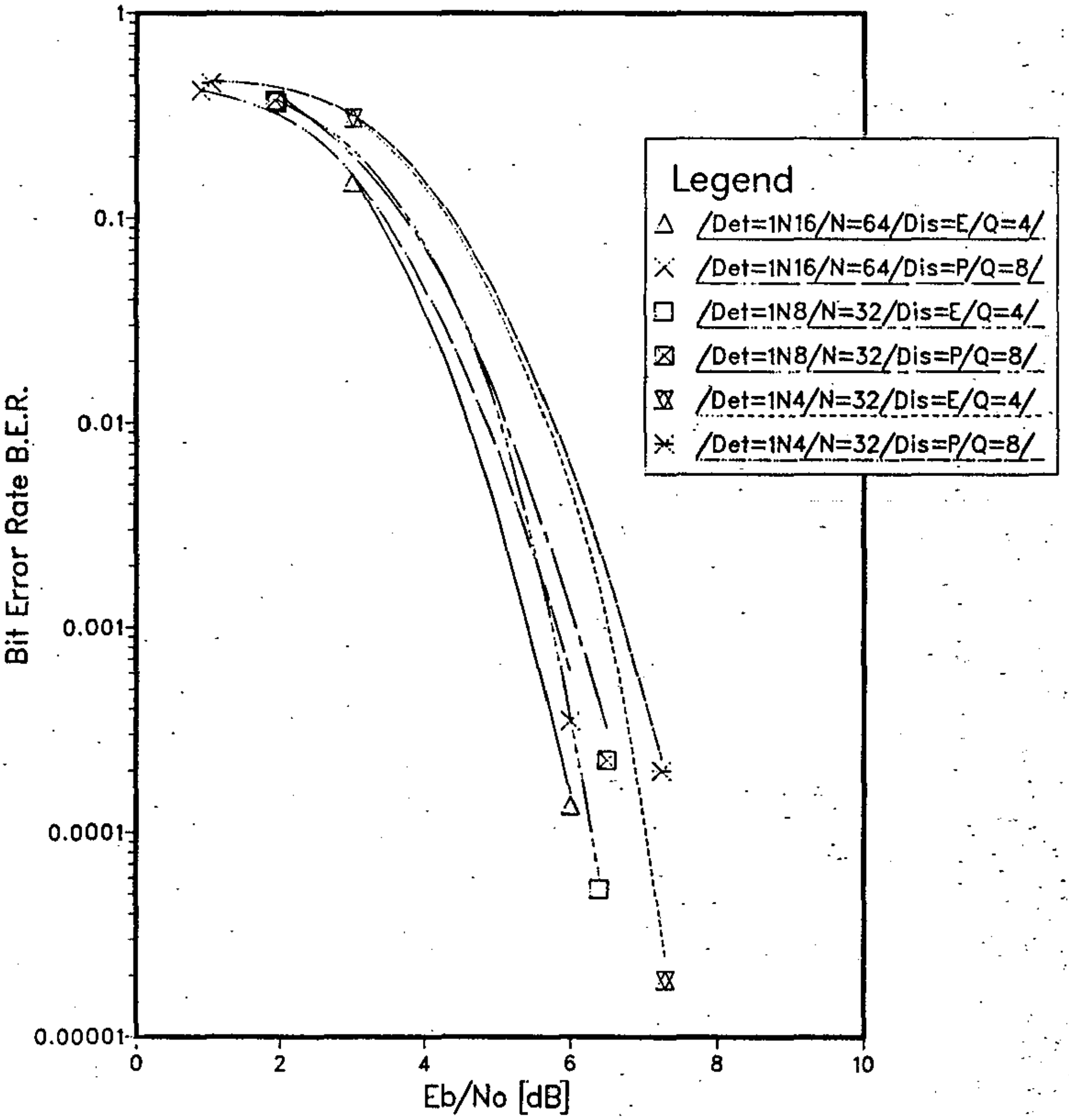
COMMON ATTRIBUTES
/M=8/C=3/Dis=E/

Graph 4.1.7 System 1 Detection. Phase Quantisation



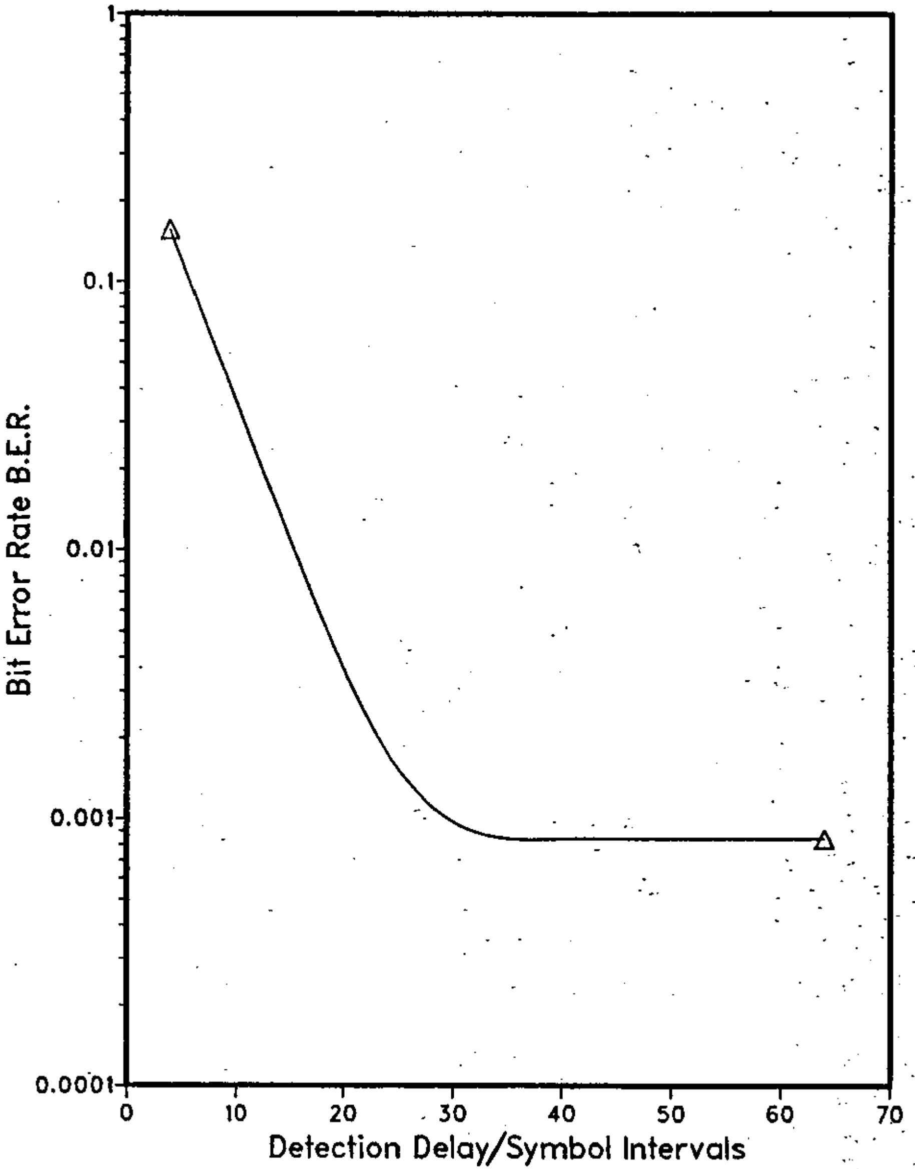
COMMON ATTRIBUTES
/M=8/C=3/Dis=P/

Graph 4.1.8 System 1 Detection. Comparison of Phase & Two Dimensional Quantisation



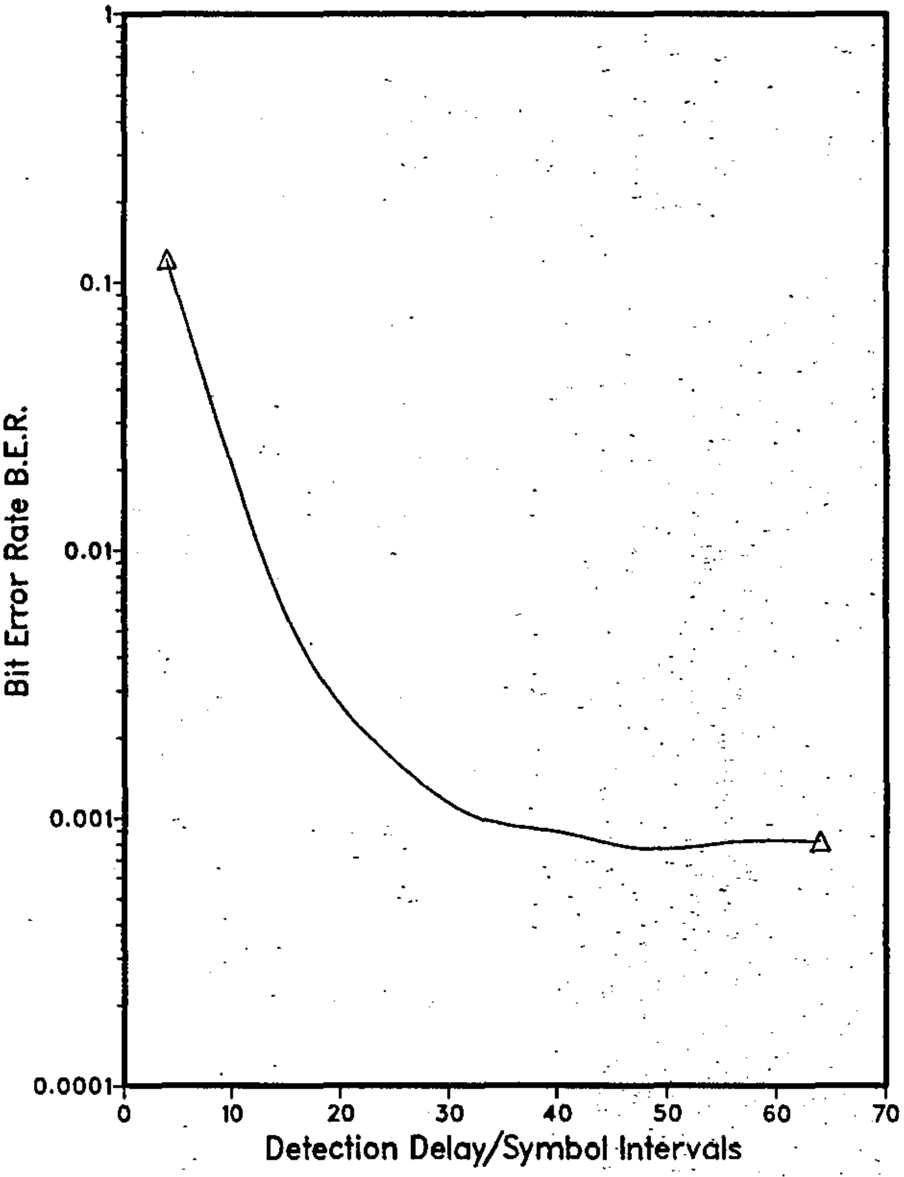
COMMON ATTRIBUTES
/M=8/C=3/

Graph 4.1.9 Variation of B.E.R. with Detection Delay at Eb/No=5.3dB



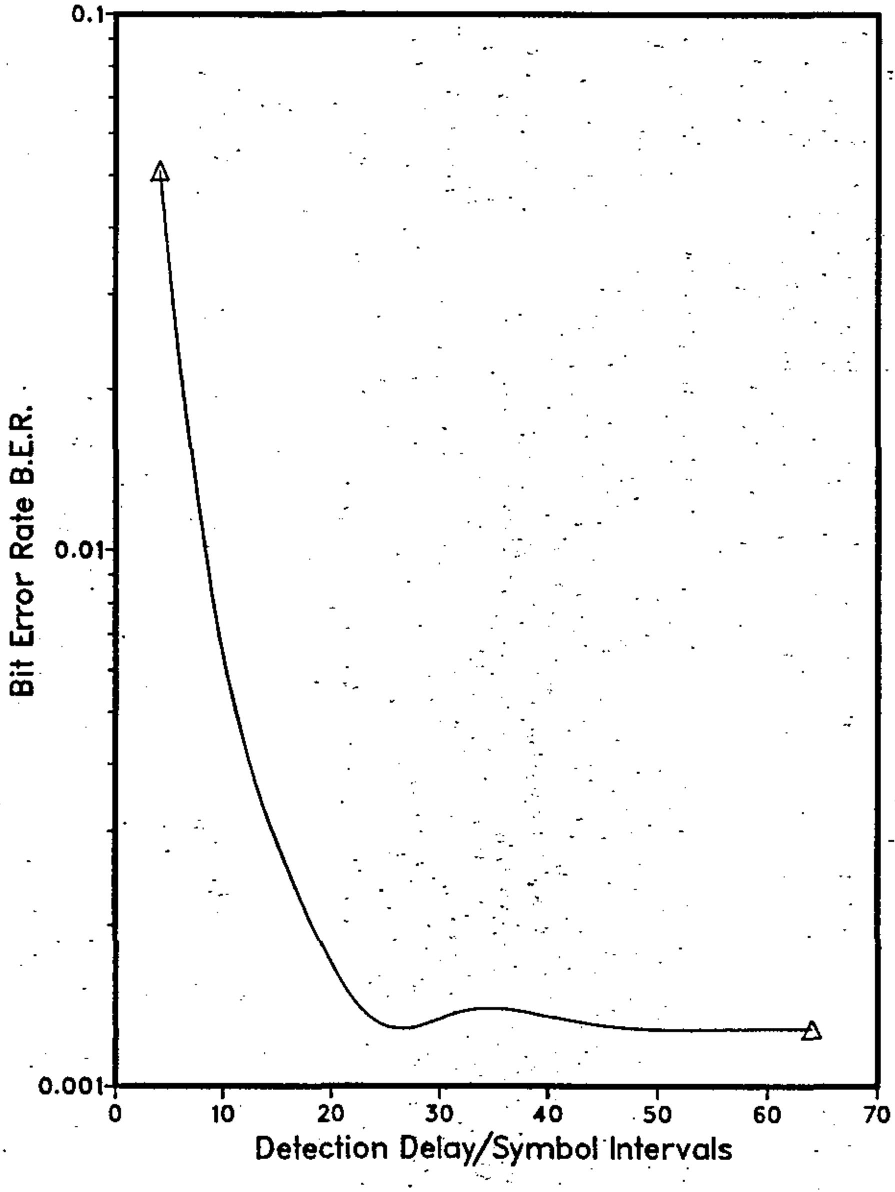
SYSTEM ATTRIBUTES
/M=8/C=3/Det=IN16/

Graph 4.1.10 Variation of B.E.R. with Detection Delay at Eb/No=5.6dB



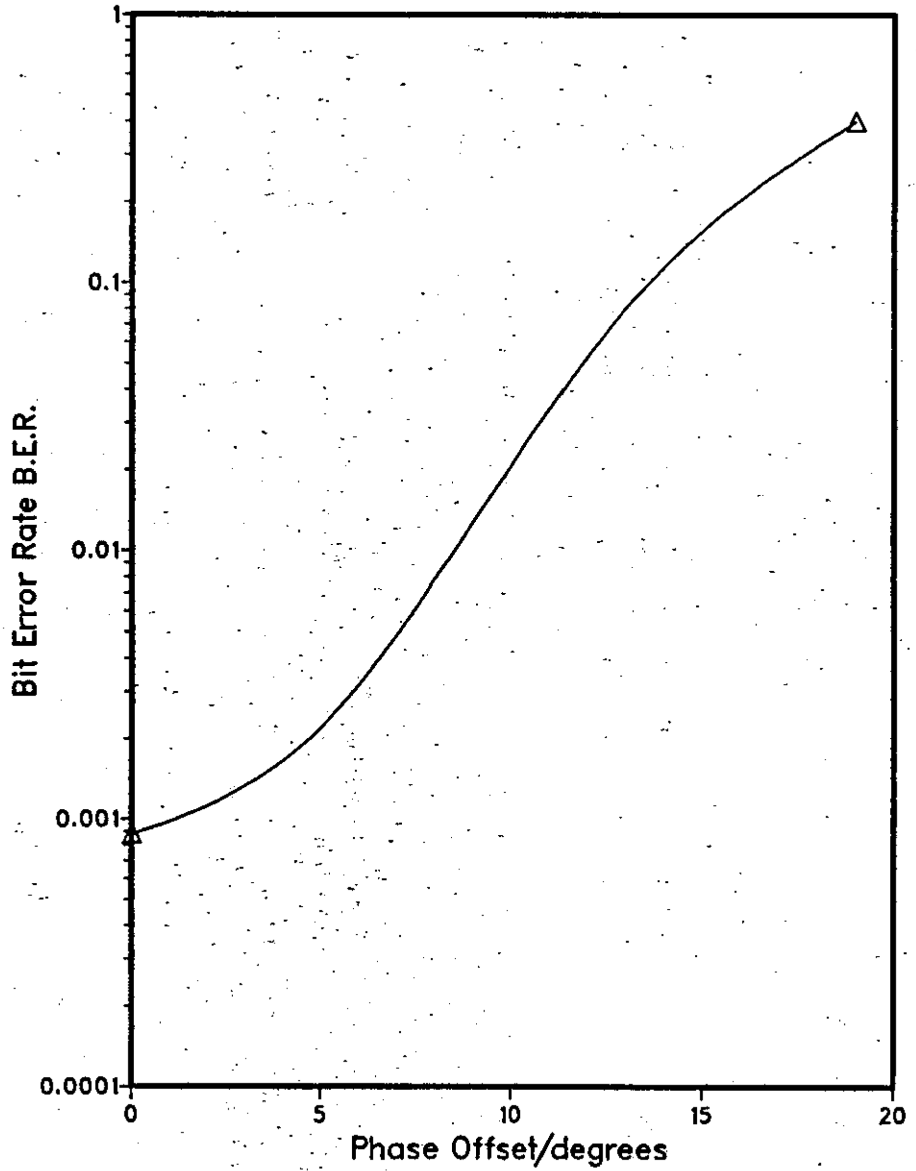
SYSTEM ATTRIBUTES
/M=8/C=3/Det=IN8/

Graph 4.1.11 Variation of B.E.R. with Detection Delay at Eb/No=6.3dB



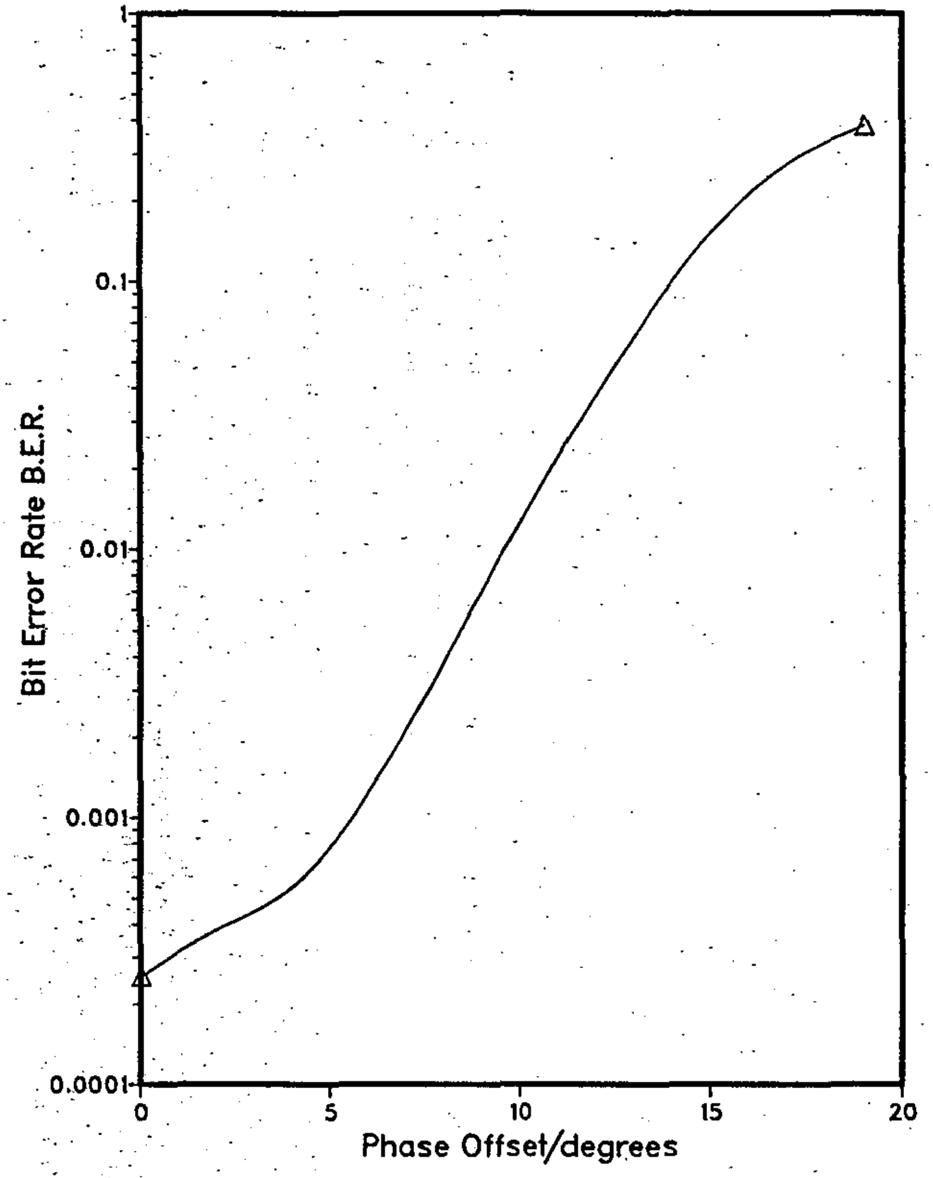
SYSTEM ATTRIBUTES
/M=8/C=3/Det=IN4/

Graph 4.1.12 Variation of B.E.R. with Received Constant Carrier Phase Offset at Eb/No=5.3dB



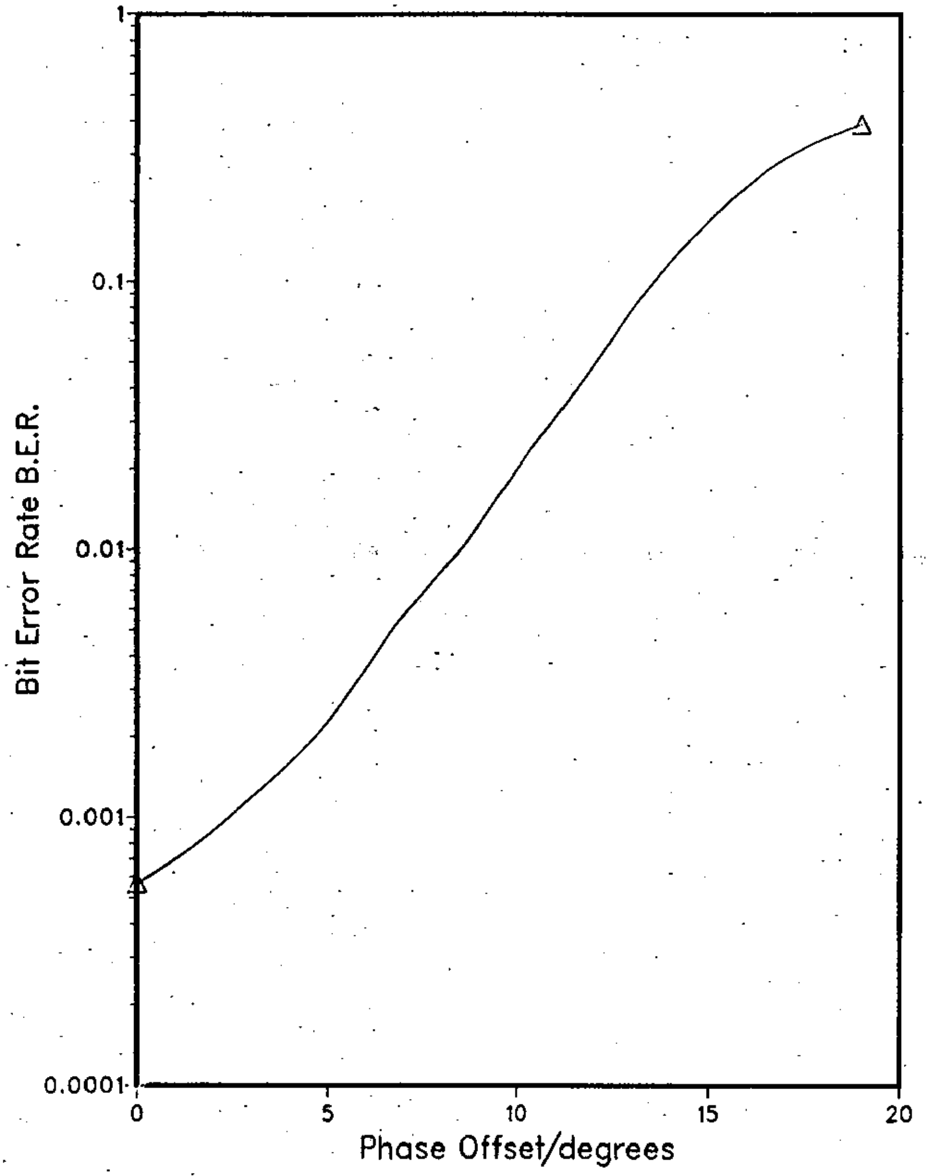
SYSTEM ATTRIBUTES
/M=8/C=3/Det=1N16/

Graph 4.1.13 Variation of B.E.R. with Received Constant Carrier Phase Offset at Eb/No=6.0dB



SYSTEM ATTRIBUTES
/M=8/C=3/Det=IN8/

Graph 4.1.14 Variation of B.E.R. with Received Constant Carrier Phase Offset at Eb/No=6.3dB



SYSTEM ATTRIBUTES
/M=8/C=3/Det=1N4/

4.2 SYSTEM 3

This detector is another of those initially investigated by A.P. Clark et al[64,65] and developed in a number of ways since.[28,66-68] It is referred to as System 3. The description of the detector begins with a description of the received signals. The detector is then described in terms of its stored vectors and costs. The algorithm, repeated during every symbol interval, which uses these stored values to produce detected data symbols, is described. The unitary distance measure (Appendix A7) is used for the stored costs. From Section 4.1 the complex received sample at the input to the detector at time t=iT is,

$$r_i = p_i + w_i \tag{4.2.1}$$

The detector holds in store k_3 vectors $\{Q_i'\}$, of possible values of the data symbols. At time t=(i-1)T these vectors have the form

$$Q_{i-1}' = [q_{i-N+1}', q_{i-N+2}', \ldots, q_{i-1}'] \tag{4.2.2}$$

where q_ℓ' is a possible value of the data symbol q_ℓ.

k_3 is a multiple of 4, which is the number of different values that q_ℓ' can have. The expansion, coding, mapping and costing processes, are analogous to those described in Section 4.1, and are not repeated here. The System 3 selection algorithm, which selects k_3 vectors $\{Q_i'\}$ from the $4k_3$ expanded vectors derived from the vectors $\{Q_{i-1}'\}$, is as follows. For each of the four possible values of q_{i-h+1}', where $h=k_3/4$, the detector selects the expanded vector with the given value of q_{i-h+1}' and with the smallest value of $|W_i'|^2$. The process is repeated in turn for $q_{i-h+2}', q_{i-h+3}', \ldots, q_i'$, the process being such that an expanded vector once selected is not available for selection again, so that no expanded vector may be selected more than once. After the selection process is completed, the minimum cost is subtracted from all costs to prevent their stored values overflowing, and the value of q'_{i-N+1} in the vector with the lowest cost is taken to be the detected value of q_{i-N+1}. The process continues in this way.

This system can be seen (loosely) as a redefinition of the meaning of a state, compared to the previous definition (Sections 3.2 and 4.1). Instead of defining a state as a combination of the values of a number of the most recent symbols, q'_{ℓ}, System 3 defines a state in terms of the position, (time $t=(i-\ell)T$), and value of a given symbol $q'_{i-\ell}$. Clearly this is not rigorously equivalent to a true Finite-State Machine representation[72], since a vector with a 'state' corresponding to a particular value of say, q'_{i-h}, will clearly also have a particular value of q'_{i-h+1}, but it will not be taken to have the 'state' corresponding to this particular value of q'_{i-h+1}. Another completely separate vector will have the 'state' corresponding to this value of q'_{i-h+1}.

System 3 is an attempt to overcome a weakness of System 1. As has been shown in Section 4.1, it is quite possible that at any time, all the k_1 stored vectors have the same values of $q'_{i-\ell}$, for some $\ell=1,2,\ldots$, so that $q'_{i-\ell}$ is effectively detected with only a correspondingly short delay. This was clearly shown in the code trellis diagrams of Figures 4.1.1 to 4.1.3. By using a system which constrains the vectors to be somewhat different, at least over a span of the h most recent symbols, it is hoped that this problem may be alleviated.

The results of the simulation tests conducted with System 3 detection for coded 8PSK, are given in Graph 4.2.1. This is a graph of bit error rate (BER) as the signal to noise ratio, E_b/N_O, varies. E_b is the average energy transmitted per bit. $N_O/2$ is the two-sided power spectral density of the additive white Gaussian noise. (See Appendix A5 for more details of the simulation techniques. Appendix A8 outlines the notation used to describe the schemes which were tested.) Both Codes 1 and 3 are used. The accuracy of the results are of the order of ±0.25dB in the range of BER, 1 in 10^3 to 1 in 10^4. Clearly from Graph 4.2.1, System 3 detection loses substantially in tolerance to noise, compared with System 1 detection. The results at a BER of 1 in 10^3 are given in Table 4.2.1.

The loss in performance due to the substitution of System 3 for System 1 is at first sight surprising, since the published results[28,64,65] (mainly concerning channels with intersymbol interference, but also involving some convolutionally coded schemes), have shown that System 3 usually fa~~i~~res better than System 1. To gain an insight into a possible mechanism for the relatively poor performance of System 3 in comparison with System 1, consider again the Viterbi Algorithm detector. Whenever the latter selects between a number of possible vectors, (choosing one and discarding the remainder), it is certain that whatever the future received samples $\{r_i\}$ may be, the chosen vector would always at each stage have the lowest cost of the vectors concerned in the present selection, if the remainder of the vectors were not discarded, (see Appendix A4). Therefore the discarded vectors will never provide detected data symbols, since none of their costs can ever be the overall lowest cost. System 3 on the other hand, selects between vectors on the basis of the value of just one symbol, q'_{i-h}, being the same in all the vectors involved in the selection process. There is absolutely no guarantee, if the non-selected vectors were not discarded, that some of their future costs would not be lower than the cost of the chosen vector, for some sequences of future received samples $\{r_i\}$. If such a retained vector were to have a lower cost than the actually chosen vector, this vector could conceivably, at some later stage, have the lowest cost and provide detected data symbols. In other words there is no guarantee that a discarded vector in one of the selection processes of System 3 is not the Maximum Likelihood vector, (see Appendix A3). Clearly this is also valid for System 1. The relative performance of the detectors is a function of how easy or difficult it is for the detector to discard vectors when they could possibly, (in the future, if retained), have lower costs than the chosen vector. System 1 simply chooses the k_1 expanded vectors with the k_1 lowest costs, at each stage. The larger k_1 is, the more unlikely it is that a discarded vector is one which could have the lowest cost, at a later stage. This is because, the larger k_1 is, the larger will be the cost of the discarded vector, of all the discarded vectors, with the lowest cost. The greater the difference in cost between the overall lowest cost and the cost of this discarded vector, the more unlikely it is that the latter vector could have the overall lowest cost at some future stage. The situation is somewhat different for System 3 and other detectors, (for example the pseudobinary detector of Section 4.3 and the detectors of Section 5.3). In these cases a number of separate selections, by means of the rankings of costs, takes place. For example for System 3 with

k_3 stored vectors, k_3 such separate selection processes take place. Therefore, a situation can be envisaged where one such selection process is amongst a number of expanded vectors with high costs, whereas another is among a number of expanded vectors with low costs, each of the costs involved in the second process being lower than every cost in the first process. Since the cost using the unitary distance measure is a good measure of the likelihood that a given vector is correct, it may well be that all the vectors which were not chosen in the second selection process, are more likely to be correct than the one chosen vector in the first selection process. Superficially, the same could be said for Viterbi detection. The major difference for the Viterbi detector is that, in the second selection process described above, each of the vectors which are not chosen is guaranteed not to be the Maximum Likelihood vector, as described earlier. Clearly this cannot be guaranteed for System 3. System 3 simply ensures some variety in the stored vectors, as described earlier. High likelihood vectors can be discarded during the selection process.

A number of points with regard to the modulation method, coded and phase mapped 8PSK, give further insight into possible reasons for the relatively poor performance of the System 3 detectors. Compare a scheme using a non-systematic convolutional code, with the transmission of four-level data over a linear channel introducing intersymbol interference. For the latter, having a sampled impulse response with (g+1) components, a received sample of the signal in the absence of noise has 4^{g+1} possible values, some of which may be very close together but no two of which are likely to be exactly the same. In principle one received sample can be used to achieve the unique detection of the (g+1) four-level data symbols involved in that sample. On the other hand, with a binary Rate-2/3 non-systematic convolutional code[19] as implemented here, the sample involved in any one detection process can have one of only eight values, (one of the eight possible values of p_i). Therefore no one data symbol can be detected from one isolated received sample. It is the particular characteristics of the mapping function of the code symbols $\{c_i\}$ onto the complex numbers $\{p_i\}$, (Section 2.5), which increase the likelihood of discarding wanted vectors. When a given vector is expanded, the values of p_i' produced, all belong to one of two sets. The two sets (A and B) are shown in Figure 4.2.1. Each such set comprises, in its own right, a QPSK constellation. If the received sample is closest to a given value of p_i' in set A of Figure 4.2.1, then if the expanded vectors of a given vector give values of p_i' in set A, one of the $\{p_i'\}$ is that which is nearest to the received sample. Conversely, if the expanded vectors give values of p_i' in set B, none of the $\{p_i'\}$ will be that which is nearest to the received sample. Therefore for an arbitrary vector, the likelihood that one of the $\{p_i'\}$ produced upon its expansion is that closest to the received sample, is about 1/2, since all the $\{p_i\}$ are equally likely.[20] The result is that quite a few of the elements of two stored vectors may be different, while the costs of the two vectors in the absence of noise may be very similar. This is because the distance between the corresponding sequences of the $\{p_i'\}$ is small. A System 3 detector may have to choose between two such vectors, but since the distance between the two sequences of the $\{p_i'\}$ is small, the chosen vector may be the wrong one. Clearly, the minimum distance properties of the scheme will eventually ensure a reasonably large distance between the two vectors, but this distance may not have been built up before the detector has to choose between the two.

As noted earlier, in the case of the Viterbi detector such a choice can be made in the complete confidence that the discarded vector will never have the lowest cost. Considering System 1 in the same light, it is very unlikely that one of the two above vectors will be discarded, while the other is retained, if they have similar costs. In the vast majority of cases either both will be retained, (if they have low costs), or both will be discarded, (if they have high costs). In the former case System 1 assumes that both vectors have a high likelihood, whereas in the latter case System 1 assumes that both have a low likelihood. Since these assumptions are based on cost relative to all $4k_1$ expanded vectors, which is a good measure of likelihood, they are likely to be correct in most cases.. On the other hand, System 3 restricts the comparisons to be between only a few expanded vectors and a cost-comparison involving the whole set of $4k_3$ expanded vectors does not take place as part of the selection process.

A related question of importance, is the ability of a detector to recover, once an error has been made. The signal characteristics discussed above also hamper such recovery, for Viterbi, System 1, and System 3 detectors. Therefore, in all cases, an initial error is followed by a number of further errors which are a direct result of the first error, (an error burst). Once an error has been made because the detector has discarded the correct vector, a number of symbol intervals pass where the values of the symbols $\{q_i'\}$ appended to the lowest-cost vector, are not equal to the values of the $\{q_i\}$. This number of symbol intervals in the case of coded and phase mapped 8PSK, depends on a number of points. The code itself may lead to this time period being long by providing a number of possible future sequences of data symbols with relatively small, and relatively similar costs. The corresponding code sequences are usually termed near-minimum distance sequences (Appendices A3 and A4 and References 12,19,21 and 74). This is discussed more fully in Section 4.1. Therefore, under noisy conditions, this period of time may be quite long even for Viterbi detection (Section 3.2 and Reference 12). The problem is increased by the characteristic of the mapper (which converts the $\{c_i\}$ into the $\{p_i\}$), described above, which may produce a number of stored vectors with similar costs. The Viterbi Algorithm has the following advantage over the other detectors. It forces the consideration of a vector with element values that are the same as the transmitted data symbol values, for the most recent (k-1) symbols, (where k is the code constraint length). It does this by ensuring that all possible combinations of the last (k-1) data symbol values exist within the stored vectors. This does not, of course, guarantee convergence. A vector which has element values which are the same as the transmitted data symbol values over the last ℓ symbol intervals, where $\ell>(k-1)$, may be discarded in preference to a vector which has element values which are the same as the transmitted data symbol values over the most recent (k-1) symbols.

The error burst characteristics of System 3 are contrasted against those of System 1, for schemes using Codes 1 and 3, in Table 4.2.2.

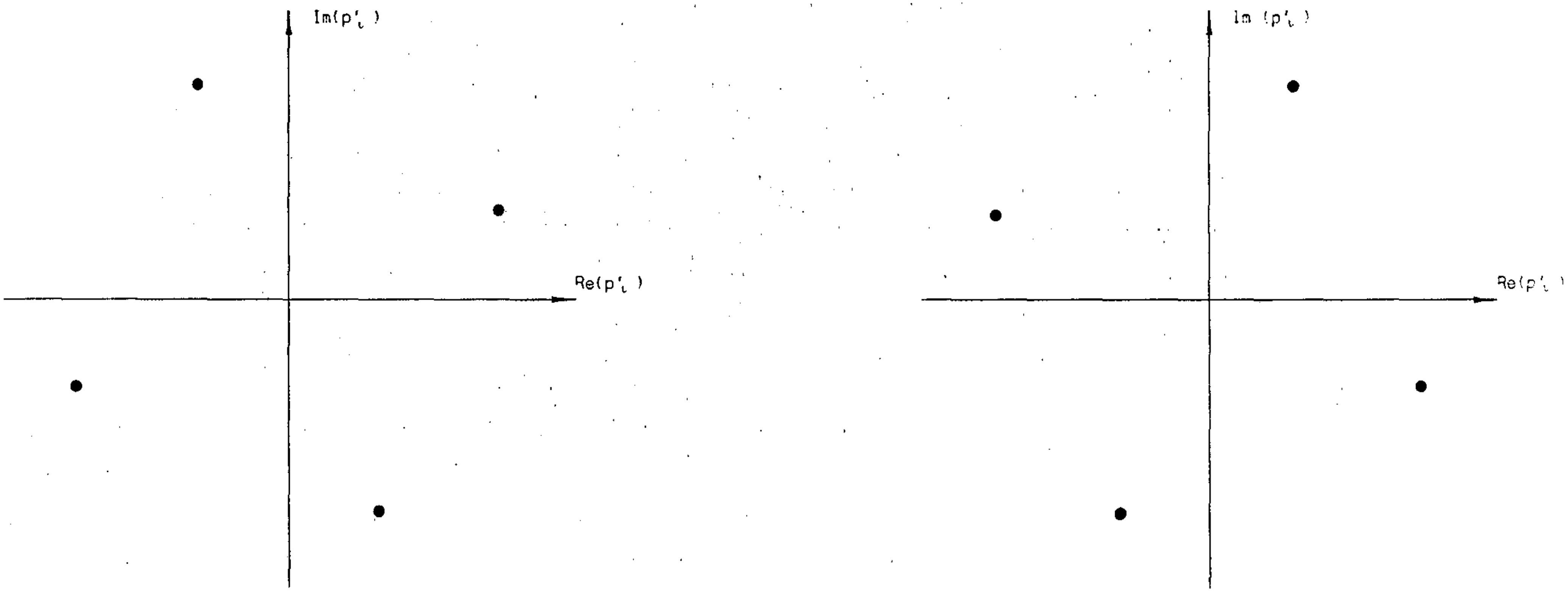
Appendix A5 defines an error burst. Clearly, System 3 has a larger number of bit errors per burst than does System 1, for the same number of stored vectors. The increased number of errors per burst is probably due to the characteristics of System 3 discussed earlier.

SCHEME	SIGNAL TO NOISE RATIO (E_b/N_o) AT WHICH BER = 1 × 10^{-3} (dB)
/C=1/Det=1N8/N=32/	5.75
/C=1/Det=3N8/N=64/	6.25
/C=1/Det=1N4/N=32/	6.9
/C=3/Det=1N16/N=64/	5.35
/C=3/Det=3N12/N=64/	5.65
/C=3/Det=1N8/N=32/	5.65
/C=3/Det=3N8/N=32/	6.5
/C=3/Det=1N4/N=32/	6.35

TABLE 4.2.1: Performance Comparisons for System 1 and System 3 Detection, for Coded 8PSK, Using Codes 1 and 3

SCHEME	APPROXIMATE AVERAGE NUMBER OF BIT ERRORS PER BURST AT GIVEN BER			
	3 × 10^{-2}	7 × 10^{-3}	1 × 10^{-3}	5 × 10^{-4}
/C=1/Det=1N8/N=32/	35	-	40	25
/C=1/Det=3N8/N=64	50	50	60	75
/C=1/Det=1N4/N=32/	140	-	120	300
/C=3/Det=1N16/N=64/	25	22	14	-
/C=3/Det=3N12/N=64/	27	21	17	14
/C=3/Det=1N8/N=32/	30	23	18	-
/C=3/Det=3N8/N=64/	40	32	30	25
/C=3/Det=1N4/N=32/	60	60	52	-

TABLE 4.2.2: Error Burst Characteristics for System 3 Detection of Coded 8PSK, Using Codes 1 and 3

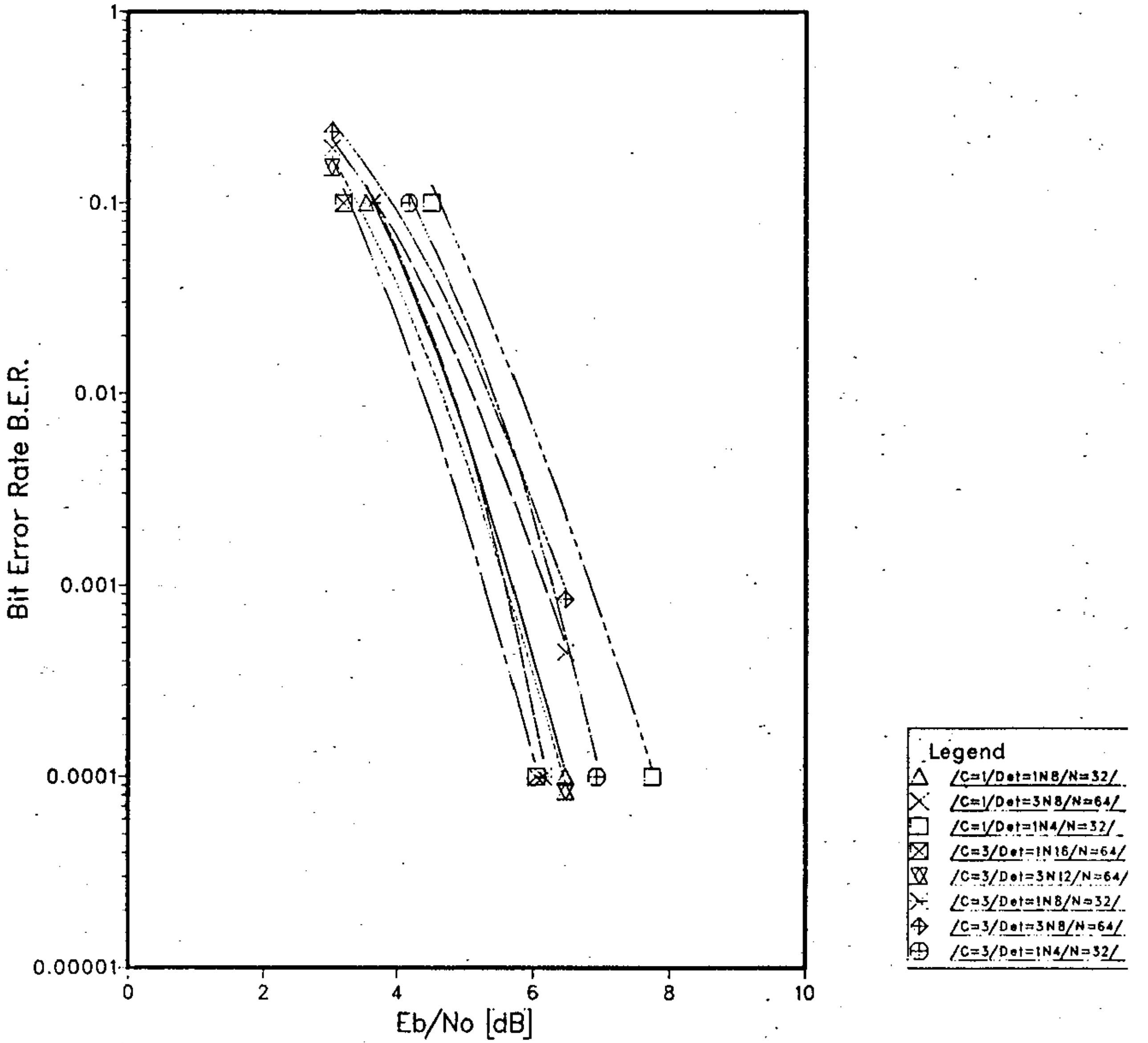


Constellation A

Constellation B

Figure 4.2.1 Subsets of The 8PSK Constellation, to One of Which All The (p_i) of The 4 Expanded Vectors of a Given Vector, Must Belong

Graph 4.2.1 Near Maximum Likelihood System 3 Detection



COMMON ATTRIBUTES
/M=8/

4.3 PSEUDOBINARY DETECTORS

Pseudobinary techniques have been studied in conjunction with Viterbi[70] and near-maximum likelihood[69,71] detection schemes. In this instance pseudobinary techniques are applied to both Viterbi Algorithm and near-maximum likelihood System 1 detectors for coded 8PSK. (See Section 2.5 for a description of the model.) The pseudobinary technique simplifies the detection process by allowing only two expanded vectors, (Section 4.1), to be derived from any one stored vector. These two expanded vectors are those with the smallest costs and are usually determined without the need to actually calculate their costs.[70] In the case of System 1 detection, k_1 vectors are still stored, so the reduction in complexity is simply a direct result of halving the total number of expanded vectors considered in each symbol interval. The possible savings for the Viterbi detector are more considerable, since the pseudobinary technique involves, in effect, a redefinition of the meaning of a state, (see Section 3.2). This redefinition leads to a reduction in the number of stored vectors. Initially the basic idea for the Viterbi detector will be outlined, followed by the differences for System 1 detection. Finally, the use of a redefined mapping for the $\{p_i\}$ is described, which simplifies the determination of the two expanded vectors of a given vector, with the smallest costs.

The initial selection process at time t=iT, (upon the receipt of sample r_i), determines the two expanded vectors of each stored vector Q'_{i-1}, with the lowest costs. This involves the selection of two values of q'_i, for each vector Q'_{i-1}. In effect the detector recodes each symbol q'_i into a binary symbol. One such symbol is an element in the expanded vector derived from Q'_{i-1} with the lowest cost, (recorded value 0). The other is an element in the expanded vector derived from Q'_{i-1} with the second lowest cost, (recorded value 1). The two chosen values of q'_i for one particular vector Q'_{i-1} are not usually the same as those of another vector. The recording does not change the value of the $\{q'_i\}$ involved, nor does it involve the storage of any other value, but rather it implies that the locations for the temporary storage of the expanded vectors before selection, are governed by the recoded binary values of a number of their most recent elements, $\{q'_h\}$.

The particular arrangement for the pseudobinary Viterbi detector will now be described. Since the algorithm is very similar to the Viterbi Algorithm, only those points which differ from the procedure of Section 3.2 will be described. As noted above, the pseudobinary process involves the recording of four-level symbols q'_h into binary symbols. In the case of the pseudobinary Viterbi detector, this also implies a recoding of the meaning of a state. The definition of a state was previously a combination of the values of a number of the most recent symbols held in a stored vector. The new definition also involves the same most recent symbols, but in this case the definition uses the recoded values, (0 and 1), of these symbols. For a constraint length-k code, the number of states reduces from 4^{k-1},(Section 3.2), to 2^{k-1}, when the new definition of a state is used. Clearly, this is not a true Finite-State machine representation.[72] In particular the newly-defined state at time t=iT, and the recoded value of q'_i, do not define the code symbol c'_i.[72] (See Appendix A4 for a full description of the Finite-

State Machine for coded 8PSK. The algorithm which processes the 2^{k-1} vectors $\{Q'_{i-1}\}$ on the receipt of the sample r_i is much the same as that for the true Viterbi detector, (see Section 3.2). The selected vectors $\{Q'_i\}$ are those with the lowest costs, for each combination of the recoded values of $q'_{i-k+2}, q'_{i-k+3}, \ldots, q'_i$, in the 2^k expanded vectors of the $\{Q'_{i-1}\}$.

For the System 1 detectors, the process is exactly as in Section 4.1, except that the selection process now involves only $2k_1$ expanded vectors instead of $4k_1$ expanded vectors. As described above, there are just two expanded vectors derived from each vector Q'_{i-1}.

In both the above detectors, the two chosen expanded vectors derived from a vector Q'_{i-1}, of the four possible expanded vectors, are chosen in the following way. The incremental costs $\{|w'_i|^2\}$ for the four expanded vectors are determined, (see Sections 3.2 and 4.1), and ranked. The two expanded vectors with the lowest values of $|w'_i|^2$ are chosen. Clearly, contrary to the assumption stated at the beginning of this section, this process does involve the determination of costs. In order to be able to perform this selection process without recourse to determining and ranking incremental costs, a slight ammendment is required to the mapping function (Figure 2.5.4). Figure 4.3.1 gives the ammended mapping function of the $\{c_i\}$ onto the $\{p_i\}$. The mapping is now such that the ratio of the real and imaginary components of every possible value of p_i is either 2:1 or 1:2. For example, for the value of p_i mapped from $c_i = 0$, the ratio of the real and imaginary components of p_i is 2:1. When a vector Q'_{i-1} is expanded, the four possible values of p'_i are fixed by the original vector. They are all either of set A, or all of set B in Figure 4.2.1. If the detector knows which of these two sets is involved for vector Q'_{i-1}, the ammendment to the mapping function allows the use of simple threshold tests to determine the two values of p'_i closest to the received sample r_i. Figure 4.3.2 shows the thresholds which are used when the possible values of p'_i are in set A. The four thresholds are the lines $\mathrm{Re}(p_i)=\frac{1}{2}\mathrm{Im}(p_i)$, $\mathrm{Re}(p_i)=-2\mathrm{Im}(p_i)$, $\mathrm{Re}(p_i)=2\mathrm{Im}(p_i)$, and $\mathrm{Re}(p_i)=-\frac{1}{2}\mathrm{Im}(p_i)$, in the complex number plane. The first two thresholds are used to determine the value of p_i nearest to r_i. The second nearest value of p_i to r_i is determined by using the threshold which passes through the value of p_i which is nearest to r_i. (The tests for set B are similar, except that the last two thresholds defined above are used to determine the value of p_i nearest to r_i.) For the original mapping function, the thresholds would have been the lines $\mathrm{Re}(p_i)=\tan(22.5°).\mathrm{Im}(p_i)$, $\tan(22.5°).\mathrm{Re}(p_i)=-\mathrm{Im}(p_i)$, $\tan(22.5°).\mathrm{Re}(p_i)=\mathrm{Im}(p_i)$, and $\mathrm{Re}(p_i)=-\tan(22.5°).\mathrm{Im}(p_i)$, in the complex number plane. Since tan(22.5°) is irrational, the tests using these thresholds are much more complex to implement. In fact, it is easier to calculate the costs of the expanded vectors in this case. (The amended mapping function was only used in one set of simulation tests, using System 1 detection.) The amendment effectively rotates the two sets of points A and B in the complex number plane, both with respect to the axes and with respect to each other. In the limit, a rotation of one of the sets of points with respect to the other will eventually cause the sets of points to coincide. When this happens, the scheme's tolerance to noise is that of uncoded QPSK. Therefore, the amendment to the mapping function must affect the tolerance to noise.

Graphs 4.3.1 to 4.3.11 give the results for the pseudobinary detectors. These are graphs of bit error rate (BER) as the signal to noise ratio, E_b/N_O, is varied. E_b is the average energy per transmitted data bit. $N_O/2$ is the two-sided power spectral density of the additive white Gaussian noise. (See Appendix A5 for more details of the simulation techniques. Appendix A8 gives the notation used to describe the many variants of these detection schemes, which were tested by computer simulation.) For Graphs 4.3.1 to 4.3.5, the accuracy of the results in the range of BER, 1 in 10^3 to 1 in 10^4, is of the order of ±0.25dB.

Graph 4.3.1 compares the performance of pseudobinary Viterbi detection, (8 stored vectors), with Viterbi detection for coded 8PSK, (Section 3.2), and threshold detection for QPSK, (Section 3.1). Code 3 is used in the coded systems. Clearly the degradation in tolerance to noise for pseudobinary Viterbi detection is substantial, compared with Viterbi detection. This degradation is 3.7dB at a BER of 1 in 10^3. Table 4.3.1 gives the error burst characteristics in comparison with Viterbi detection. Appendix A5 defines an error burst. Clearly the severe degradation of Graph 4.3.1 is linked with very long error bursts. An analysis of pseudobinary Viterbi detection for coded 8PSK has been undertaken, related specifically to Code 1, (with four stored vectors since $2^{k-1}=4$). (No curves for Code 1 are presented here, but computer simulation tests have produced results in broad agreement with those for Code 3.) The analysis was undertaken under near-noiseless and typical noise level conditions. Figures 4.3.3 and 4.3.4 illustrate typical code trellis diagrams for the detector, for the near-noiseless (E_b/N_O=37dB), and typical noise level (E_b/N_O=6dB), conditions. These diagrams are essentially graphs of the state of a vector (vertical axis), as it varies with time in symbol intervals (horizontal axis). The state is that of the original definition (Section 3.2), not the redefined states of this section. It is given by the combination of the values of the vector elements $q'_{i-k+1}, q'_{i-k+2}, \ldots, q'_{i-1}$ at time t=iT. An integer value is given to each possible state, as described in Section 2.5. The code trellis diagram gives the states for each of the stored vectors over a period of time up to the current time t=iT. Each line in the diagram is for one of the stored vectors. More details are given in Appendix A4. In Figure 4.3.5, a section of the code trellis diagram is given where the redefined values of the states are used. (This will be used, in conjunction with Figures 4.3.6 and 4.3.7 to explain some of the features of the code trellis diagrams of Figures 4.3.3 and 4.3.4.) Here, all possible values of the redefined states of the vectors at time t=iT are shown, rather than the actual redefined states of the vectors at time t=iT. The redefined states are given in terms of the recoded values of the two most recent symbols in the associated vectors. In Figure 4.3.5 the left-most value in the state definition is the recoded value of the oldest of the two symbols defining the state, for Code 1. The right-most value in the state definition is the recoded value of the most recent of the two symbols defining the state for Code 1.

Figure 4.3.5 can be split into smaller units called sub-trellises, (in this case two), where the vectors $\{Q_i'\}$ in a sub-trellis are derived from the vectors $\{Q_{i-1}'\}$ in the same sub-trellis. These four vectors and the values of their redefined states are part of no other sub-trellis. Appendix A4 deals with this in more detail. The two sub-trellises for the redefined states for Code 1 are given in Figure 4.3.6. These sub-trellises can be used in the explanation of the near-noiseless code trellis diagram of Figure 4.3.3. From Figure 4.3.3 it is evident that all the vectors at any point in time t=jT are most often derived from just two vectors at time t=(j-1)T. This is shown more clearly in Figure 4.3.7. Under no-noise conditions it is clear that the vector Q_{i-1}', which has redefined state (OO), is the Maximum Likelihood vector (and is in fact correct). It will therefore always yield the vectors $\{Q_i'\}$ which have the redefined states (OO) and (O1). (See sub-trellis 'A' in Figure 4.3.6.) In the case of sub-trellis 'B' it is clear that the vector Q_{i-1}' which has state (O1) is likely to have a lower cost than the vector Q_{i-1}' which has state (11) because of the definition of the recoded symbols O and 1. Therefore it is likely that the vectors $\{Q_i'\}$ which have states (1O) and (11) will be derived from the vector Q_{i-1}' which has state (O1). This is seem to be true in Figure 4.3.3. Figure 4.3.4, the typical noise level code trellis diagram, indicates that this is still largely true when the signal to noise ratio is lower. For example the four vectors at time t=iT are derived from the single vector which has the redefined state (OO) at time t=(i-4)T.

An interesting point arising from Figure 4.3.4, specifically at time t=iT, is the possibility that two differing redefined states are actually the same state as originally defined, (vectors 2 and 3). In this particular case, although both vectors 2 and 3 have the same values of q'_{i-1} and q'_{i-2}, they do not have the same values of the recoded symbols q'_{i-1} and q'_{i-2}. From Appendix A4 and the discussion in Section 4.2, when the Viterbi detector selects between possible data sequences which have the same state as originally defined, it is assured that, of the vectors it discards, one can never be the Maximum Likelihood vector. Clearly then, a system which allows vectors with the same state (as originally defined) to exist includes wasteful redundancy, in that only the vector with the lowest cost which has this given state, needs to be stored. Again, as for System 3 of Section 4.2, there is no guarantee that, when the pseudobinary detector selects an expanded vector from those expanded which have the same redefined state, the discarded vectors could not in the future if retained, have lower costs than the vector actually chosen. In other words, such a discarded vector could be the Maximum Likelihood vector. As discussed for System 1, the shorter the period of time, (i-j)T<t<iT, for which the stored vectors' element values differ, the smaller will be the cost differences between the vectors in the absence of noise. In such a case it is more likely that the algorithm could discard the correct vector in the presence of noise, than in the case where the stored vectors have different element values over a longer period of time. This point is even more important for the pseudobinary detector, because the time interval over which the vectors' element values differ is typically very much shorter than in the System 1 detector.

This point also affects the ability of the detector to recover, once an error has been made. The tendency, even under noisy conditions, that all vectors are derived from one vector only a very few symbol intervals in the past, means that once an error has occurred such that the lowest-cost vector, (whose state is (OO)), is wrong, the system will often discard the correct vector very rapidly. Once the correct vector is discarded, since the lowest-cost vector is very often derived from the previous lowest cost vector, (where both have redefined state value (OO)), it may be a long time before the lowest-cost vector's element values are the same as those of the data sequence at the transmitter.

State redefinition has produced a system which is very inferior. This system, which redefines the states on the basis of the relative costs of the expanded vectors over (k-1) symbol intervals, yields a set of 2^{k-1} stored vectors which is very often not the set of most likely transmitted sequences. The extremely poor performance of this detector may indicate that the detector is effectively storing only one truely unique vector. (In Section 5.1 a detector is described which does store only one vector. It will be seen that its performance is not very inferior to that of this detector.) Therefore, an important point is the fact that anti-merging, (which is incorporated in System 1 detection), is not incorporated in this detector. The original pseudobinary Viterbi detector was proposed for linear channels with intersymbol interference.[70] Anti-merging was not incorporated because there was no tendency for the vectors to merge, (become the same). Studies have shown[28] that there is a greater tendency for vectors to merge when coded signals are used. Anti-merging was not incorporated into this detector because analyses of the stored vectors during a number of simulation tests showed that all the stored vectors were different, (although they differed over only a few symbol intervals). (The systems described in Section 5.3 are attempts at redefining the meaning of a state, based on the actual values of the symbols q_i'.)

Graph 4.3.2 gives the results for pseudobinary System 1 detection, compared with System 1 detection, for a scheme using Code 3. The original mapping function was used, so that the two chosen expanded vectors were determined using their costs. It is apparent, except for the schemes where k_1=8, that the performances of the two techniques are very similar. (As noted in Section 4.1, the relatively good performance of System 1 detection with k_1=8 may be due to the accuracy quoted.) At a BER of 1 in 10^4, the degradations in tolerance to noise compared with System 1 detection are <0.05dB, 0.4dB and 0.15dB, for k_1=16,8, and 4 respectively. The error burst characteristics are outlined in Table 4.3.2. Clearly, there is very little difference between the schemes' error burst characteristics. A fair comparison of the schemes must include some idea of the schemes' relative complexities. Leaving aside the method by which the two best expanded vectors are determined, (see earlier), the pseudobinary scheme, for a given number of stored vectors k_1, at best halves the processing time compared with the System 1 scheme. Conversely, for the same available processing time per symbol interval, the pseudobinary scheme can at best deal with twice the number of stored vectors of the System 1 scheme. It must be stressed that this is an 'upper-bound'. In practice the advantage will be less, because of the required processing which is identical in both detectors, and because of the time spent determining the $2k_1$ expansions to be processed. In the light of the above, System 1 detection with $k_1=8$ should be compared with pseudobinary System 1 detection with $k_1=16$, and System 1 detection with $k_1=4$ should be compared with pseudobinary System 1 detection with $k_1=8$. Particularly in the case of the latter comparison, the pseudobinary scheme provides a reasonable improvement in tolerance to noise (0.4dB at a BER of 1 in 10^4).

Graph 4.3.3 contrasts the results of pseudobinary System 1 detection using the redefined mapping function of Figure 4.3.1, with pseudobinary System 1 detection using the original mapping function of Figure 2.5.4. Clearly, the degradation sustained by using the redefined mapping function is negligible. At a BER of 1 in 10^4, the degradations in tolerance to noise, compared with the use of the original mapping function, are <0.05dB, 0.15dB and 0.1dB, for $k_1=16,8$ and 4, respectively. The error burst characteristics are presented in Table 4.3.3. It is apparent that overall the average number of bit errors per burst is increased for the scheme using the ammended mapping function.

For all the remaining tests presented here, the original mapping function was used. Graph 4.3.4 presents the results where the sub-optimal phase distance measure is used. The phase distance is simply the difference in the phase angles of the received sample r_i and a possible received sample in the absence of noise. This measure involves no squaring operations and is therefore simpler to implement than the unitary distance measure. Appendix A7 gives further details.

Clearly, as in Sections 3.2 and 4.1, the degradation in tolerance to noise compared with System 1 detection using the unitary distance measure, is substantial. At a BER of 1 in 10^4 the degradations are 0.5dB, 0.35dB, and 0.4dB, for k_1=16,8 and 4, respectively. Table 4.3.4 gives the error burst characteristics. Clearly, there is very little difference in the error burst characteristics, when the phase distance measure is substituted for the unitary distance measure.

Graph 4.3.5 gives the results for a scheme which is basically a cross between the Viterbi detector and the pseudobinary System 1 detector. It is called two-symbol expansion Viterbi-type detection. The scheme uses the original definition of a state, but only allows two expanded vectors per vector Q'_{i-1}. In this way the total number of expanded vectors is reduced from 4^k to 2.4^{k-1}, the reduction being therefore a factor of two. Otherwise the detector is exactly as in Section 3.2. This upper-bound complexity advantage may not be achievable in practice since it depends very heavily on the implementation. In the case simulated, expanded vectors were chosen by ranking their incremental costs, as for the pseudobinary System 1 scheme using the original mapping function of Figure 2.5.4. The results of Graph 4.3.5 are for Codes 1 to 4, the comparison being with Viterbi detection. Table 4.3.5 outlines the degradations in tolerance to noise compared with the corresponding Viterbi detectors, at a BER of 3 in 10^4. Clearly, the degradations are severe, especially in the light of the low achievable complexity reductions. The average numbers of bit errors per burst are very similar to those for Viterbi detection, (see Table 3.2.1).

The degradations in tolerance to noise for this detector are much more severe than those for System 1 detection, although in both cases, the same expanded vectors are discarded if $k_1=4^{k-1}$. For the two-symbol expansion Viterbi-type detector, it can clearly no longer be guaranteed that the Maximum Likelihood vector is among the stored vectors. The System 1 detector ranks the costs of all the non-discarded expanded vectors to select each stored vector. The two-symbol expansion Viterbi-type detector ranks the costs of disjoint subsets of the non-discarded expanded vectors, (see Chapter 3.2). In the latter case it is more likely that a relatively low-cost expanded vector, which is not discarded by the pseudobinary algorithm, will be discarded by the selection process which follows.

Graphs 4.3.6 to 4.3.8 show the effect, (for k_1=16,8 and 4, respectively), of varying the delay in detection, N, in symbol intervals for pseudobinary System 1 detection, at a value of E_b/N_0 for which the BER is approximately 1 in 10^3 for large N. (Code 3 is used.) These should be compared with Graphs 4.1.9 to 4.1.11 for System 1 detection. It is clear that the results for the pseudobinary version are very similar to those for System 1, (see Section 4.1, and in particular Table 4.1.11).

Graphs 4.3.9 to 4.3.11 show the results of introducing constant phase offsets, (constant phase errors in the receiver estimate of carrier phase), for pseudobinary System 1 detection. (Code 3 is used.) These are to be compared with Graphs 4.1.12 to 4.1.14 for System 1 detection. Table 4.3.6 gives the phase offsets for which the BER is both 10 times and 100 times the BER when there is no phase offset, for both System 1 detection (repeated from Table 4.1.12), and pseudobinary System 1 detection. Although comparison between the pseudobinary and original versions of System 1 detection is difficult, since the BERs when there is no offset tend to be somewhat different, it can be concluded that the pseudobinary system suffers less when constant phase offsets occur. This is particularly so for $k_1=4$, since the BERs where there is no offset in Figures 4.1.14 and 4.3.11 are very similar (6 in 10^4 and 5 in 10^4 respectively).

SCHEME	APPROXIMATE AVERAGE NUMBER OF BIT ERRORS PER BURST, AT GIVEN BER		
	3×10^{-2}	7×10^{-3}	1×10^{-3}
/Det=V64/N=80	20	13	11
/Det=V8/PB=Pb/N=80/	120	93	116

TABLE 4.3.1: Error Burst Characteristics for Pseudobinary Viterbi Detection for Coded 8PSK, using Code 3

SCHEME	APPROXIMATE AVERAGE NUMBER OF BIT ERRORS PER BURST, AT GIVEN BER	
	3×10^{-2}	1×10^{-3}
/Det=1N16/N=64/	25	14
/Det=1N16/PB=Pb/N=64/	25	11
/Det=1N8/N=32/	30	18
/Det=1N8/PB=Pb/N=32/	31	17
/Det=1N4/N=32/	55	27
/Det=1N4/PB=Pb/N=32/	53	32

TABLE 4.3.2: Error Burst Characteristics for Pseudobinary System 1 Detection for Coded 8PSK, using Code 3

SCHEME	APPROXIMATE AVERAGE NUMBER OF BIT ERRORS PER BURST, AT GIVEN BER	
	3×10^{-2}	1×10^{-3}
/Det=1N16/PB=Pb/	25	11
/Det=1N16/PB=Pbr/	24	20
/Det=1N8/PB=Pb/	31	17
/Det=1N8/PB=Pbr/	46	20
/Det=1N4/PB=Pb/	53	32
/Det=1N4/PB=Pbr/	65	50

TABLE 4.3.3: Error Burst Characteristics for Reduced Complexity Pseudobinary System 1 Detection for Coded 8PSK, using Code 3

SCHEME	APPROXIMATE AVERAGE NUMBER OF BIT ERRORS PER BURST, AT GIVEN BER	
	3×10^{-2}	1×10^{-3}
/Det=1N16/Dis=E/	25	11
/Det=1N16/Dis=P/	24	12
/Det=1N8/Dis=E/	31	17
/Det=1N8/Dis=P/	28	15
/Det=1N4/Dis=E/	53	32
/Det=1N4/Dis=P/	51	35

TABLE 4.3.4: Error Burst Characteristics for Pseudobinary System 1 Detection for Coded 8PSK, using Code 3, when the Phase Distance Measure is Used.

SCHEME	DEGRADATION IN TOLERANCE TO NOISE IN COMPARISON WITH THE CORRESPONDING VITERBI DETECTOR AT BER = 3×10^{-4} (dB)
/C=1/Det=V16/PB=2E/	0.4
/C=2/Det=V64/PB=2E/	0.9
/C=3/Det=V64/PB=2E/	0.65
/C=4/Det=V64/PB=2E/	0.8

TABLE 4.3.5: Performance of Two-Symbol Expansion Viterbi-type Detection for Coded 8PSK

SCHEME	PHASE OFFSET FOR WHICH BER IS 10 × BER FOR NO PHASE OFFSET (degrees)	PHASE OFFSET FOR WHICH BER IS 100 × BER FOR NO PHASE OFFSET (degrees)
/Det=1N16/PB=0/	8.5	12.5
/Det=1N16/PB=Pb/	10	16
/Det=1N8/PB=0/	6.5	11.5
/Det=1N8/PB=Pb/	9	14
/Det=1N4/PB=0/	7	12.5
/Det=1N4/PB=Pb/	12	>18

TABLE 4.3.6: Measures of the Effect of Constant Phase Offsets on Pseudobinary System 1 Detection for Coded 8PSK, Using Code 3

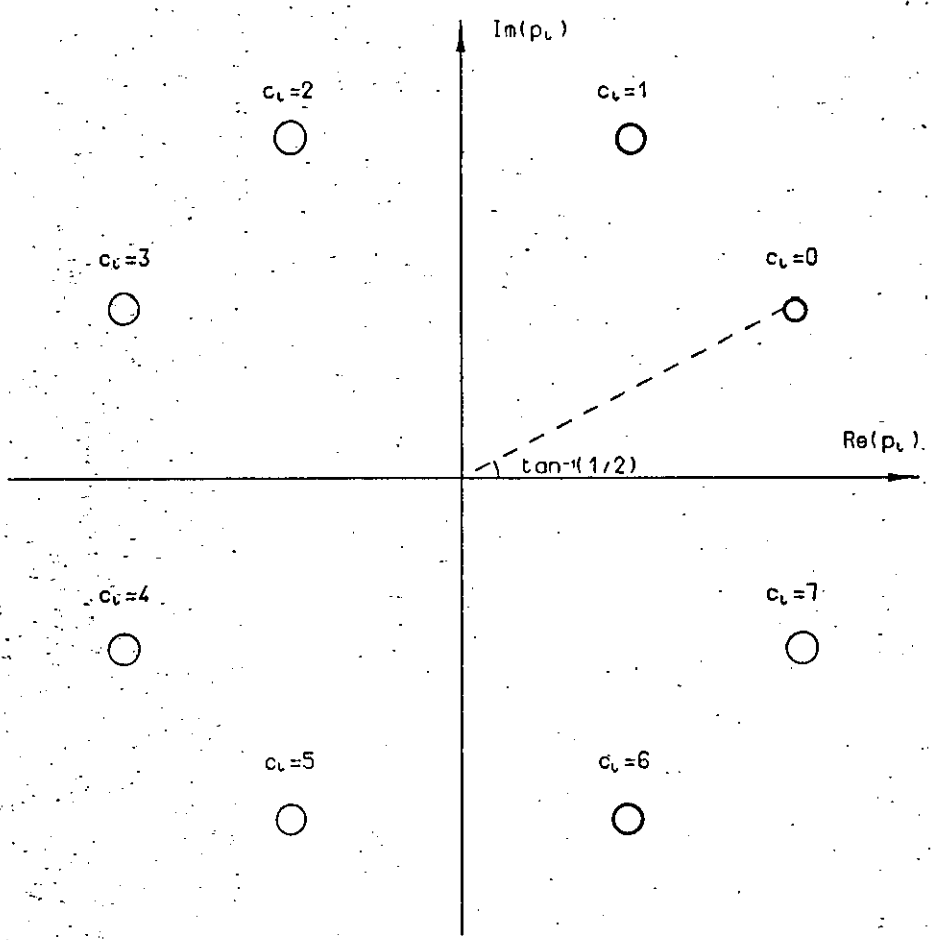


Figure 4.3.1 Redefined Mapping Function for The Reduced-Complexity Pseudobinary Detectors

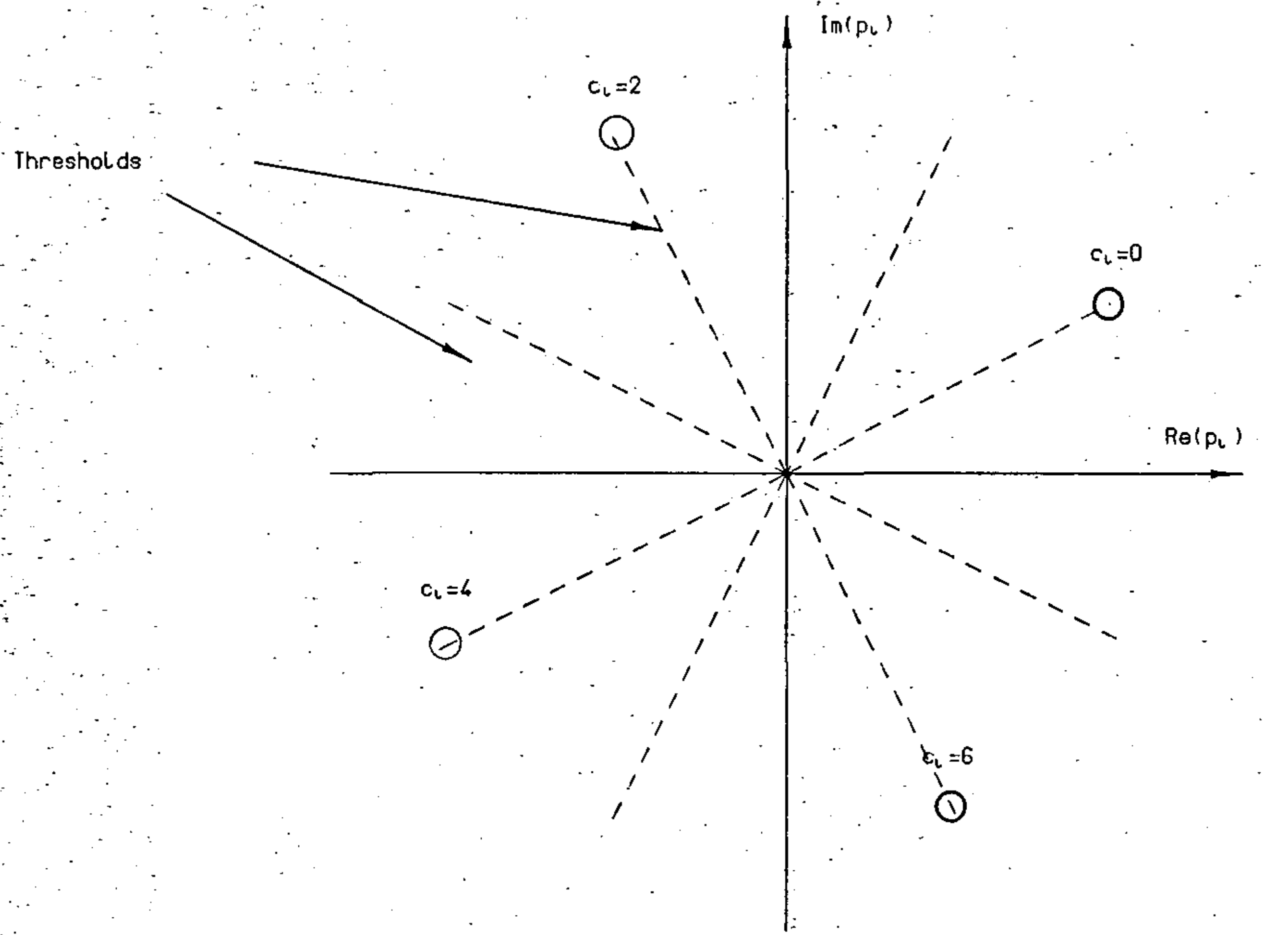


Figure 4.3.2 Thresholds in The Complex Number Plane when The Redefined Mapping Function is used

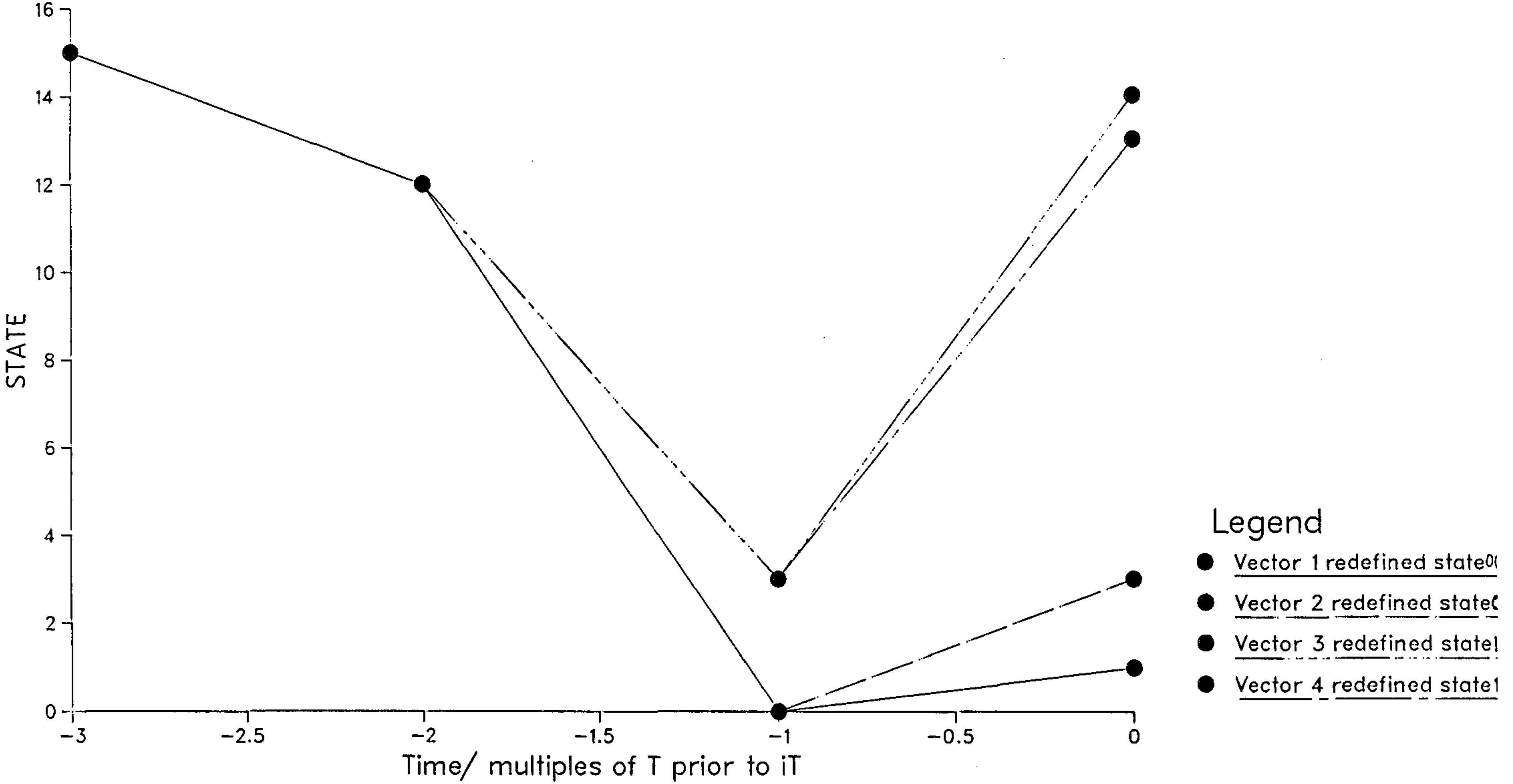


Figure 4.3.3 Code Trellis Diagram for a Scheme using Code 1
Pseudobinary Viterbi Detection. 4 Stored Vectors. Eb/No=37dB

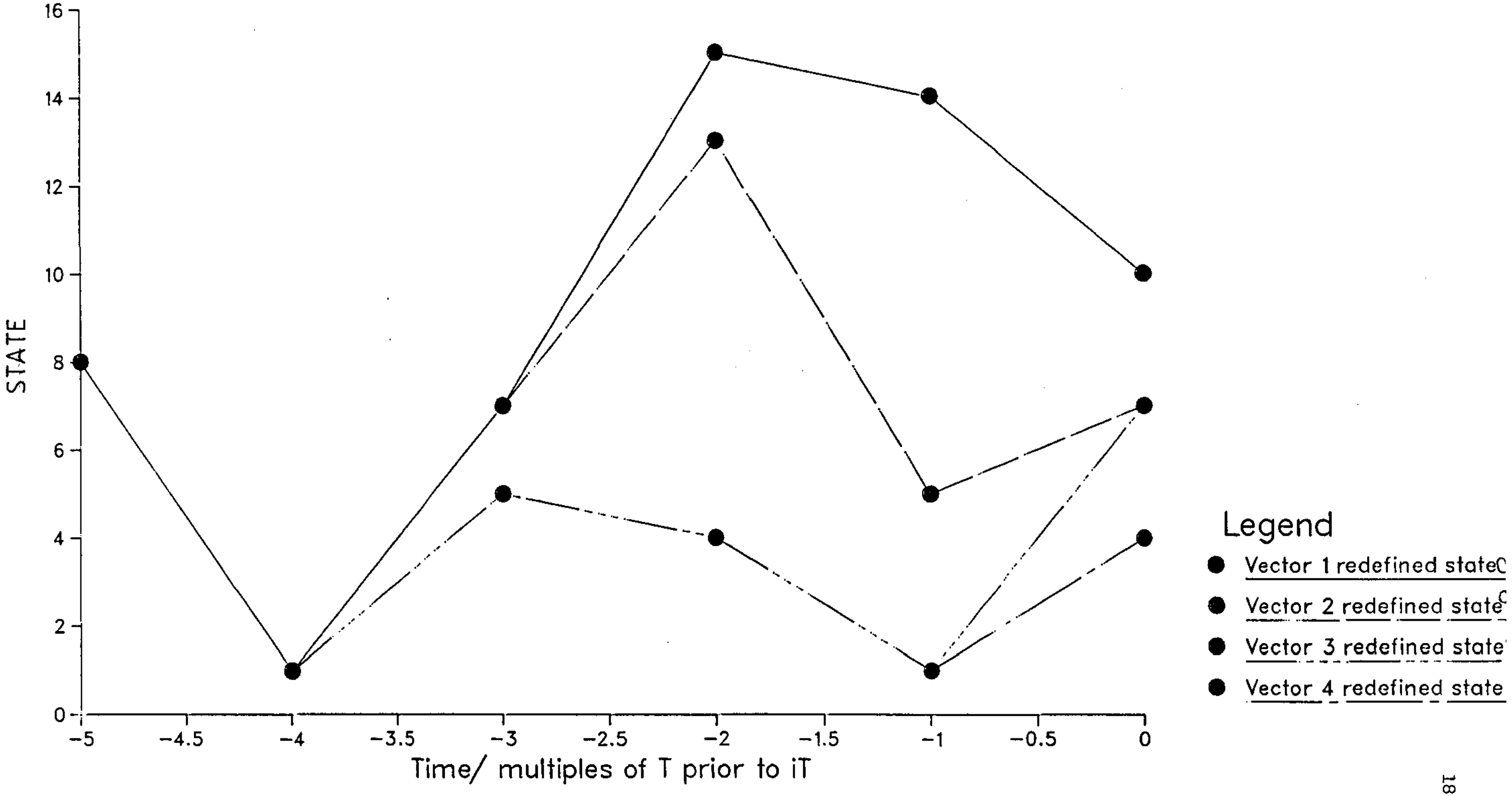


Figure 4.3.4 Code Trellis Diagram for a Scheme using Code 1
Pseudobinary Viterbi Detection. 4 Stored Vectors. Eb/No=6dB

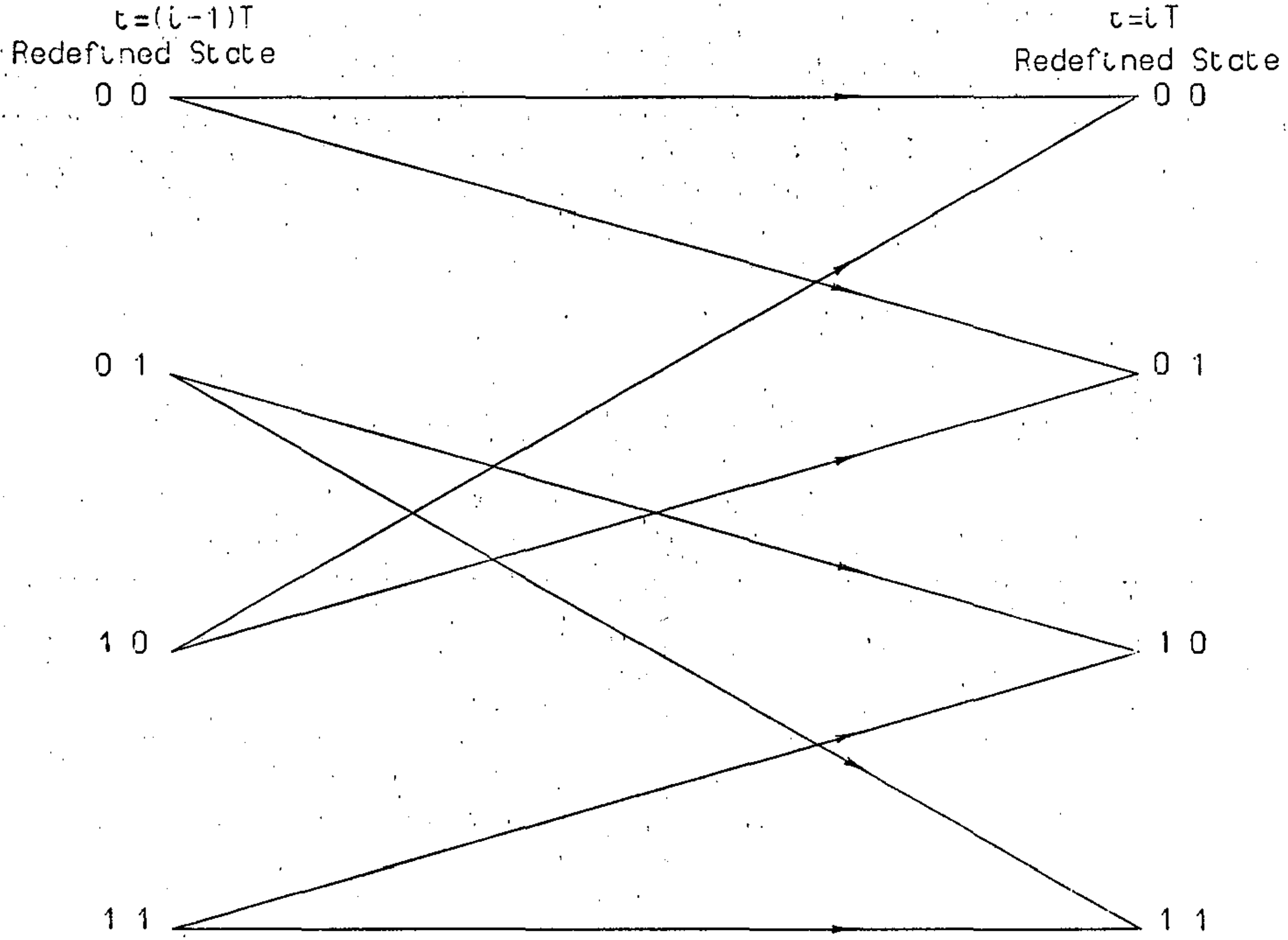
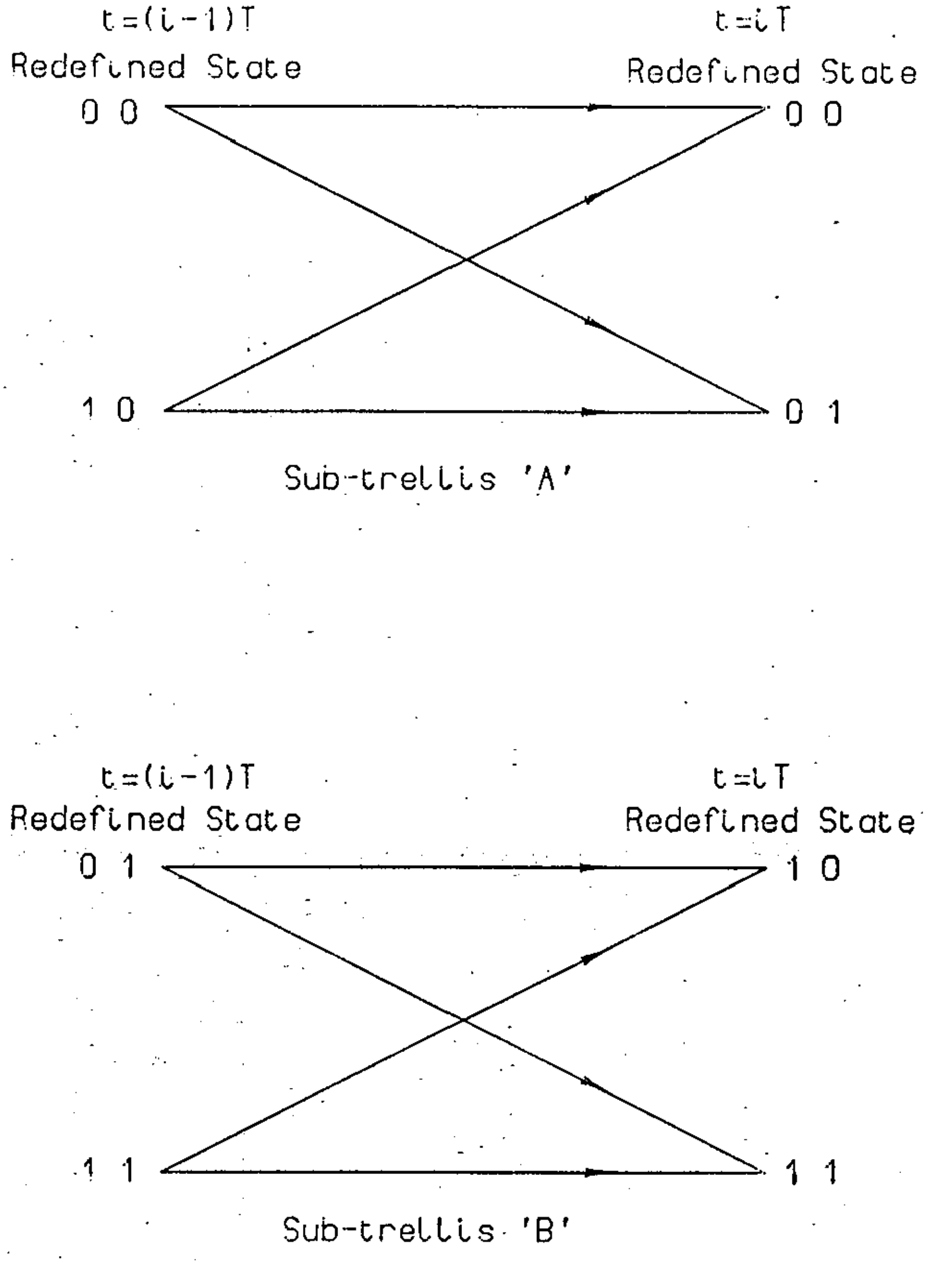


FIGURE 4.3.5 Code Trellis Diagram for The Pseudobinary Viterbi Detector (Code 1) using The Redefined States



Sub-trellis 'A'

Sub-trellis 'B'

Figure 4.3.6 Code Trellis Diagram for The Pseudobinary Viterbi Detector Split into Two Sub-trellises over One Symbol Interval

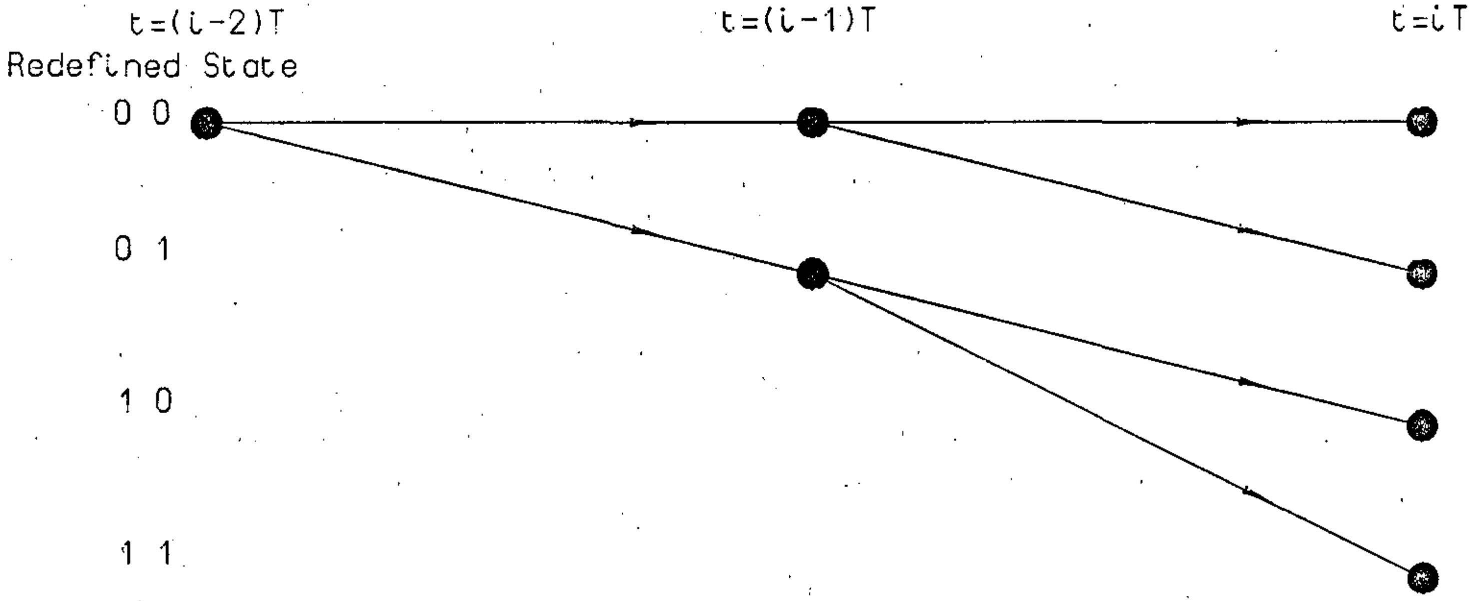
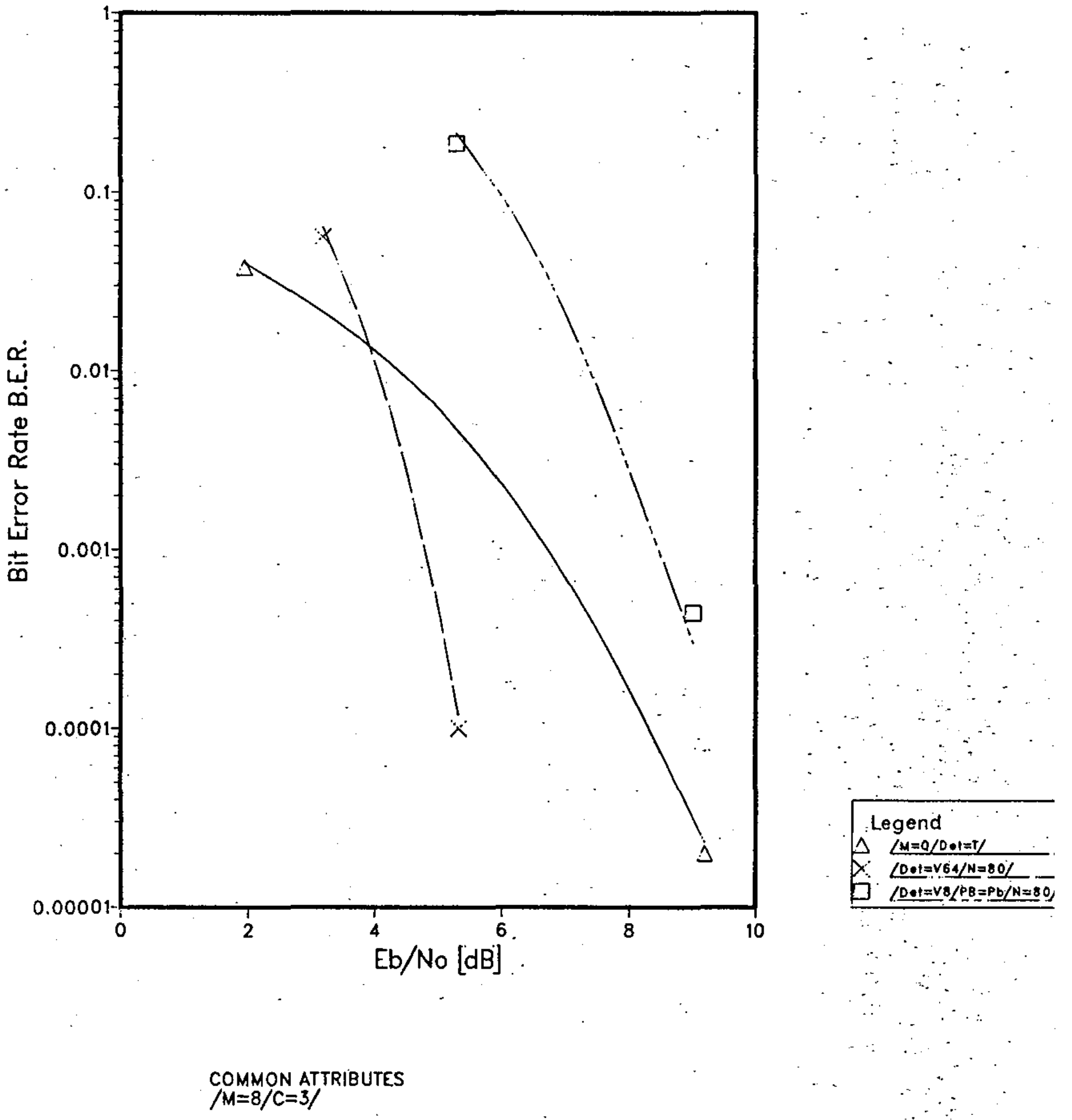
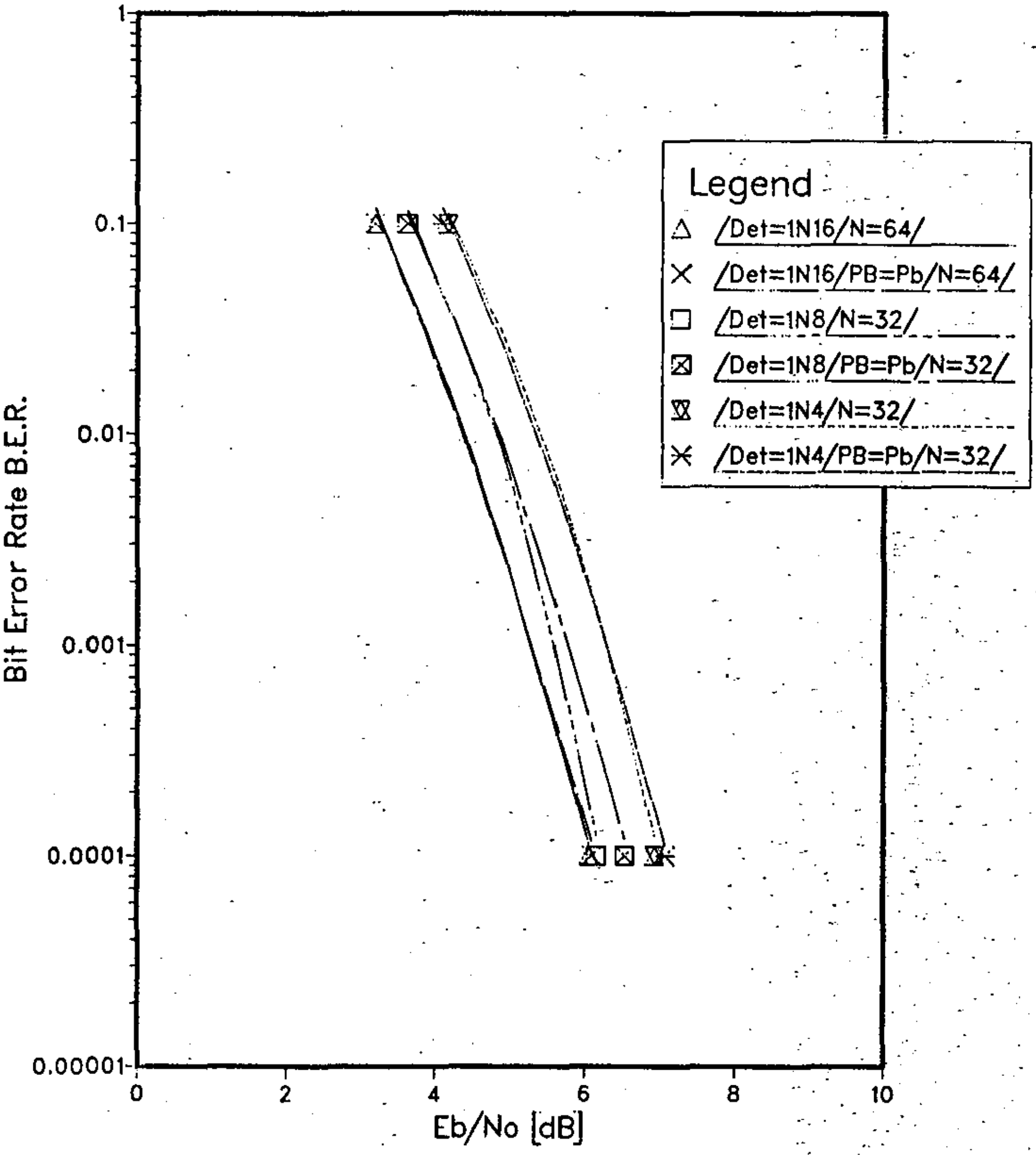


Figure 4.3.7 Part of a Typical Code Trellis Diagram for Pseudobinary Viterbi Detection at High Signal to Noise Ratios

Graph 4.3.1 Pseudobinary Viterbi Detection

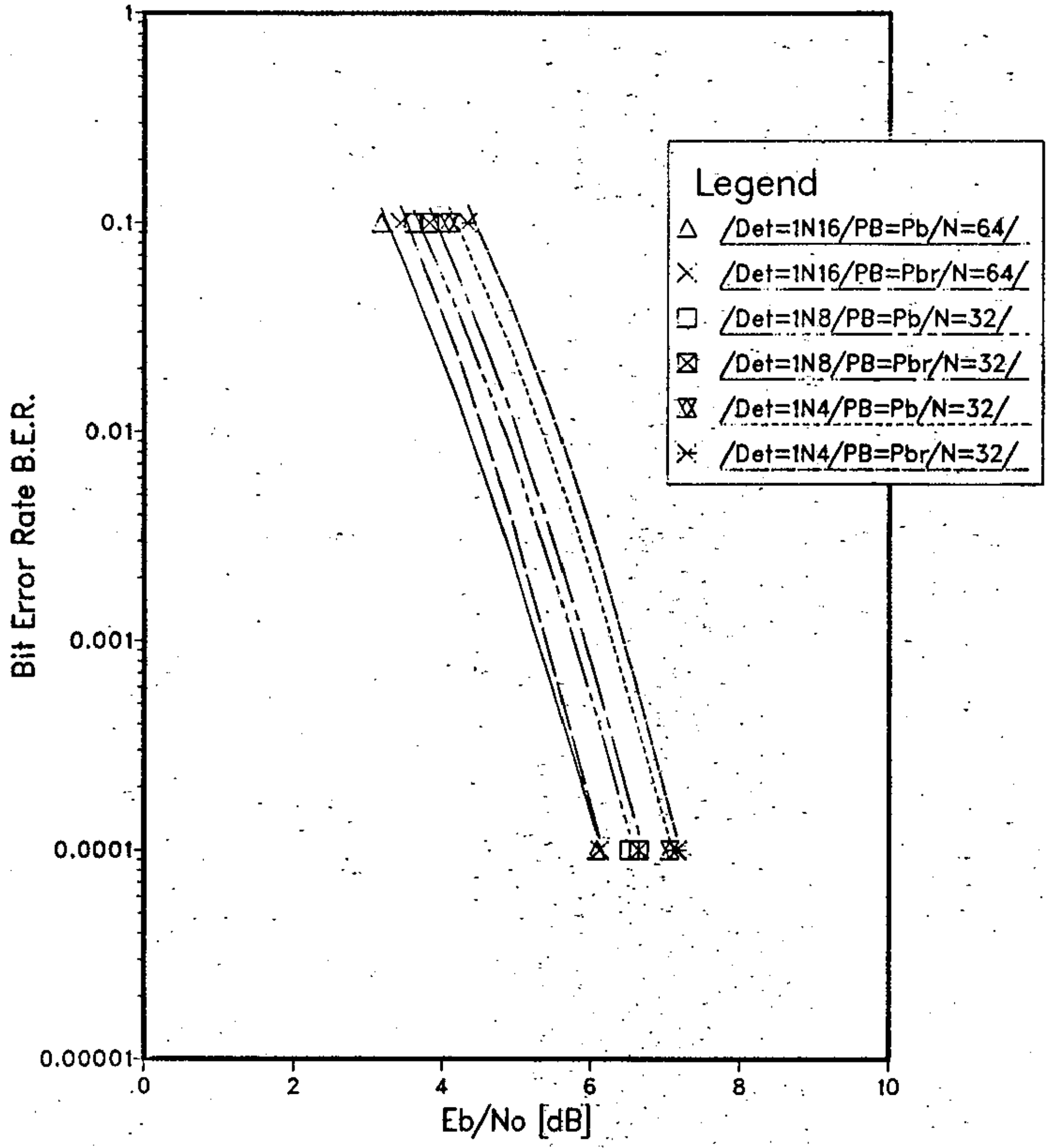


Graph 4.3.2 Pseudobinary System 1 Detection



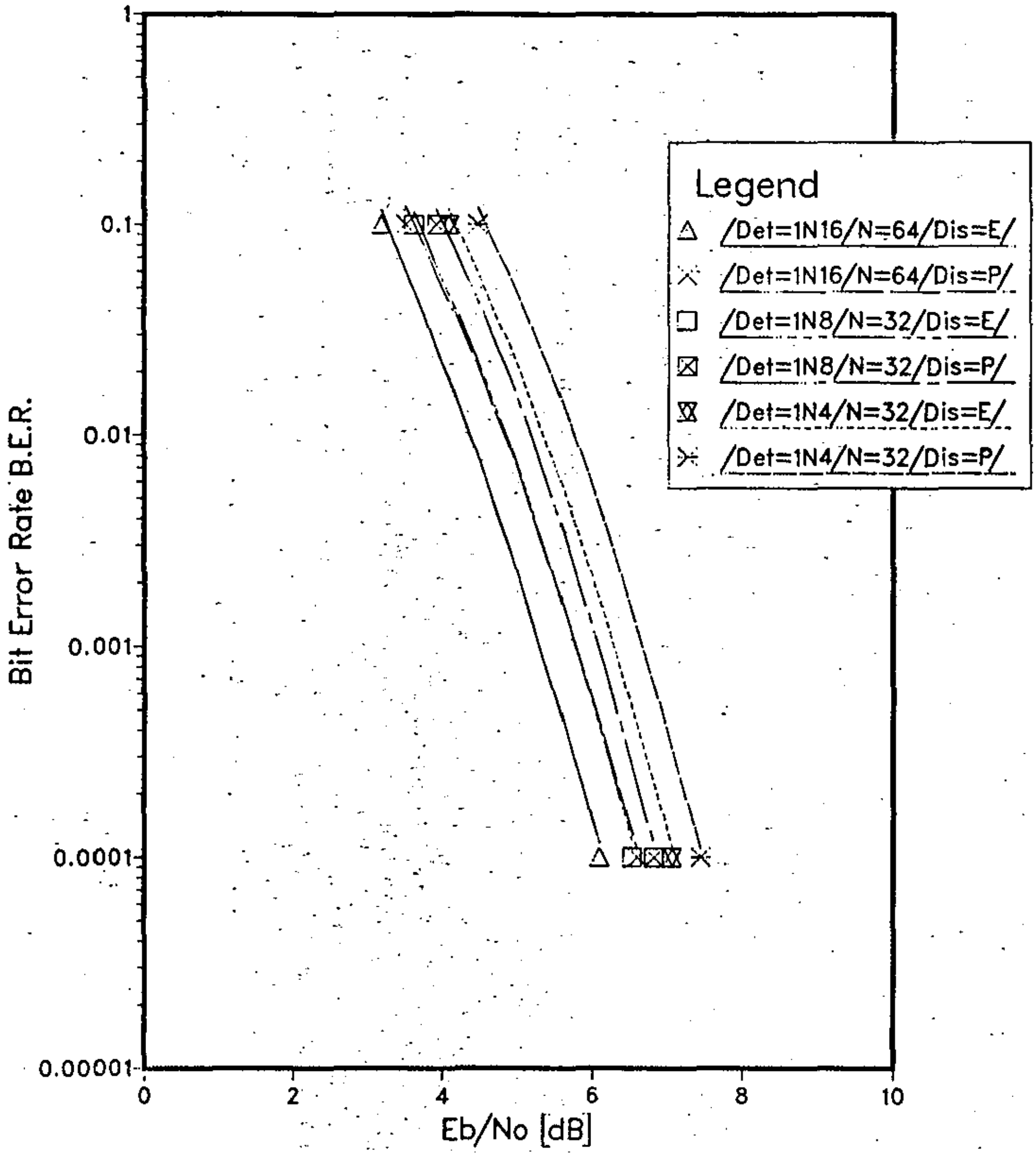
COMMON ATTRIBUTES
/M=8/C=3/

Graph 4.3.3 Reduced Complexity Pseudobinary System 1 Detection



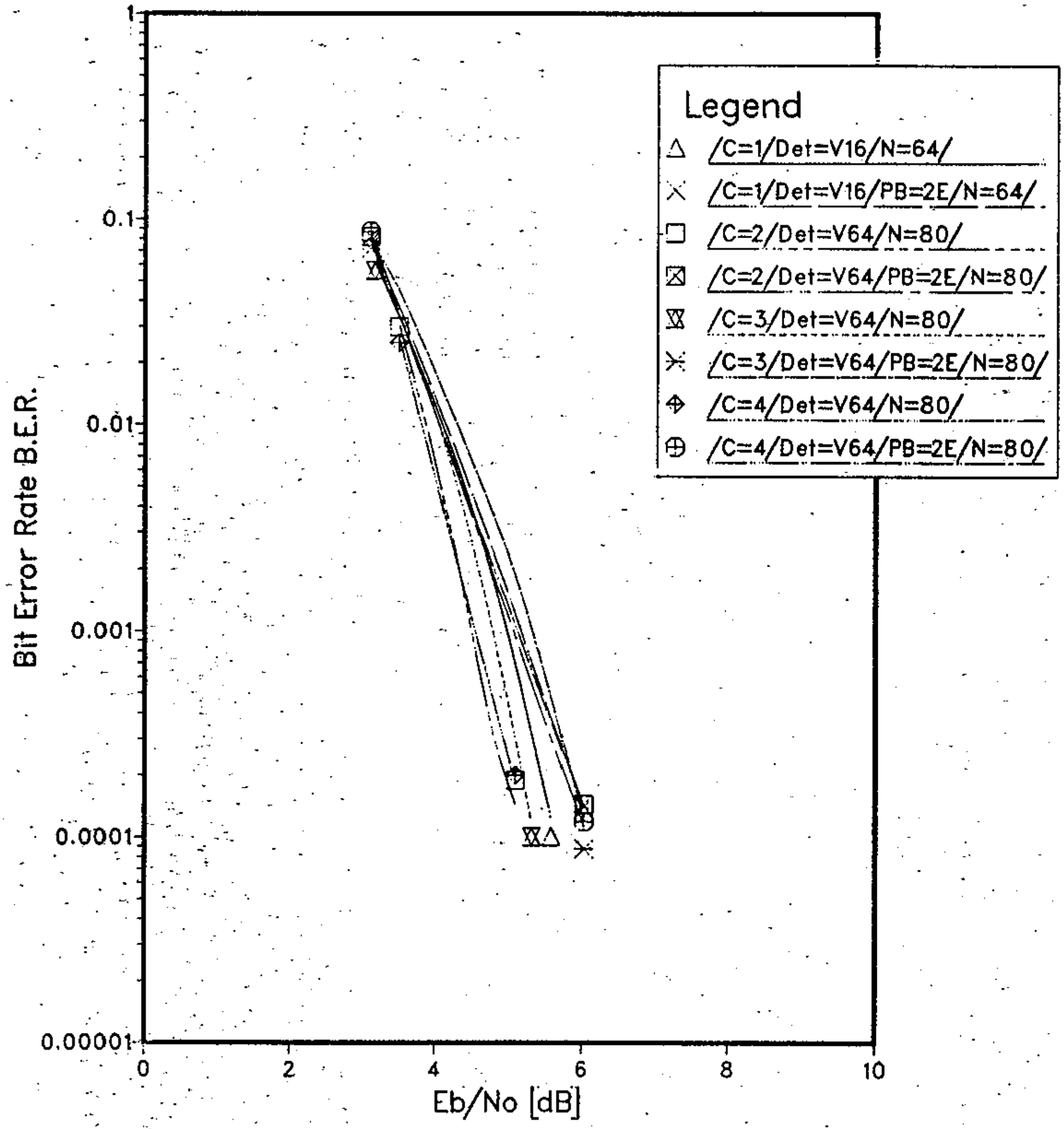
COMMON ATTRIBUTES
/M=8/C=3/

Graph 4.3.4 Pseudobinary System 1 Detection. Phase Distance Measure



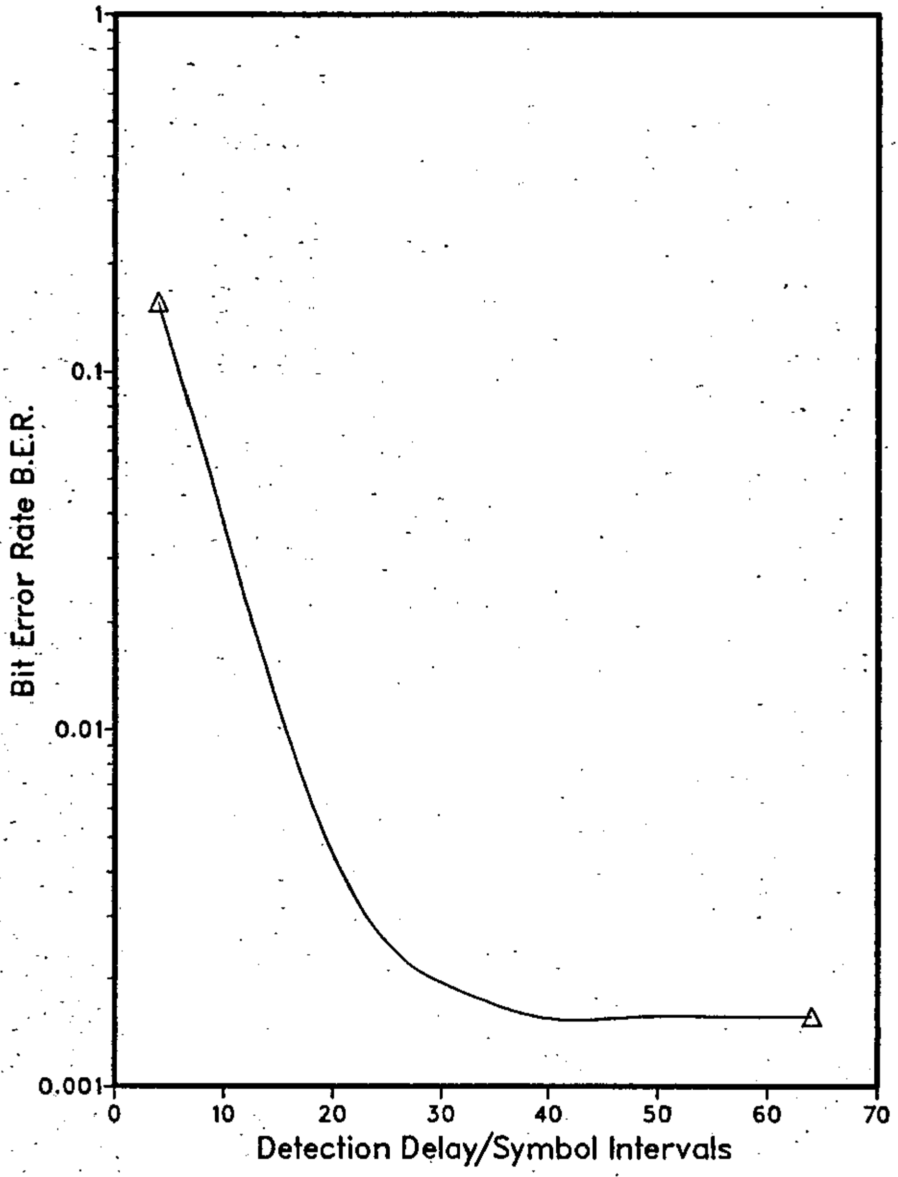
COMMON ATTRIBUTES
/M=8/C=3/PB=Pb/

Graph 4.3.5 Two Symbol Expansion Viterbi–type Detection



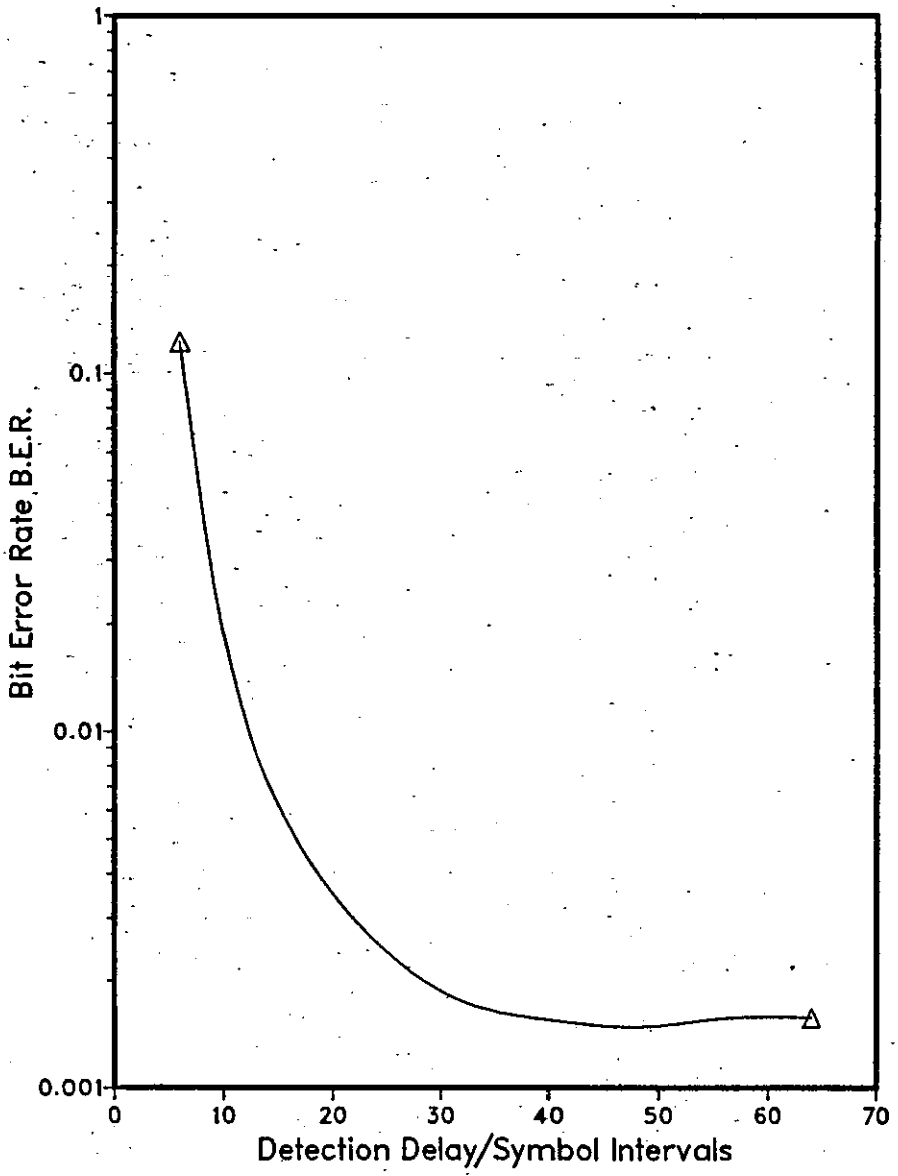
COMMON ATTRIBUTES
/M=8/

Graph 4.3.6 Variation of B.E.R. with Detection Delay at Eb/No=5.3dB



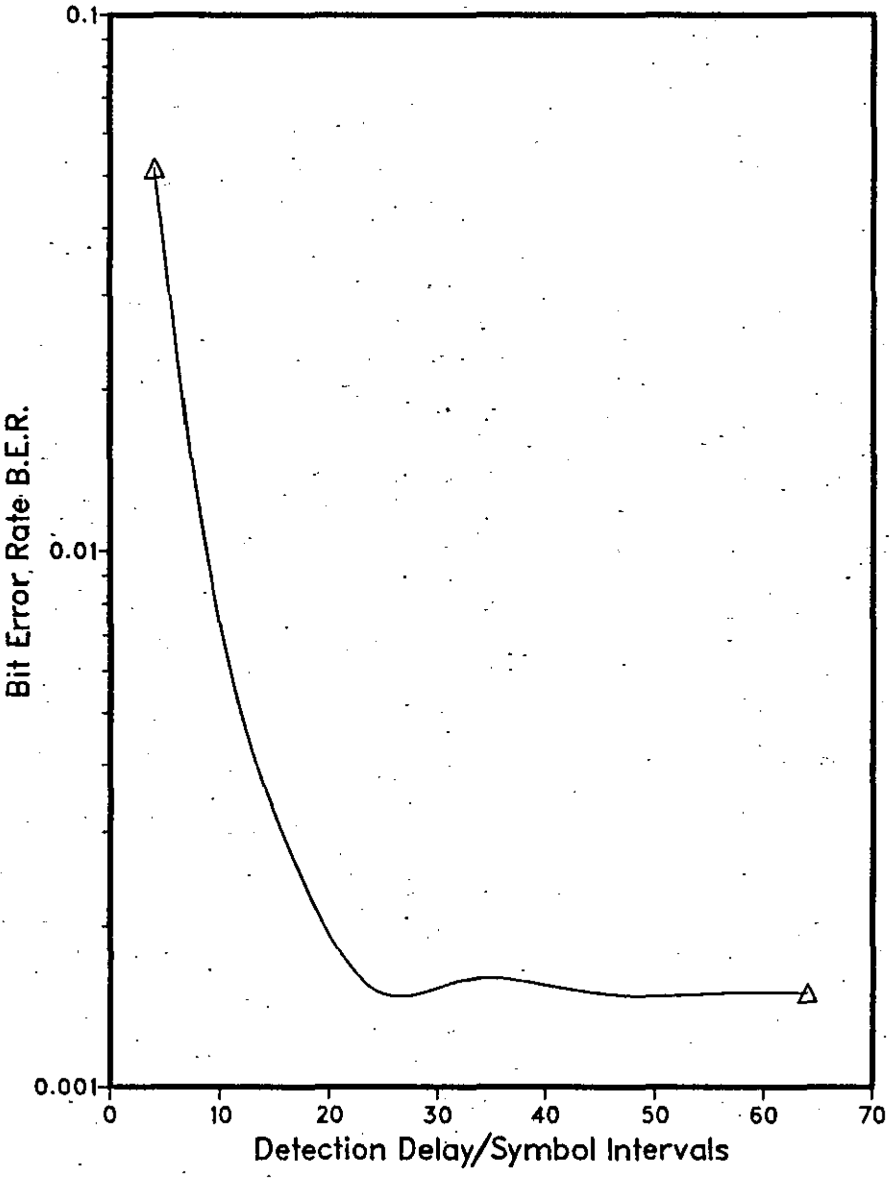
SYSTEM ATTRIBUTES
/M=8/C=3/Det=1N16/PB=Pb/

Graph 4.3.7 Variation of B.E.R. with Detection Delay at Eb/No=5.6dB



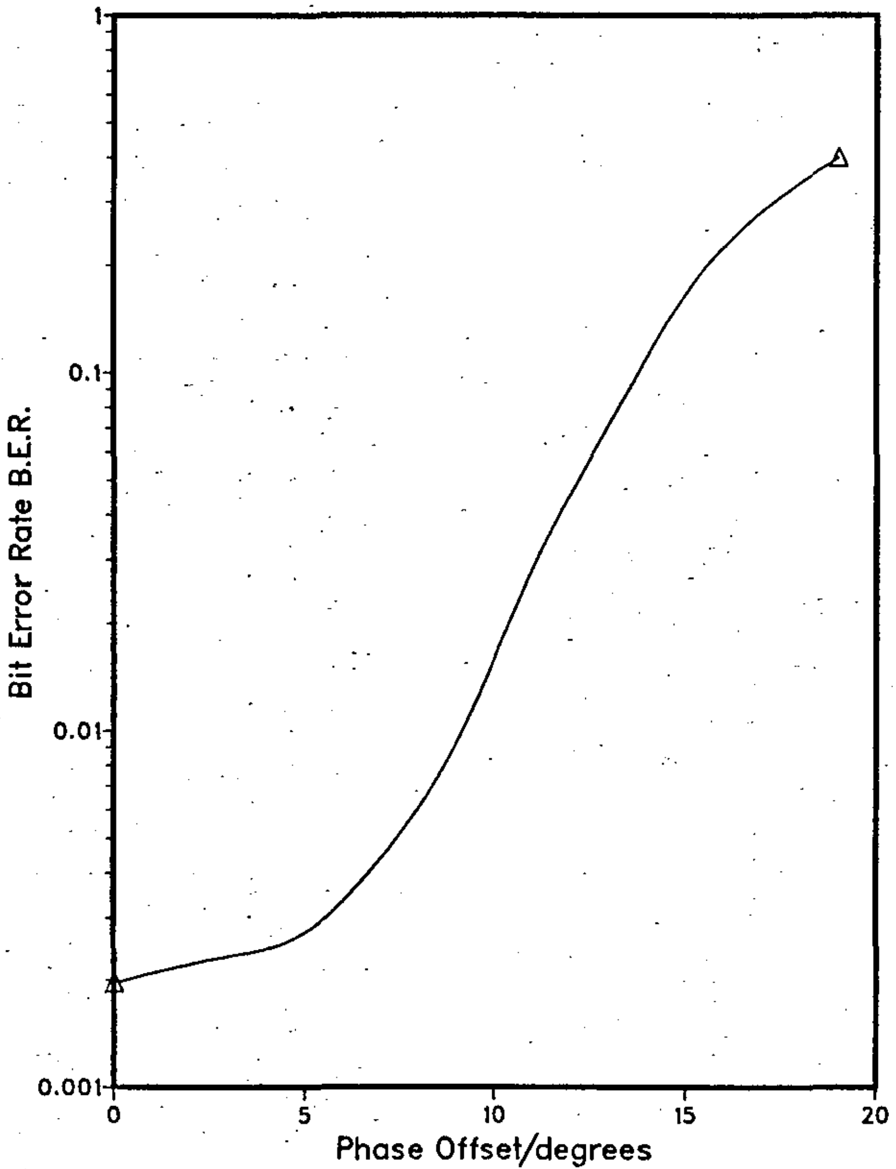
SYSTEM ATTRIBUTES
/M=8/C=3/Det=IN8/PB=Pb/

Graph 4.3.8 Variation of B.E.R. with Detection Delay at Eb/No=6.3dB



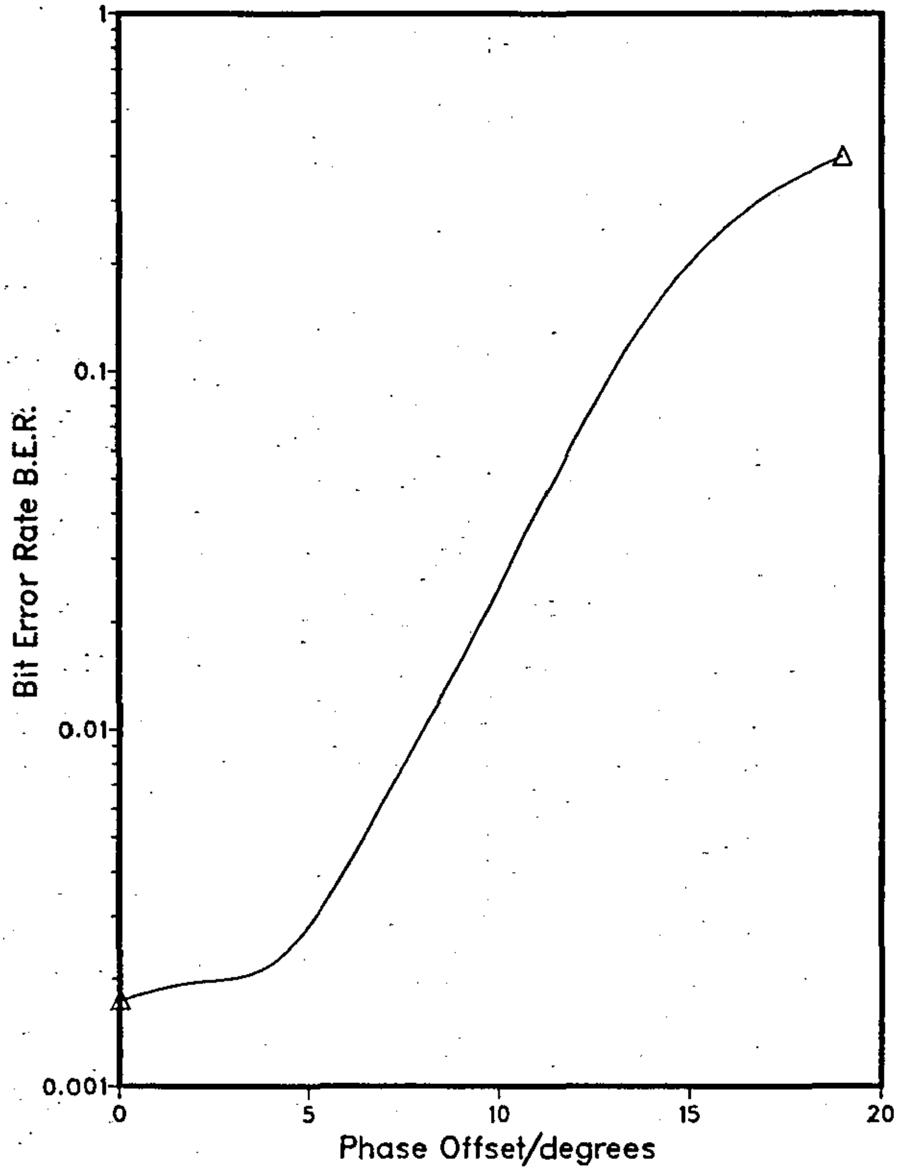
SYSTEM ATTRIBUTES
/M=8/C=3/Det=IN4/PB=Pb/

Graph 4.3.9 Variation of B.E.R. with Received Constant Carrier Phase Offset at Eb/No=5.3dB



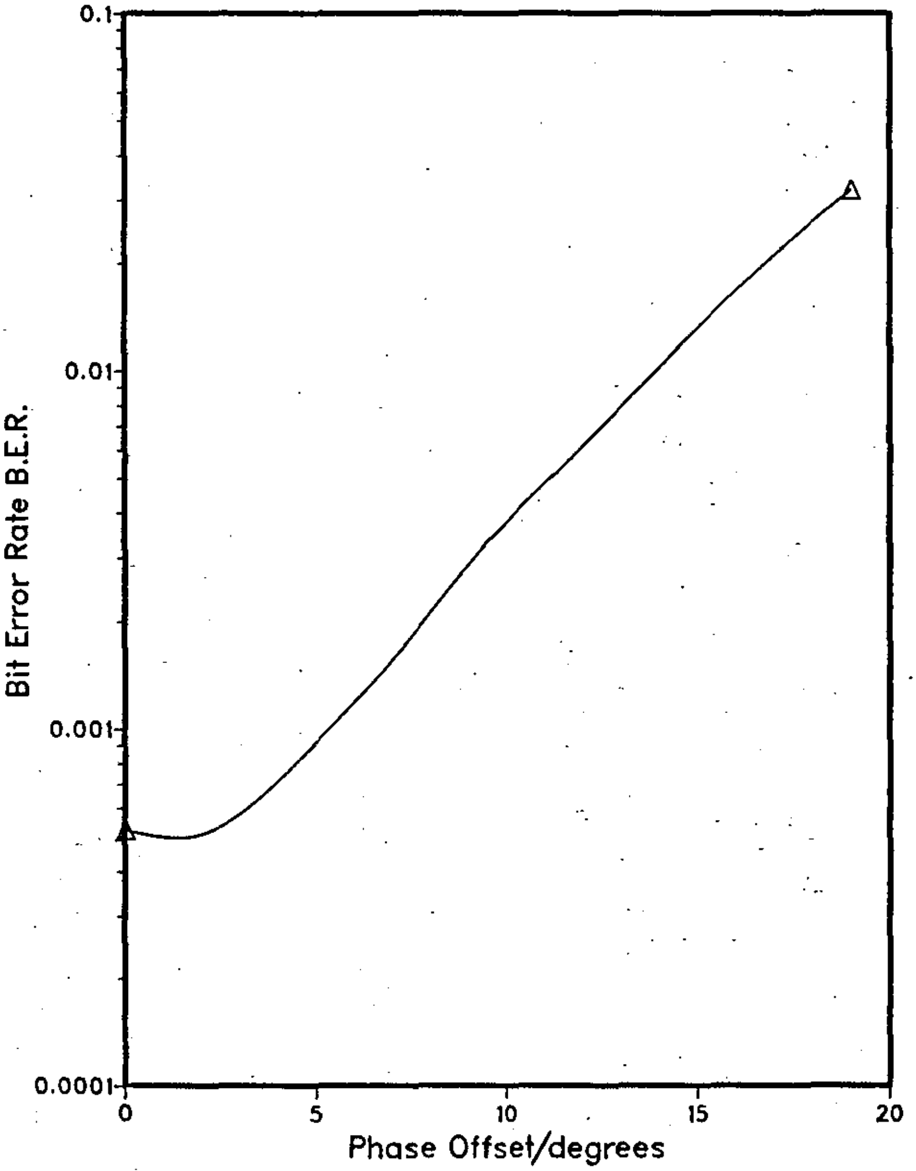
SYSTEM ATTRIBUTES
/M=8/C=3/Det=1N16/PB=Pb/

Graph 4.3.10 Variation of B.E.R. with Received Constant Carrier Phase Offset at Eb/No=5.6dB



SYSTEM ATTRIBUTES
/M=8/C=3/Det=IN8/PB=Pb/

Graph 4.3.11 Variation of B.E.R. with Received Constant Carrier Phase Offset at Eb/No=6.3dB



SYSTEM ATTRIBUTES
/M=8/C=3/Det=1N4/PB=Pb/

4.4 LOOK-FORWARD DETECTION SCHEMES

This algorithm is a development of near-maximum likelihood detection and is an attempt to provide a solution to some of the problems discussed in Sections 4.1 to 4.3, which are particularly apparent for the coded 8PSK scheme, but which also occur to some extent in all convolutionally encoded schemes. It was noted in Section 4.2 that all convolutional codes have, in a certain sense, a non-uniqueness, when a single code symbol is considered. To take the example of a constraint length-k, Rate-2/3 convolutional code, a code symbol has one of 8 possible values. In contrast there are 4^k possible combinations of the values of the four-level data symbols at the input to the coder, which determine this code symbol. Therefore, for each of the 8 possible code symbols, there are $4^k/8$ possible combinations of the data symbols. As stated in Section 4.2 therefore, no one data symbol can be detected from one received symbol in the absence of noise. As the code constraint length (k) increases, this non-uniqueness becomes greater. This is apparent, for example, in the longer error bursts for the codes where k=4 than for the codes where k=3, (see Table 3.2.2).

In conclusion the degradations in tolerance to noise for the near-maximum likelihood detectors of Sections 4.1 to 4.3, could be reduced by ammendments to the above detectors which aim to reduce the non-uniqueness discussed above. Such an attempt is embodied in the look-forward detectors. The basic idea is that the incremental costs of the four expanded vectors of vector Q'_{i-1}, should be functions of the distances between more than one received sample $\{r_j\}$, where $j \geq i$, and the corresponding possible received samples in the absence of noise,$\{p_j'\}$ By involving more than one received sample in the determination of the incremental costs it is hoped that this non-uniqueness can to some extent be reduced. Figure 4.4.1 illustrates the case for one vector Q_{i-1}'. This diagram shows the four expanded vectors at time t=iT, and the four expanded vectors for each expanded vector at time t=(i+1)T. This is the case where r_i and r_{i+1} are used to determine the incremental costs of the expanded vectors of Q_{i-1}'. The vectors at time t=(i+1)T are called level-1 extended vectors, since they extend the expanded vectors of Q_{i-1}' by one symbol. They are called extended rather than expanded vectors, since the expanded vectors are those from which the vectors $\{Q_i'\}$ are selected. There are clearly four extended vectors per expanded vector at time t=(i+1)T. In general, if ℓ samples, $r_{i+1}, r_{i+2}, \ldots, r_{i+\ell}$, are used in addition to r_i in the determination of the incremental costs, the extended vectors at time (i+j)T, for j=1,2,...,ℓ, are called level-j extended vectors, and there are clearly a total of 4^{ℓ} extended vectors per expanded vector at time t=(i+ℓ)T.

In all cases the method of operation is as follows. For a given expanded vector at time t=iT, of a given vector Q_{i-1}', the costs of a number of the 4^{ℓ} possible extended vectors at time (i+ℓ)T are calculated,

$$|W_i'|^2 = |W_{i-1}'|^2 + |w_i'|^2 + |w_{i+1}'|^2 + \ldots + |w_{i+\ell}'|^2 \quad (4.4.1)$$

Here $|w_i'|^2 + |w_{i+1}'|^2 + \ldots + |w_{i+\ell}'|^2$ is called the incremental cost. In all cases, given the $\{|W_i'|^2\}$ for a given expanded vector at time t=iT, the detector simply attributes the lowest of these costs to the given expanded vector at time t=iT by ranking the costs. In the standard implementation, the costs of all 4^{ℓ} extended vectors at time $(i+\ell)T$ are ranked. The alternative pseudobinary implementation, is a derivation of the pseudobinary scheme of Section 4.3 using the original mapping function of Figure 2.5.4. This means that the number of extended vectors is reduced to 2^{ℓ} per expanded vector at time $(i+\ell)T$. At time t=jT (where $j\geq i$) the two extended vectors of a given vector at time (j-1)T are those, of the four possible ones, with the lowest costs. Figure 4.4.2 illustrates the case for one vector Q'_{i-1}, where r_i and r_{i+1} are used to determine the incremental costs of the expanded vectors of Q'_{i-1}.

The justification for including a pseudobinary version is as follows. At reasonable error rates, errors usually involve the received sample r_i being closer to a possible value of p_i adjacent to the transmitted value or closer to a value p_i two points removed from the transmitted value. It is very unlikely that a noise spike occurs which gives a received sample r_i more than two points removed from the transmitted value in the complex number plane. Given this, the scheme as described above should not be degraded substantially compared with the standard implementation.

The standard implementation is applied to both Viterbi and System 1 detection, while the pseudobinary implementation is applied to System 1 detection. In all cases, once the costs of all the expanded vectors have been determined as described earlier, the detection processes are exactly the same as the corresponding non-look-forward schemes (Section 3.2 for Viterbi detection, Section 4.1 for System 1 detection, and Section 4.3 for pseudobinary System 1 detection).

Clearly the implementation of Equation 4.4.1 and the associated storage of extended vectors and costs, must be such that as little extra complexity as possible is entailed. All extended vectors at time $(i+\ell-1)T$ for a given vector Q'_{i-1} are stored along with their costs. At time t=iT the previous level-1 extended vectors become the expanded vectors, and the previous level-j extended vectors, for $j=2,3,\ldots\ell$, become the level-(j-1) extended vectors. The outer-most level is now level-$(\ell-1)$. For each new expanded vector (which was a level-1 extended vector), the $\alpha^{\ell-1}$ extended vectors at the outer-most level, level $-(\ell-1)$ (where α is 4 for the standard implementation and 2 for the pseudobinary implementation), are expanded to give α^{ℓ} level-ℓ extended vectors. The costs of all these α^{ℓ} extended vectors are calculated from Equation 4.1.1. These costs are ranked, and the lowest is taken to be the cost of the expanded vector, as described earlier. The remainder of the detection process is that for the corresponding non-look-forward detector.

Graphs 4.4.1 to 4.4.9 give the results of computer simulation tests of the various look-forward schemes. These are graphs of bit error rate (BER) as the signal to noise ratio, E_b/N_O, is varied. E_b is the average energy per transmitted data bit. $N_O/2$ is the two-sided power spectral density of the additive white Gaussian noise. (See Appendix A5 for more details of the simulation techniques. Appendix A8 defines the notation used to describe the variants of these detection schemes, which are tested by computer simulation.)

The accuracy of the results in the range of BER, 1 in 10^3 to 1 in 10^4, is of the order of ±0.3dB.

Graph 4.4.1 gives the results for the standard implementation of look-forward Viterbi detection for coded 8PSK using Code 1. Systems incorporating level-1 extended vectors (LF=1), and level-2 extended vectors (LF=2), are compared with the basic Viterbi Algorithm, (LF=0).

From Graph 4.4.1 there is no appreciable difference between the three schemes depicted. Table 4.4.1 outlines the error burst characteristics, in terms of the average number of bit errors per burst at various bit error rates. (See Appendix A5 for the definition of an error burst.) Table 4.4.1 suggests that the look-forward scheme may reduce the number of errors per burst for the Viterbi detector, but only very marginally, and not sufficiently to affect the scheme's performance.

Graph 4.4.2 gives the results for the standard implementation of look-forward System 1 detection where k_1=8 and Code 1 is used. Clearly some improvement is apparent, increasing as LF is increased. At a BER of 3 in 10^4 the gains in tolerance to noise, compared with the non-extended, (LF=0), scheme are, 0.3dB, 0.5dB and 0.5dB, for the schemes where LF=1, LF=2, and LF=3, respectively. The scheme where LF=1 provides the greatest incremental gain in tolerance to noise, whereas the added gains fall off as LF is increased. Therefore there is no advantage to be gained from the use of large values of LF. Table 4.4.2 gives the error burst characteristics. Clearly, these look-forward schemes reduce the number of errors per burst considerably. For the scheme where LF=3, the number of errors per burst is similar to that for Viterbi detection.

Graph 4.4.3 gives the results for the standard implementation of the look-forward technique applied to coded 8PSK using Code 1, for System 1 detection with k_1=4. The gains in tolerance to noise are relatively substantial. At a BER of 1 in 10^3 the gains over System 1 detection where k_1=4 are 0.5dB, 0.85dB, and 1.2dB, for LF=1, LF=2, and LF=3, respectively. The scheme where k_1=4, and LF=3 is equivalent, in terms of tolerance to noise, to the scheme where k_1=8, and LF=0. The error burst characteristics are given in Table 4.4.3. Clearly, although the number of errors per burst is still quite large, there is a dramatic reduction in the number of bit errors per burst, especially for the scheme where LF=3. The gains in tolerance to noise of Graph 4.4.3 are probably largely due to this reduction in the number of errors per burst.

Graph 4.4.4 gives the results for the standard implementation of look-forward System 1 detection when Code 3 is used. Clearly the gains are not as large as in the case where Code 1 is used, (Graph 4.4.2). At a BER of 3 in 10^4, the gains in tolerance to noise over the scheme where k_1=8 and LF=0 are 0.1dB, 0.1dB, and 0.3dB, for LF=1, LF=2, and LF=3, respectively. The latter scheme, where LF=3, actually gains with respect to System 1 detection where k_1=16. The error burst characteristics are given in Table 4.4.4. Clearly the reduction in the number of bit errors per burst is noticeable, but not dramatic.

Graph 4.4.5 gives the results for the standard implementation of look-forward System 1 detection, using Code 3, for k_1=4. The improved performance is more apparent than in Figure 4.4.4. At a BER of 1 in 10^3 the gains in tolerance to noise in comparison with the scheme where $k_1=4$ and LF=0 are 0.2dB, 0.4dB, and 0.5dB, for LF=1,LF=2, and LF=3, respectively. The error burst characteristics are outlined in Table 4.4.5 which indicates that the number of bit errors per burst does decrease as LF increases, but not substantially.

Graphs 4.4.6 to 4.4.9 give the results for the pseudobinary implementation of look-forward System 1 detection. In all cases the average numbers of bit errors per burst are very similar to those of the standard implementations of the look-forward System 1 schemes, which use the same number of samples $\{r_i\}$ in the calculation of the incremental costs. These graphs give comparisons with the standard implementations of look-forward System 1 detection, which use the same number of samples $\{r_i\}$ in the calculation of the incremental costs.

Graph 4.4.6 gives the results for the pseudobinary implementation of System 1 look-forward detection for coded 8PSK using Code 1, for $k_1=8$. The degradations in tolerance to noise at a BER of 1 in 10^3, where each scheme using the pseudobinary implementation is compared with the scheme using the standard implementation which has the same value of LF, are 0.45dB, 0.45dB and 0.25dB for LF=1, LF=2, and LF=3, respectively. Clearly such a comparison is unfair, since the schemes using the pseudobinary implementation are less complex, than those using the standard implementation. This will be discussed later within an analysis of the schemes' relative complexities.

Graph 4.4.7 gives the results for the pseudobinary implementation of look-forward System 1 detection, for coded 8PSK using Code 1, for $k_1=4$. The degradations in tolerance to noise compared with the standard implementations using the same values of LF at a BER of 3 in 10^3 are 0.25dB, 0.2dB and 0.2dB for LF=1, LF=2 and LF=3, respectively. Clearly at lower BERs, the losses increase somewhat.

Graph 4.4.8 gives the results for the pseudobinary implementation of look-forward System 1 detection, for Coded 8PSK using Code 3, for k_1=8. At a BER of 3 in 10^4 the degradations in tolerance to noise compared with the standard implementations using the same values of LF are 0.15dB, 0.2dB and 0.1dB, for LF=1, LF=2 and LF=3, respectively.

Graph 4.4.9 gives the results for the pseudobinary implementation of look-forward System 1 detection, for coded 8PSK using Code 3, for k_1=4. At a BER of 1 in 10^3, the degradations in tolerance to noise compared with the standard implementations using the same values of LF are 0.05dB, 0.1dB and 0.1dB for LF=1, LF=2 and LF=3, respectively.

Given that the comparisons used for the pseudobinary implementation are unfair, their performance, especially for Code 3, is definitely useful. Given that the look-forward schemes differ from the non-look-forward schemes only up to the point at which the expanded vectors' costs are determined, Table 4.4.6 gives estimates of their relative complexities, in terms of the processing time required to determine the expanded vectors' costs. Clearly, these measures do not indicate the relative complexities of the full algorithms, but they give an indication of the increase in complexity when look-forward techniques are used. (It also ignores the possibility of devising an algorithm to determine the lowest-cost extended vector at time $(i+\ell)T$ for each expanded vector without calculating and ranking the costs of all extended vectors. No such scheme has been devised.) Clearly the added complexity of the look-forward technique is not balanced by a commensurate improvement in performance. The table does indicate that the pseudobinary implementation is to be favoured, especially for large values of LF, but it is clear that these still do not provide an improved performance to balance their increase in complexity.

In the case of the look-forward Viterbi Algorithm, the absence of any noticeable performance gain is clearly because the look-forward technique cannot reduce the probability of a first error in an error burst, (the error event probability). The number of errors per burst for the Viterbi detector is also largely unchanged, when the look-forward technique is used. The gains in tolerance to noise for System 1 detection, when the look-forward techniques are incorporated, are also relatively small. Therefore it must be concluded that the look-forward technique cannot reduce the non-uniqueness, (described at the beginning of this section), in order to yield substantial gains in tolerance to noise over System 1 detection.

SCHEME	APPROXIMATE AVERAGE NUMBER OF BIT ERRORS PER BURST, AT GIVEN BER		
	3×10^{-2}	7×10^{-3}	1×10^{-3}
/LF=0/	17	13	11
/LF=1/	17	13	10
/LF=2/	15	13	10

TABLE 4.4.1: Error Burst Characteristics for Look-forward Viterbi Algorithm Detection of Coded 8PSK, Using Code 1

SCHEME	APPROXIMATE AVERAGE NUMBER OF BIT ERRORS PER BURST, AT GIVEN BER		
	2×10^{-2}	1×10^{-3}	5×10^{-4}
/LF=0/	36	40	25
/LF=1/	34	23	18
/LF=2/	27	19	14
/LF=3/	23	19	13

TABLE 4.4.2: Error Burst Characteristics for Look-forward System 1 Detection of Coded 8PSK, Using Code 1, with 8 Stored Vectors

SCHEME	APPROXIMATE AVERAGE NUMBER OF BIT ERRORS PER BURST, AT GIVEN BER	
	2×10^{-2}	1×10^{-3}
/LF=0/	140	120
/LF=1/	85	100
/LF=2/	42	50
/LF=3/	30	31

TABLE 4.4.3: Error Burst Characteristics for Look-Forward System 1 Detection of Coded 8PSK, Using Code 1, with 4 Stored Vectors

SCHEME	APPROXIMATE AVERAGE NUMBER OF BIT ERRORS PER BURST, AT GIVEN BER		
	3×10^{-2}	7×10^{-3}	1×10^{-3}
/LF=0/	30	23	18
/LF=1/	33	25	15
/LF=2/	30	20	12
/LF=3/	28	22	11

TABLE 4.4.4: Error Burst Characteristics for Look-forward System 1 Detection of Coded 8PSK, Using Code 3, with 8 Stored Vectors

SCHEME	APPROXIMATE AVERAGE NUMBER OF BIT ERRORS PER BURST, AT GIVEN BER		
	3×10^{-2}	7×10^{-3}	1×10^{-3}
/LF=0/	55	53	27
/LF=1/	45	41	35
/LF=2/	43	33	25
/LF=3/	40	30	22

TABLE 4.4.5: Error Burst Characteristics for Look-forward System 1 Detection of Coded 8PSK, for Code 3, with 4 Stored Vectors

SCHEME	APPROXIMATE RELATIVE PROCESSING TIMES PER SYMBOL INTERVAL, RELATIVE TO /Det=1N4/LF=0/PB=0/
/Det=1N4/LF=0/PB=0/	1
/Det=1N4/LF=1/PB=Pb/	2
/Det=1N4/LF=2/PB=Pb/	4
/Det=1N4/LF=3/PB=Pb/	8
/Det=1N4/LF=4/PB=Pb/	16
/Det=1N4/LF=1/PB=0/	4
/Det=1N4/LF=2/PB=0/	16
/Det=1N4/LF=3/PB=0/	64
/Det=1N8/LF=0/PB=0/	2
/Det=1N8/LF=1/PB=Pb/	4
/Det=1N8/LF=2/PB=Pb/	8
/Det=1N8/LF=3/PB=Pb/	16
/Det=1N8/LF=4/PB=Pb/	32
/Det=1N8/LF=1/PB=0/	8
/Det=1N8/LF=2/PB=0/	32
/Det=1N8/LF=3/PB=0/	128

TABLE 4.4.6: Measures of the Relative Complexity of Look-forward System 1 Detectors

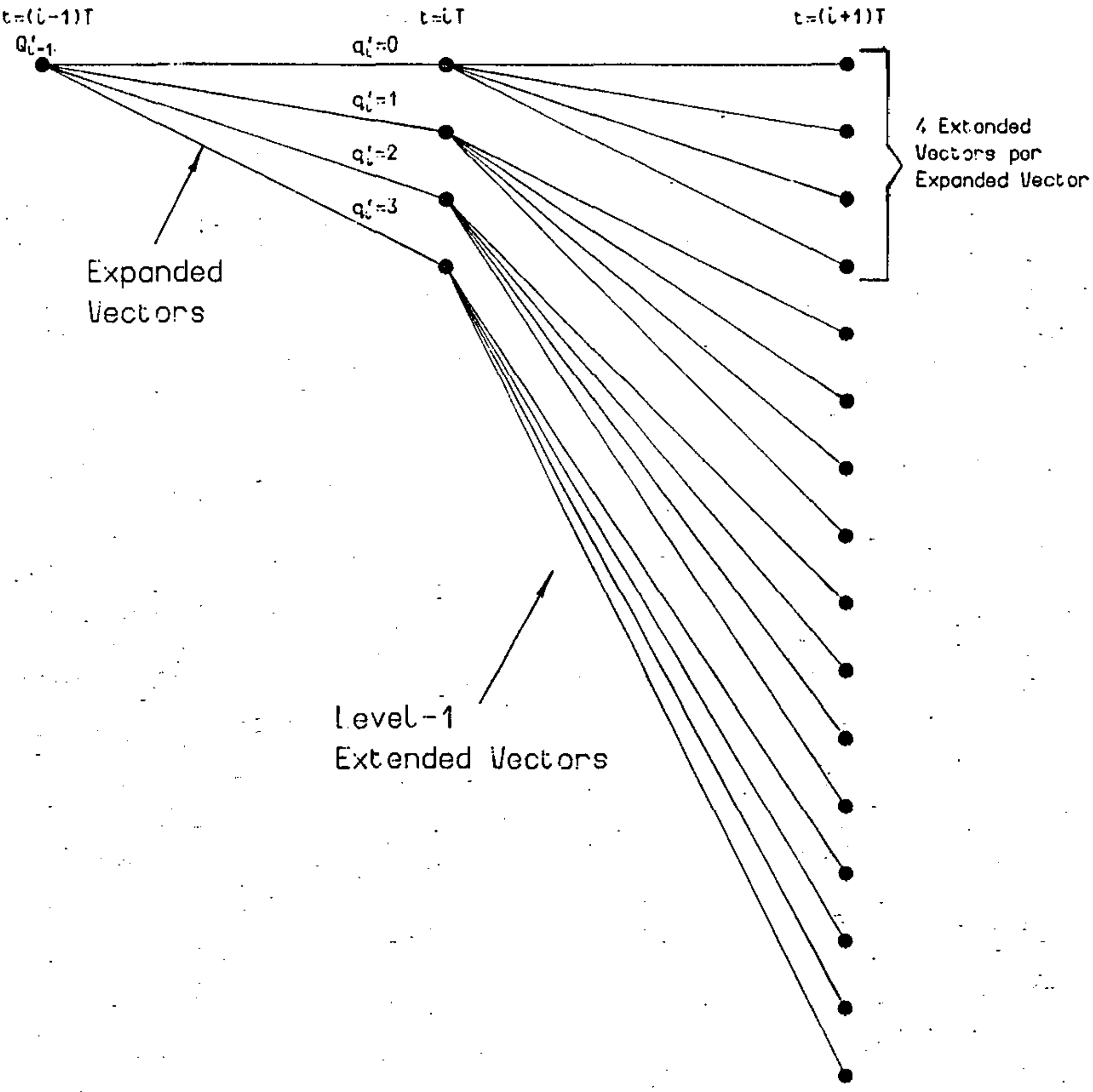


Figure 4.4.1 Expanded and Extended Vectors for One Initial Vector Single-Extension (l=1) Look-Forward Scheme

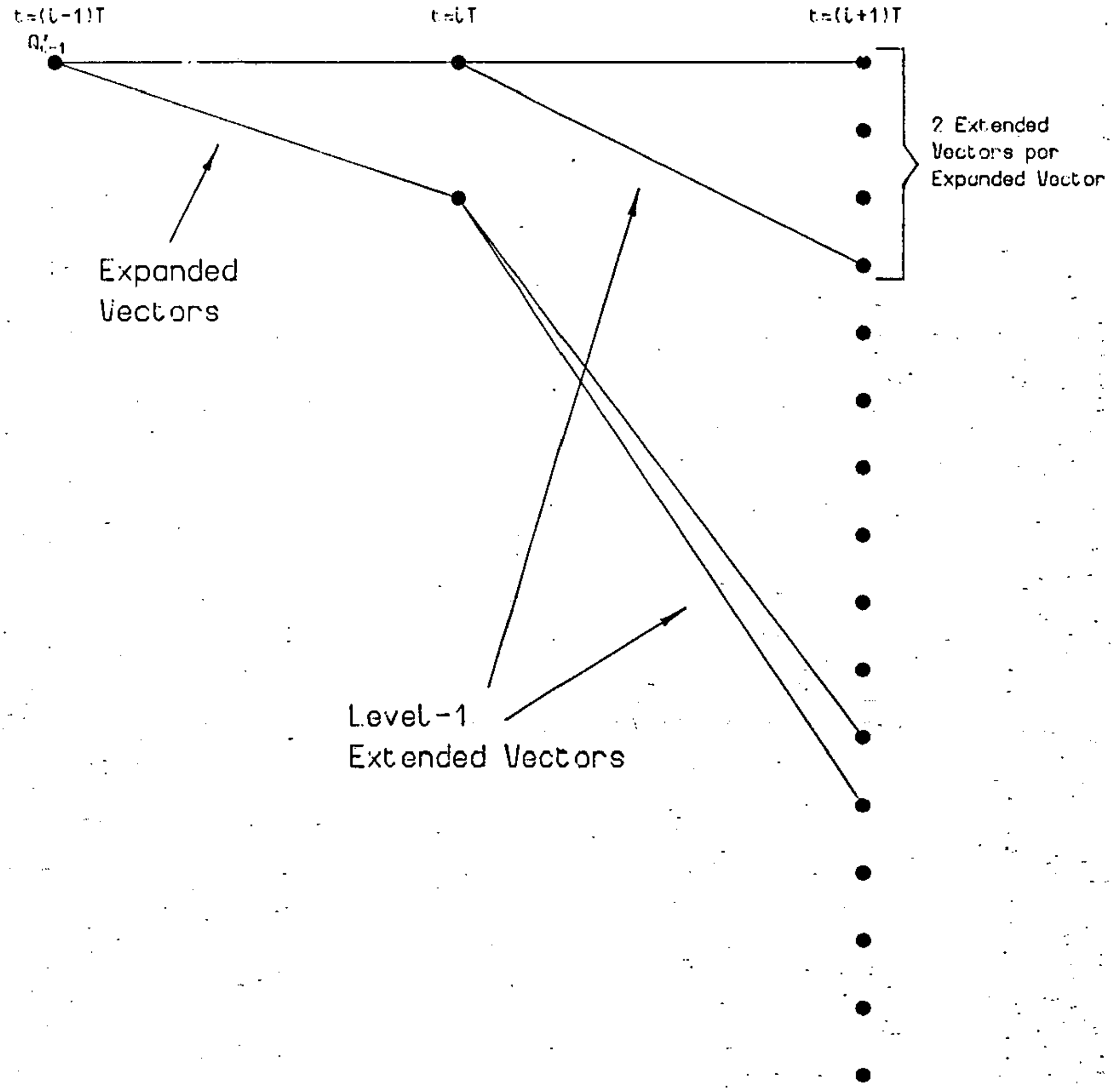
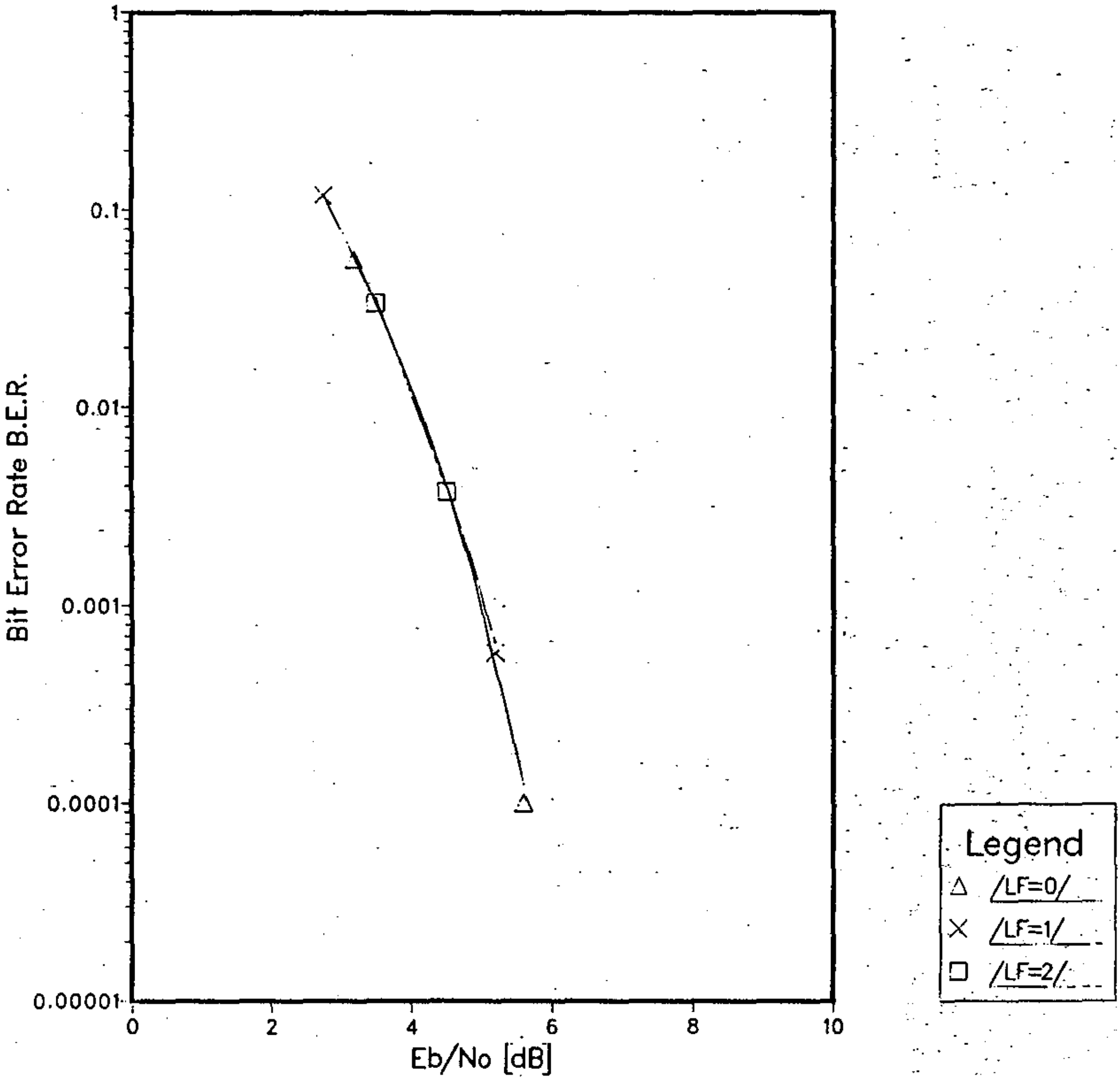


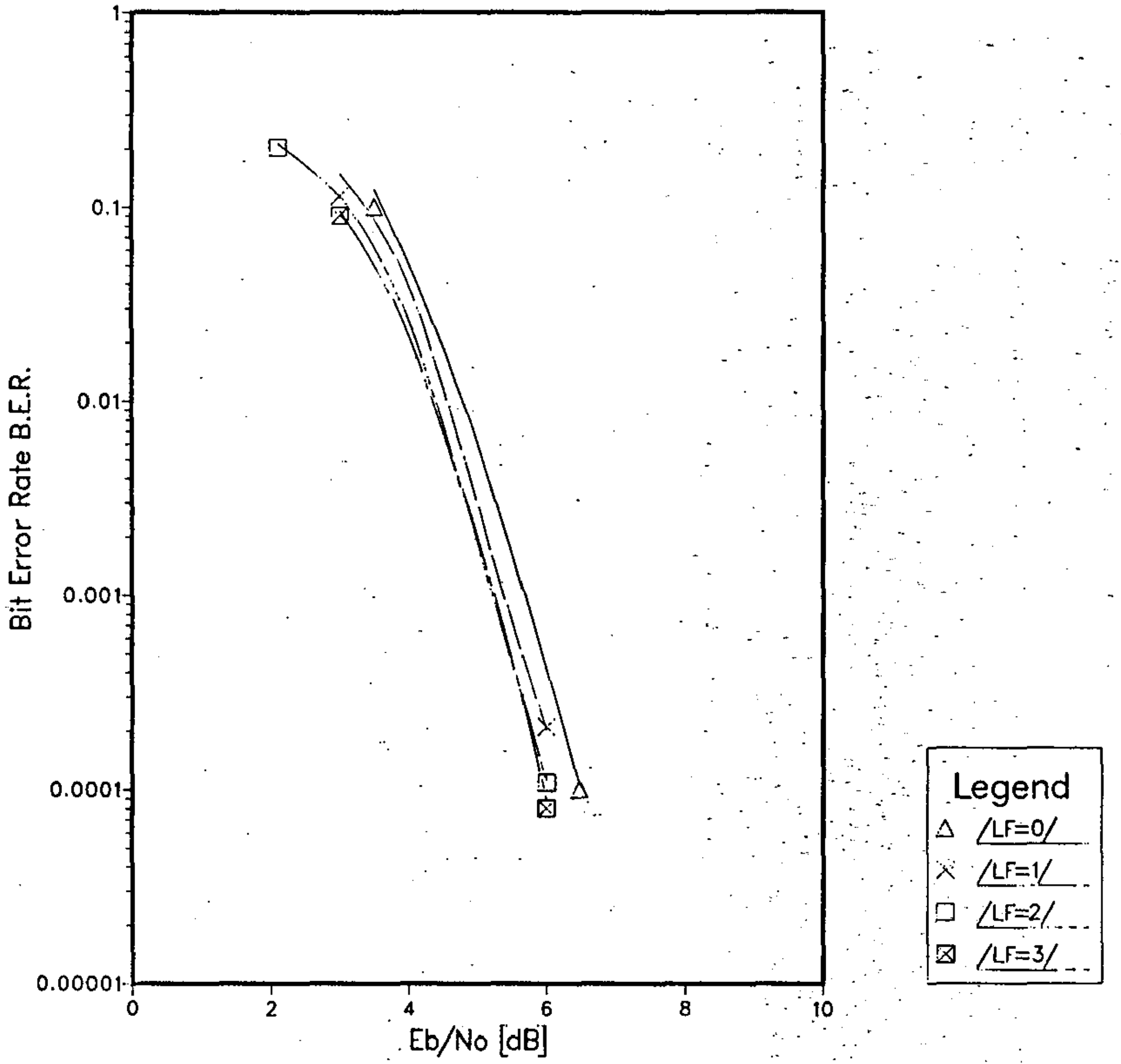
Figure 4.4.2 Expanded and Extended Vectors for One Initial Vector Single-Extension (l=1) Pseudobinary Look-Forward Scheme

Graph 4.4.1 Look Forward Viterbi Detection. Code 1



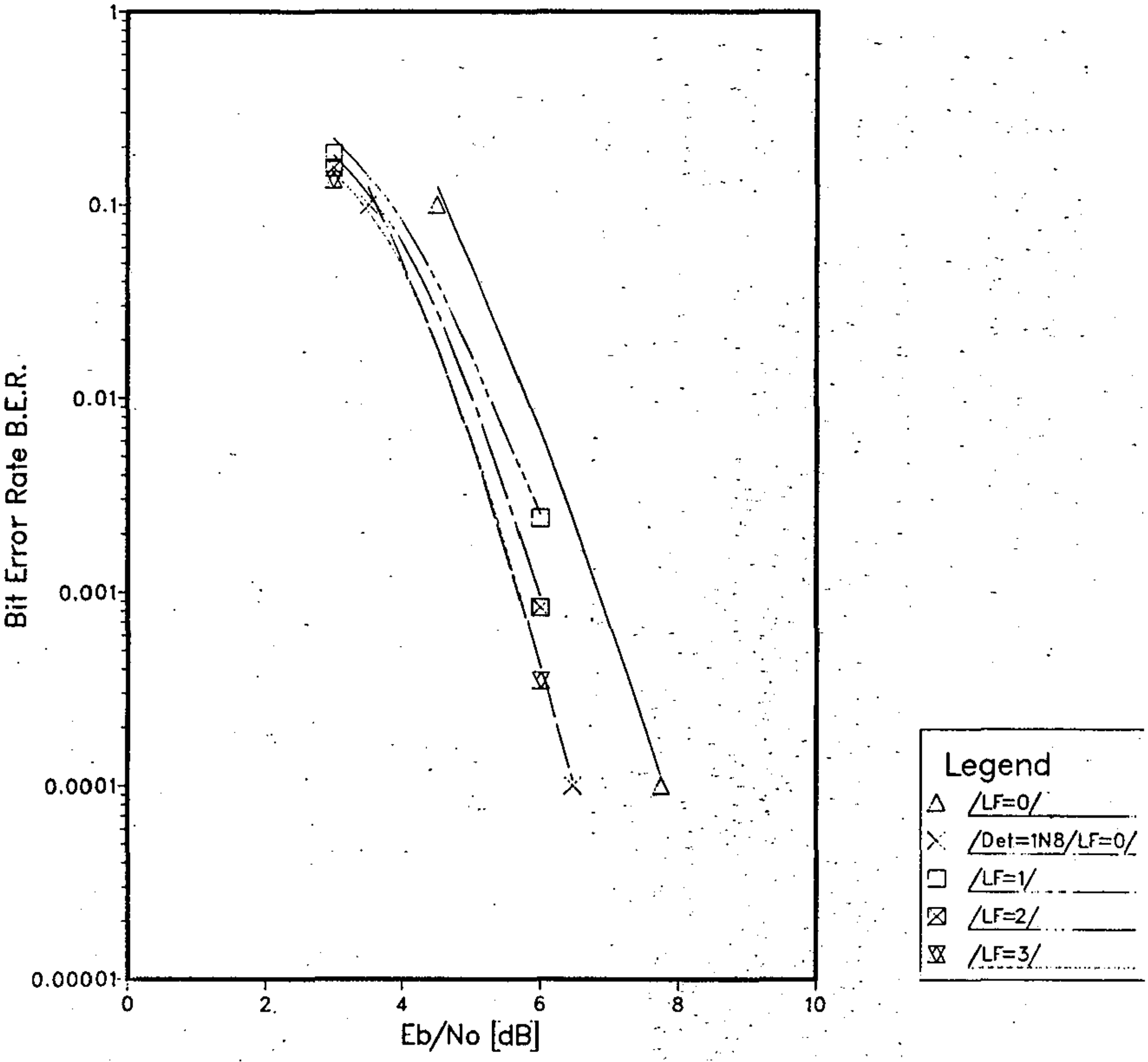
COMMON ATTRIBUTES
/M=8/C=1/Det=V16/N=64/

Graph 4.4.2 Look Forward System 1 Detection. Code 1. 8 Stored Vectors



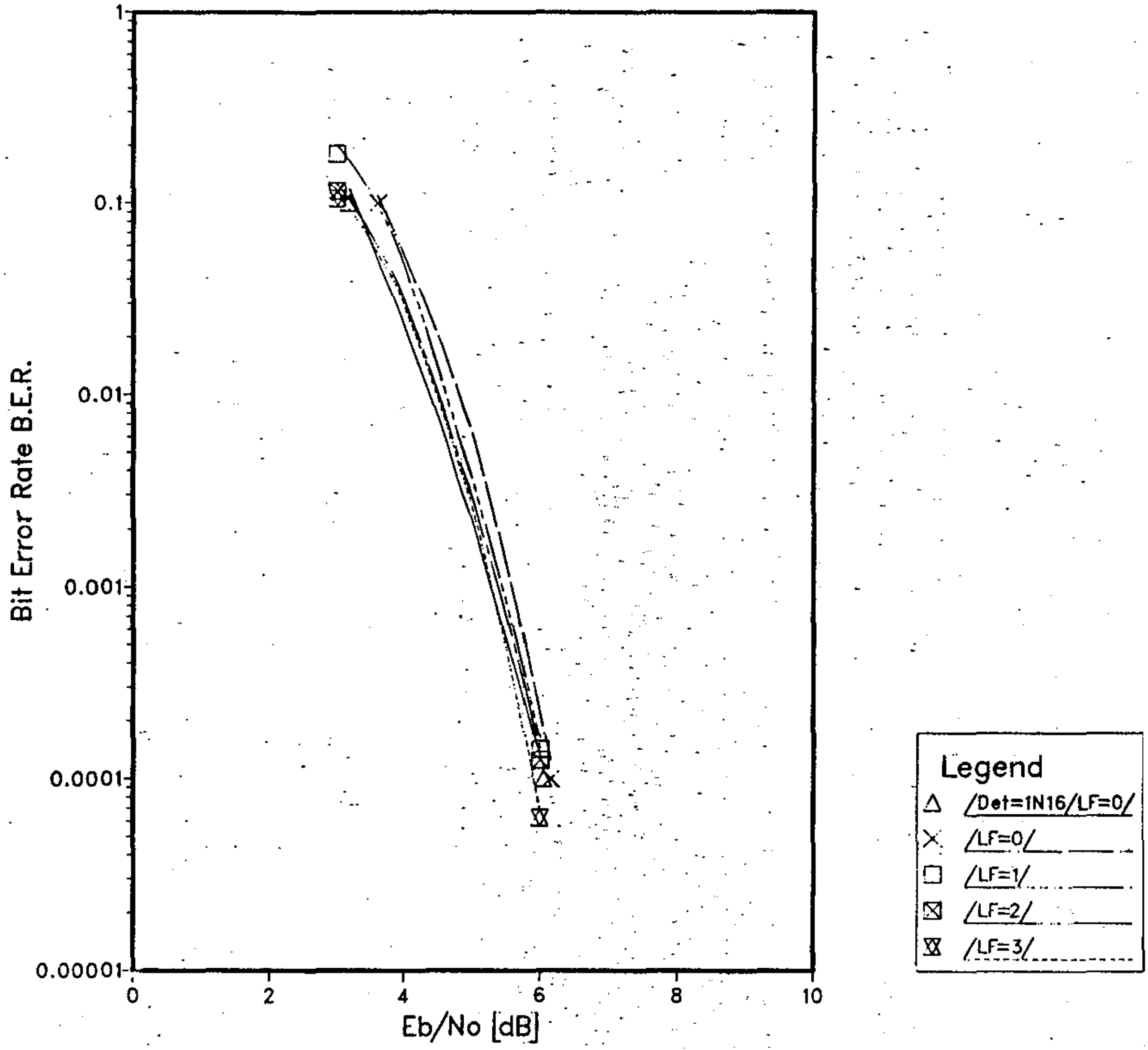
COMMON ATTRIBUTES
/M=8/C=1/Det=1N8/N=32/

Graph 4.4.3 Look Forward System 1 Detection. Code 1; 4 Stored Vectors



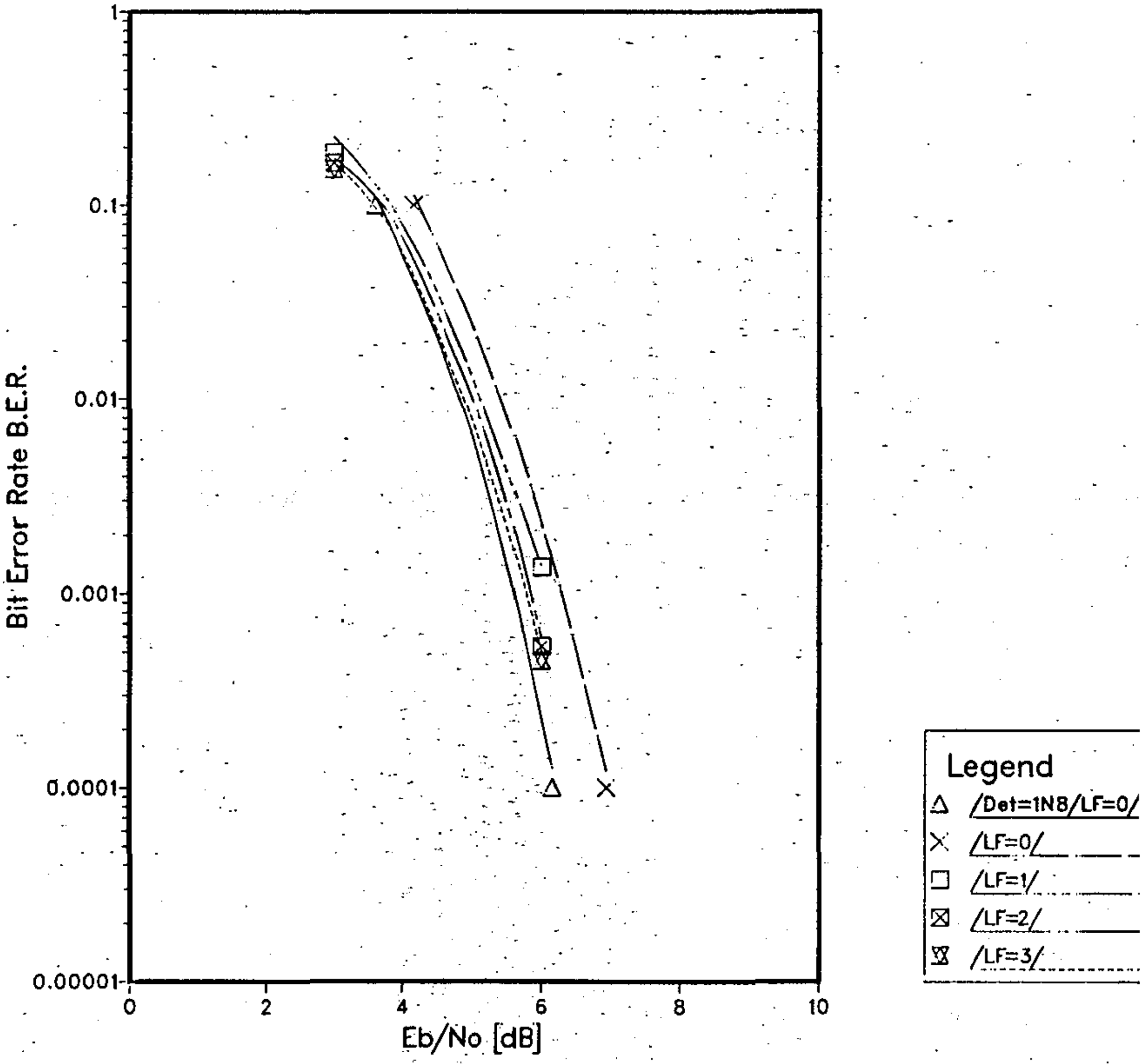
COMMON ATTRIBUTES
/M=8/C=1/Det=1N4/N=32/

Graph 4.4.4 Look Forward System 1 Detection. Code 3. 8 Stored Vectors



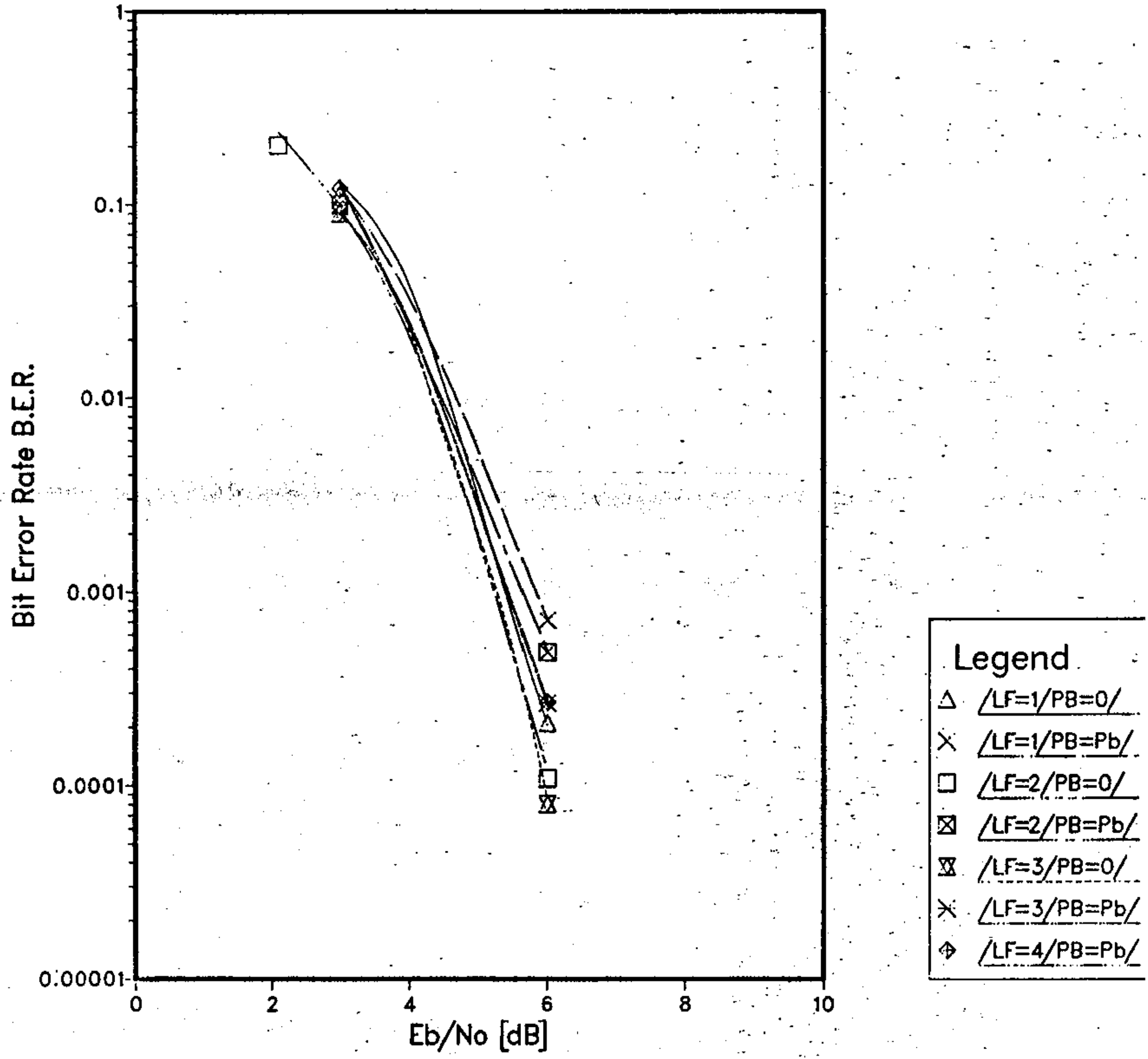
COMMON ATTRIBUTES
/M=8/C=3/Det=1N8/N=32/

Graph 4.4.5 Look Forward System 1 Detection. Code 3. 4 Stored Vectors



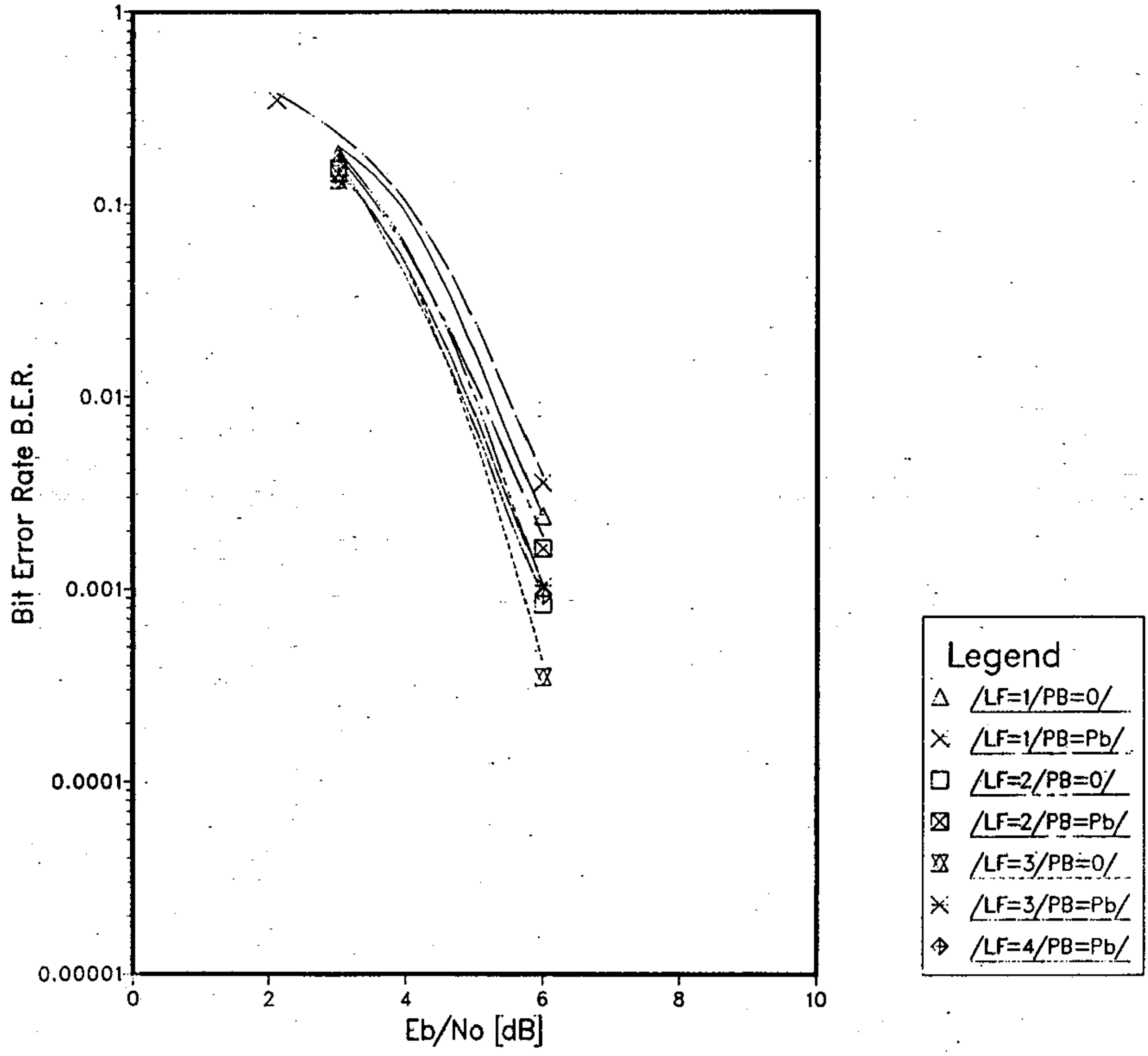
COMMON ATTRIBUTES
/M=8/C=3/Det=1N4/N=32/

Graph 4.4.6 Pseudobinary Look Forward System 1 Detection
Code 1. 8 Stored Vectors



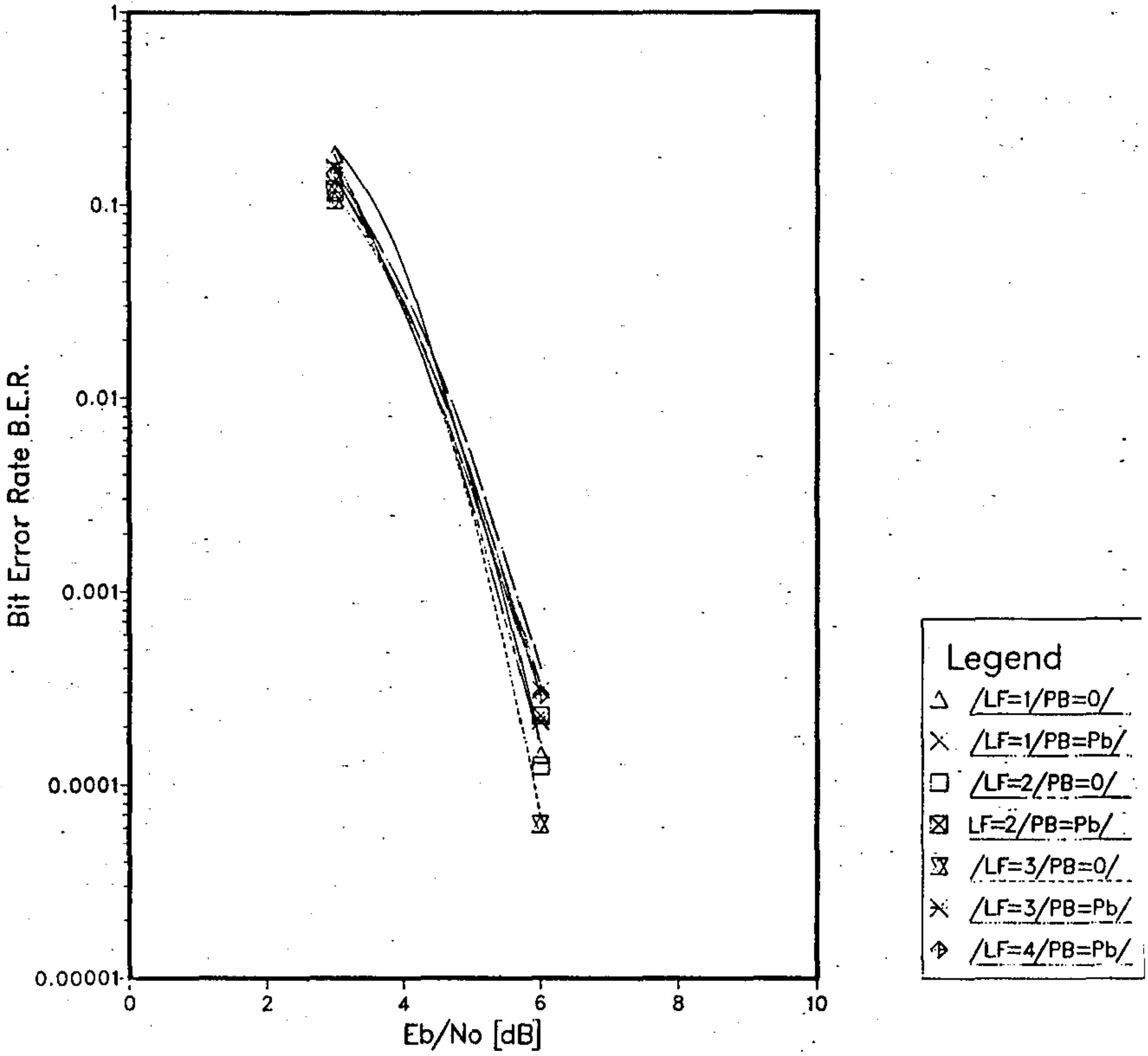
COMMON ATTRIBUTES
/M=8/C=1/Det=1N8/N=32/

Graph 4.4.7 Pseudobinary Look Forward System 1 Detection
Code 1. 4 Stored Vectors



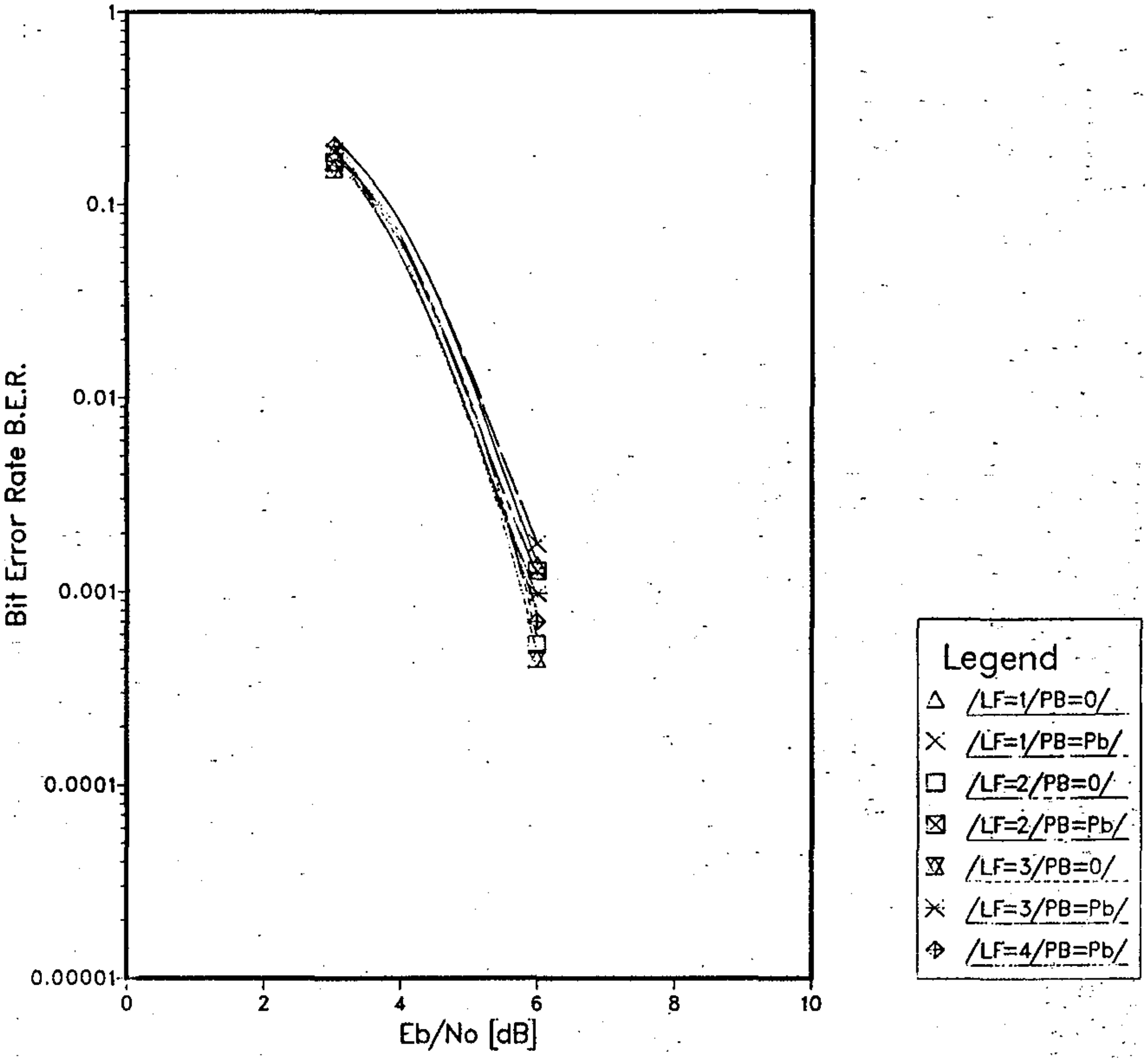
COMMON ATTRIBUTES
/M=8/C=1/Det=1N4/N=32/

Graph 4.4.8 Pseudobinary Look Forward System 1 Detection
Code 3. 8 Stored Vectors



COMMON ATTRIBUTES
/M=8/C=3/Det=1N8/N=32/

Graph 4.4.9 Pseudobinary Look Forward System 1 Detection
Code 3. 4 Stored Vectors



COMMON ATTRIBUTES
/M=8/C=3/Det=1N4/N=32/

4.5 VECTOR RETENTION-FORCING ALGORITHM FOR SYSTEM 1 DETECTION

This scheme involves a relatively minor change to System 1 detection as described in Chapter 4.1. The scheme is based on an idea by H. Najdi.[75] The basis of the amendment is an attempt to force the retention of the stored vector whose element values are those of the actual transmitted sequence of data symbols, if the noise causes this vector to have a cost which is higher than the costs of one or more of the other stored vectors. No amendment of costs is involved in this. The algorithm simply tries to retain such a vector for a long enough period, to allow the difference between its cost and that of the lowest-cost vector to decrease sufficiently, so that the correct vector will not be discarded due to the effects of the noise. Previous work[75] simply forced the retention of the vector Q'_{i-1} which has the lowest cost, as a selected vector Q'_i at time t=iT, if the latter did not have the lowest cost. For time t>iT, no attempt was made to retain this vector. From previous sections (Section 4.2 in particular) the characteristics of coded 8PSK suggest that this may not be sufficient, since quite a few symbol intervals may be required for the correct vector's cost to decrease enough to approach that of the (incorrect) lowest-cost vector. Therefore, even if retention of the correct vector is forced at the time at which it no longer has the lowest cost (at time t=iT), it could still be discarded later on. This vector retention algorithm attempts to reduce this problem by forcing the retention of the lowest cost expanded vector of a vector which was, (but is no longer), the lowest-cost vector, over a longer period of time, (ℓT seconds, where ℓ=1,2,...,11). (ℓ is called the retention period.) Such a scheme could, if ℓ is too large, be counterproductive for System 1 detection, because it could conceivably fill all, or nearly all, the available k_1 storage locations with such retained vectors. In such a case, System 1 no longer selects vectors on the basis of lowest cost, so that the vectors actually selected may have large costs and correspondingly, low likelihoods.

The algorithm is implemented as follows, for a scheme which forces the retention of such previous lowest-cost vectors, for ℓ succeeding symbol intervals. The retention algorithm is inserted after all the expanded vectors' costs have been determined, but before the System 1 selection algorithm (see Section 4.1). At this point, the detector searches through a list of previous lowest-cost vectors, $Q'_{i-\ell}, Q'_{i-\ell+1}, \ldots, Q'_{i-1}$, and notes which of these, at the times $t=(i-\ell+1)T, (i-\ell+2)T, \ldots, iT$, respectively, did not yield the corresponding lowest-cost vectors. For each such previous lowest-cost vector Q'_{i-h} noted in the above test, the algorithm ensures that the lowest-cost expanded vector of the vector Q'_{i-1} which is derived from the vector Q'_{i-h}, is retained at time $t=iT$ as a vector Q'_i. Once such a lowest-cost expanded vector has been selected for a given Q'_{i-h}, it cannot be selected again for a differing previous lowest-cost vector Q'_{i-g}, where $g \neq h$. In such a case, the single expanded vector is retained for both previous lowest-cost vectors, so that no additional expanded vector is retained for Q'_{i-g}. After this, any remaining vector storage locations are filled by implementing the System 1 selection algorithm, ensuring that any expanded vectors selected by the retention algorithm cannot be re-selected at this stage.

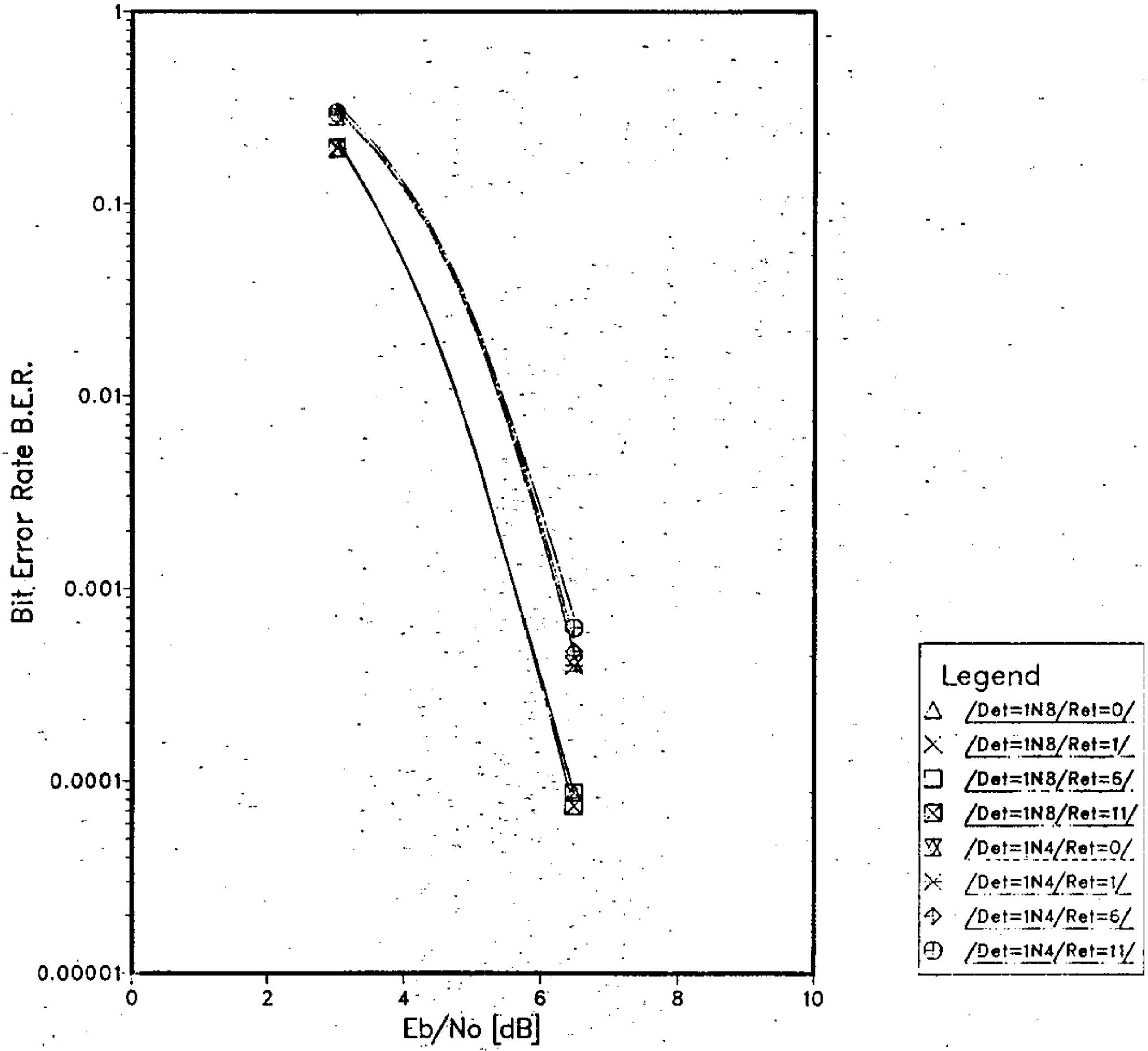
Graph 4.5.1 gives the results for the vector retention-forcing algorithm, when applied to near-maximum likelihood System 1 detection for coded 8PSK, (Code 3), with k_1=4 and k_1=8. This is a graph of bit error rate (BER) as the signal to noise ratio, E_b/N_0, is varied. E_b is the average energy transmitted per data bit. $N_0/2$ is the two-sided power spectral density of the additive white Gaussian noise. (See Appendix A5 for more details of the simulation techniques. Appendix A8 gives the notation used to describe the schemes tested.) The accuracy of the results in the range of BER, 1 in 10^3 to 1 in 10^4, is of the order of ±0.25dB. It can be seen that, for k_1=8, no performance gains are apparent, while for k_1=4, the results are inferior to System 1 detection, especially as the retention period, (Ret), is increased. At a BER of 1 in 10^3, the scheme where k_1=4 and Ret=11 is degraded by 0.15dB in tolerance to noise, compared with System 1 detection with k_1=4. The error burst characteristics, in terms of the average number of bit errors per burst, are outlined in Table 4.5.1. Appendix A5 defines an error burst. Clearly there is not a great deal of difference in the number of errors per burst, for System 1 compared with the retention-forcing variants. For k_1=4 there is a tendency for the number of errors per burst to increase, as the retention period increases. This is parallelled by an increasing degradation in tolerance to noise, from Graph 4.5.1. This can be explained by noting that all, or nearly all, the vector storage locations will be involved in the retention algorithm as the retention period is lengthened, as discussed earlier. With k_1=8, this is clearly less of a problem, since there are twice as many available storage locations, before the retention algorithm is implemented.

Clearly the vector retention-forcing algorithm does not provide any useful advantage for System 1 detection of coded 8PSK.

SCHEME	APPROXIMATE AVERAGE NUMBER OF BIT ERRORS PER BURST, AT GIVEN BER		
	3×10^{-2}	7×10^{-3}	1×10^{-3}
/Det=1N8/Ret=0/	30	23	18
/Det=1N8/Ret=1/	29	23	18
/Det=1N8/Ret=6/	29	24	19
/Det=1N8/Ret=11/	30	23	20
/Det=1N4/Ret=0/	55	53	27
/Det=1N4/Det=1/	55	48	28
/Det=1N4/Ret=6/	56	49	30
/Det=1N4/Ret=11/	58	51	35

TABLE 4.5.1: Error Burst Characteristics for System 1 Detection Incorporating the Vector Retention-Forcing Algorithm, for Coded 8PSK Using Code 3

Graph 4.5.1 Near Maximum Likelihood System 1 Detection
Vector Retention Scheme



COMMON ATTRIBUTES
/M=8/C=3/N=32/

4.6 NEAR-MAXIMUM LIKELIHOOD DETECTION BASED ON SEQUENCE NUMBERS

A reduced-complexity detection scheme has been developed, for constant envelope modulation methods.[76] The technique considers the case where many of the costs of the stored vectors in the Viterbi Algorithm detector are large compared with that of the lowest-cost vector. In such cases many of the stored vectors have very low likelihoods of containing values which are the same as those of the corresponding data symbols, and the detector should be able to discard such vectors without affecting performance very greatly. For many continuous phase modulations,[49-62] this is typical when Viterbi detection is used.

This particular scheme relies on there being a simple relationship between the difference in cost between two stored vectors and the difference in the vectors' element values. The vectors are ordered in the store in the following way (where the possible data symbols, q_i', have the values 0,1,2 or 3). The contents of each N-component stored vector (see Section 3.2),

$$Q_{i-1}' = q_{i-N}', q_{i-N+1}', \ldots, q_{i-1}' \qquad (4.6.1)$$

are used to define a sequence number, which is an integer, as below.

$$u_{i-1} = \sum_{h=0}^{i-2} 4^h q_{i-1-h}' \qquad (4.6.2)$$

where the possible data symbols $q_1', q_2', \ldots, q_{i-N-1}'$, are those shifted out of the vector in question, over previous symbol intervals. Clearly, as $i \to \infty$, $u_{i-1} \to \infty$.

If each stored vector is different, no two vectors have the same sequence number. Note that the sequence number is interpreted 'base-4' with the oldest symbol (q_1') being the most significant digit and the most recent symbol (q_{i-1}') being the least significant digit. These sequence numbers are never actually calculated. The detector simply requires to know, given two stored vectors, that their sequence numbers are separated by greater than some value s.[76] The detector orders the vectors in descending order of sequence number, the vector with the highest sequence number occup ying the first storage location. The following example indicates how this storage structure is maintained by the detector. Consider a detector which stores k_u vectors Q_{i-1}' of qua ternary symbols q_i', where their sequence numbers are the $\{u_{i-1}\}$. On receipt of r_i each stored vector forms four expanded vectors by appending the four possible values of q_i' (see Section 3.2). Clearly, the sequence numbers for the expanded vectors are of the form,

$$u_i = 4.(u_{i-1}) + q_i' \qquad (4.6.3)$$

where q_i' takes on one of its four possible values. From Equation 4.6.3 it is clear that the sequence numbers of all the expanded vectors, of a vector Q_{i-1}' whose sequence number u_{i-1} is greater than the sequence number of a second vector Q_{i-1}', are greater than the sequence numbers of the expanded vectors of the second vector. This is shown diagrammatically in Figure 4.6.1, which lists the vectors $\{Q_{i-1}'\}$ and their expanded vectors in order of sequence number. This ordering can be achieved for any modulation scheme but, in order to exploit this structure, the modulation method must have the following property.[76] There is some integer Δ, such that all stored vectors with sequence numbers separated by Δ or more have distance separations, (that is the squared unitary distance between the sequences of the $\{p_i'\}$ in the complex number plane corresponding to the element values in the vectors), of d_u^2 or more, for some value d_u. This implies a direct monotonic relationship between sequence number separation and the distance between the corresponding points in the unitary vector space.[2] Given this, the detection algorithm is as follows. $4k_u$ expanded vectors are derived from the k_u stored vectors $\{Q_{i-1}'\}$ on the receipt of sample r_i, and their costs are calculated (see Section 4.1). The detector then performs $3k_u$ selection operations, each selection operation involving just two expanded vectors. The detector considers the expanded vector with the highest sequence number, and the expanded vector with the lowest sequence number, (a selection which is very simple, see Figure 4.6.1), and compares their costs. The detector discards the expanded vector with the highest cost. There are $4k_u-1$ remaining expanded vectors at this point. The detector then repeats this operation on the reduced set of expanded vectors, yielding $4k_u-2$ remaining expanded vectors. This selection operation is performed $3k_u$ times in all, at the end of which k_u vectors $\{Q_i'\}$, remain and are stored along with their costs. As for System 1 detection, the detected symbol is taken to be the value of q_{i-N+1}' in the vector with the lowest cost. The detection operation therefore involves $3k_u$ "binary" cost ranking operations, (where binary implies the involvement of just two costs), and one ranking operation involving k_u costs to find the overall minimum cost. Simmons et al.[76] note that for constant envelope modulation schemes involving correlative-level coding, and with a modulation index of less than unity, a minimum value of k_u can be found which ensures that d_u is equal to the minimum distance between points in the unitary vector space, d_{min} (see Appendix A3). For these modulations the minimum value of k_u is $M^{(L-1)}$ for M-level data symbols q_i where the modulator's composite frequency modulating pulse has a duration of LT seconds (see Appendix A2). If k_u is such that $d_u \geq d_{min}$, Simmons et al.[76] show that the simplified detector's tolerance to noise should be asymptotic to that of Viterbi detection, at high signal to noise ratios, as long as the length of the error bursts is finite. The possible application of this detection scheme, to CORSPK(4-7,1+D) and coded 8PSK, will be discussed.

CORSPK(4-7,1+D) is a constant envelope, correlatively encoded scheme, with a modulation index of 1/2, (see Appendix A2 and Sections 2.3 and 2.4), and fits the requirements for this detection scheme, as described above. The premodulation filter is the Nyquist III-ammended 0% Roll-Off Raised Cosine filter of Section 2.4. The duration of the filter's composite frequency modulating pulse, (see Appendix A2), is taken to be 2T. (This assumes that the effect of the components of the frequency modulating pulse outside this inverval are negligible.) In this case $k_u=M^1=4$. With $k_u=4$, the number of binary cost rankings becomes 12, followed by one ranking operation through 4 costs to determine the lowest overall cost. On the other hand, the Viterbi detector stores 4 vectors and requires 4 rankings, each involving 4 costs, followed by one ranking through 4 costs to find the overall lowest cost. Making the assumption that a ranking of four costs to find the lowest is twice as complex as a similar ranking involving two costs, the sequence number-directed detector is more complex than Viterbi detection by a factor of approximately 1.4. (This assumes that the cost rankings are the most time-intensive processes in the detector.) In addition, at bit error rates of 1 in 10^3 to 1 in 10^4, the performance of the sequence number-directed scheme may well be inferior to Viterbi detection, since the detector's performance is asymptotic to that of the Viterbi detector at high signal to noise ratios. In addition this detector, as in the case of the other detectors of Chapter 4, does not guarantee the retention of the Maximum Likelihood vector, (see Sections 4.2 and 4.3), although because the costs are widely separated for CORPSK(4-7,1+D), this is not as great a problem as for coded 8PSK, (see Section 4.2). In conclusion, no advantage is gained from applying this detector to the CORPSK(4-7,1+D) modulation scheme.

On the face of it, the application of the sequence number-directed detection scheme to coded 8PSK would seem to be beneficial, since the Viterbi detectors require a large number of stored vectors (16 for the constraint length k=3 codes, 64 for the k=4 codes). Unfortunately, there are a number of problems. To begin with, there is no formula for the minimum value of k_u as given above for continuous phase modulations. In addition, the neat ordering of the vectors breaks down for coded 8PSK, since there is no simple monotonic relationship between the sequence numbers of the possible data sequences, and the distances between the corresponding points in unitary vector space. This is due to the coding and phase mapping. Therefore in this application the sequence numbers would have to be based on the coded 8-level symbols, rather than on the possible data symbol values, and would have to be calculated, after which the stored vectors would have to be ordered according to their sequence numbers. Note that these sequence numbers cannot be interpreted 'base-8', in the same way that those of the 4-level data in the CORPSK(4-7,1+D) scheme are interpreted 'base-4', since some way of noting that the complex number p_i mapped from the coded symbol $c_i=\emptyset$ is very close to the complex number p_i mapped from the coded symbol $c_i=7$, is required. Simply interpreting the coded symbol sequences 'base-8' would imply that two vectors which only differ in one code symbol, $c_i=\emptyset$ and $c_i=7$ respectively, are much further apart in the unitary space, than two vectors which differ in the same code symbol, where $c_i=\emptyset$ and $c_i=1$ respectively. In fact in both cases the distance in unitary vector space is identical. The sequence number-directed scheme is clearly designed for differential-phase schemes, where the phase at a given time instant is the accumulation of previous phase shifts. In such cases, a sequence number difference between two vectors due to a difference in their respective values of symbol q'_{i-h}, produces a larger cost difference between the vectors than for a sequence number difference due to a difference in their respective values of symbol $q'_{i-\ell}$, where $h<\ell$. This is because the cost difference is cummulative for time $t>hT$ in the first case, and time $t>\ell T$ in the second case, since the signal phase itself is cummulative. This means that the difference in the value of the earlier symbol, q'_{i-h}, has had a longer period of time over which the cost difference can increase, compared with the difference in the value of $q'_{i-\ell}$. Clearly the 'base-4' sequence number definition reflects this fact by making the older symbol, q'_{i-h}, more significant than the more recent symbol, $q'_{i-\ell}$, (see Equation 4.6.2). Providing a sequence number definition for the direct phase mapping associated with coded 8PSK is a much more difficult task and will not have the preferred structure of Figure 4.6.1. This means that the sequence numbers of the expanded vectors derived from a vector Q'_{i-1}, where its sequence number u_{i-1} is greater than that of a second vector Q'_{i-1}, may not all be greater than the sequence numbers of the expanded vectors of the second vector. This would obviously negate the major advantage of the scheme as envisaged for continuous phase modulation, where vector reordering is not required. In addition, the initial premise (which holds for continuous phase schemes), that many of the vectors' costs are large compared to the lowest cost, is not as valid for coded 8PSK. Section 4.2 considers the properties of coded 8PSK which can produce very similar costs among the stored vectors. Because of this k_u may well need to be large for coded 8PSK. Therefore sequence number-directed detection is not considered as a practical alternative for coded 8PSK.

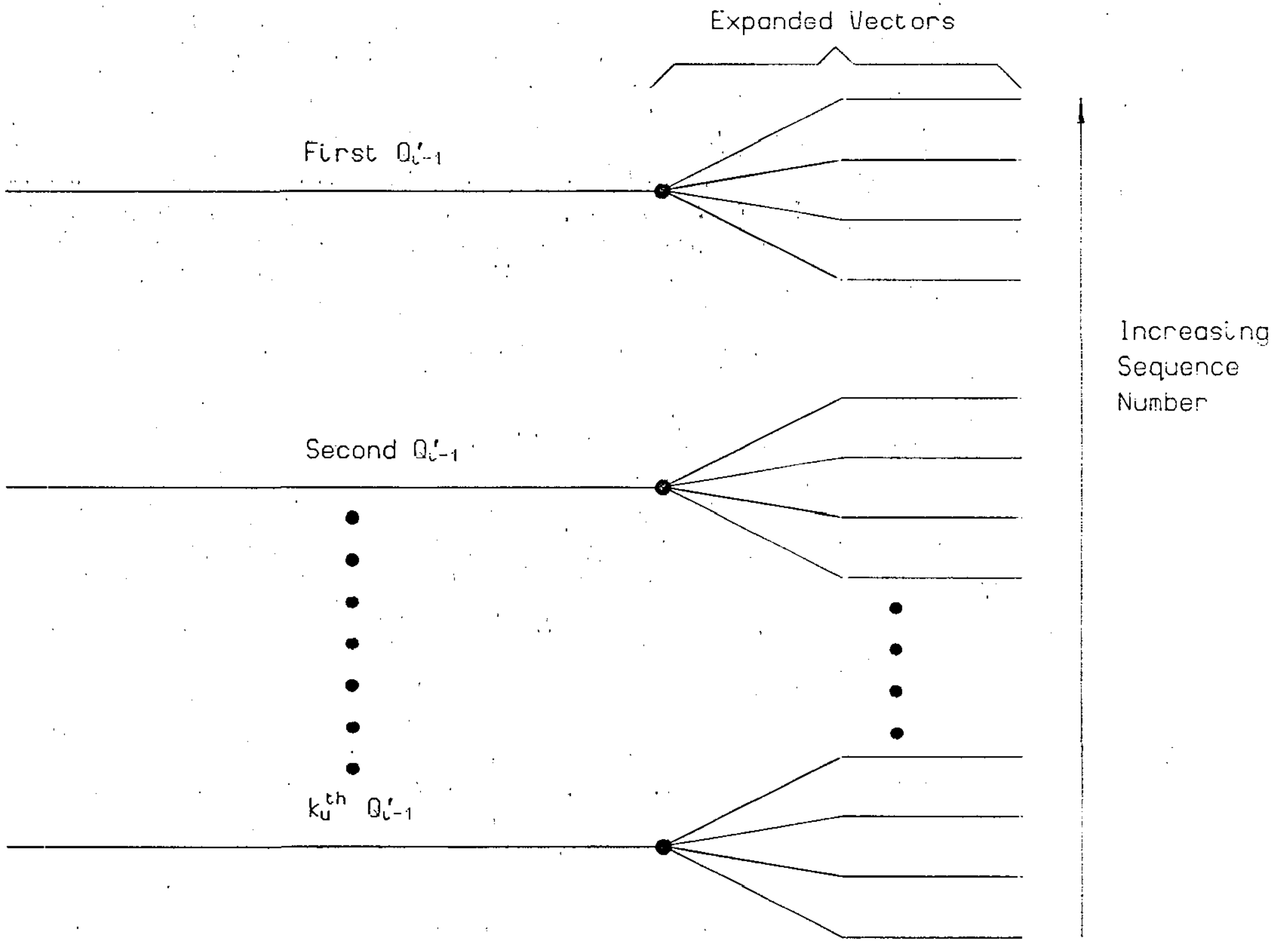


Figure 4.6.1 Sequence Number-Directed Vector Storage

Chapter 5

SUBOPTIMAL DETECTION SCHEMES FOR CODED 8PSK

This Chapter describes a number of detectors which differ significantly from the Maximum Likelihood techniques of Chapter 3, and the near-maximum likelihood techniques of Chapter 4. Only one, (Section 5.3), holds stored vectors of possible data symbol values, as do the detectors of Section 3.2 and Chapter 4. This is an amendment of the Viterbi detector which redefines the meaning of a state, (as did the pseudobinary Viterbi detector of Section 4.3). Section 5.1 describes a detector which operates in much the same way as a nonlinear equaliser.[1,2] It is equivalent to System 1 detection, (Section 4.1), with only one stored vector. Section 5.2 describes a very simple detector which uses the feedforward filter which is the inverse of the convolutional coder at the transmitter. Section 5.4 describes a detector which uses a syndrome decoding technique. Such techniques are widely used to decode block codes.[19]

Table A8.1 defines the notation which is used to describe the schemes whose performance is tested by computer simulation.

5.1 PSEUDO-NONLINEAR EQUALISER; A DECISION-FEEDBACK TECHNIQUE

Figure 5.1.1 illustrates the configuration of the pseudo-nonlinear equaliser at the receiver for coded 8PSK. This is a process of decision-directed cancellation of components of the received sample r_i, involving already detected data $\{q'_j\}$, where $j<i$. The decision-directed cancellation is here effected by feeding back the detected data symbols to a store of k-1 previous detected data symbols $q'_{i-k+1}, q'_{i-k+2}, \ldots, q'_{i-1}$, where k is the constraint length of the code. The coder, (Section 2.5), uses these values and each of the four possible values of q'_i, to give the four possible vectors of binary code symbols $[c'_i(1), c'_i(2), c'_i(3)]$, where

$$c'_i(j) = \bigoplus_{\ell=1}^{2} \bigoplus_{h=0}^{k-1} q'_{i-h}(\ell) g_h(\ell,j) \tag{5.1.1}$$

$[q'_i(1),q'_i(2)]$ is a two-component vector that is uniquely related to q'_i according to Table 2.1.1. The $\{g_h(\ell,j)\}$ are binary-valued and $\bigoplus$ denotes MODULO-2 summation. Each vector $[c'_i(1),c'_i(2),c'_i(3)]$ is mapped onto the eight level code symbol c'_i

$$c'_i = 2^2 c'_i(1) + 2^1 c'_i(2) + 2^0 c'_i(3) \tag{5.1.2}$$

Since $c'_i(1),c'_i(2)$, and $c'_i(3)$ each have the two possible values 0 or 1, c'_i takes on one of the eight values 0,1,...,7. A possible value of the received sample r_i in the absence of noise is given by mapping each of the four possible values of c'_i onto a complex number p'_i, where the mapping is defined in Figure 2.5.4. These $\{p'_i\}$ are used to calculate the costs of choosing each possible value of q'_i as the value of the data symbol q_i.

$$\begin{aligned} |w'_i|^2 &= |r_i - p'_i|^2 \\ &= \{\mathrm{Re}(r_i - p'_i)\}^2 + \{\mathrm{Im}(r_i - p'_i)\}^2 \end{aligned} \tag{5.1.3}$$

The value of q'_i with the lowest cost is taken to be the detected data symbol, (no precoding having been used at the transmitter, see Section 2.1), and is fed back to the store of k-1 detected data symbols. This now consists of the detected data symbols $q'_{i-k+2},q'_{i-k+3},\ldots,q'_i$. Clearly this is equivalent to near-maximum likelihood System 1 detection (Section 4.1) with one stored vector (k_1=1) and no delay in detection.

Graph 5.1.1 gives the results of computer simulation tests, where

Code 3 is used. This is a graph of bit error rate (BER) as the signal to noise ratio, E_b/N_O, is varied. E_b is the average energy transmitted per data bit and $N_O/2$ is the two-sided power spectral density of the additive white Gaussian noise. (See Appendix A5 for more details of the simulation techniques. Appendix A8 gives the notation used to describe the variants of System 1 which were tested by computer simulation.) Because of the large size of the error bursts, (see later), the accuracy of the results is of the order of ±0.6dB in the range of BER, 1 in 10^3 to 1 in 10^4. The definition of an error burst is given in Appendix A5. At a BER of 1 in 10^4 the pseudo-nonlinear equaliser has a tolerance to noise which is approximately 2.3dB worse than that of threshold detected QPSK, and 5.3dB worse than that of (64 vector) Viterbi-detected coded 8PSK using Code 3. Table 5.1.1 gives the error burst characteristics in terms of the average number of bit errors per burst at a number of BERs. The fact that the average number of bit errors per burst increases as the BER decreases, indicates that future detected data symbol values often only become the same as the actual data symbol values after further noise-induced errors in the received samples. This is because, as the noise level reduces, such noise-induced effects become less likely, so that the number of errors per burst increases. Once a false detection q'_{i-1} has been made, the store of previous detected data is incorrect. As a result the values of p'_i produced on the receipt of the next sample r_i will in all probability be incorrect. Therefore, in all probability the next detected data symbol, q'_i will be incorrect. Errors will therefore perpetuate as long as the values in the store of previous detected symbols are incorrect. Resumption of correct detection is code-dependent in that it depends on the likelihood, given one or more incorrect previous detections, that the correct values of p_i can be produced by the correct (k-1) following data symbol values. For Codes 2 and 4 the error burst were thousands of bits long, so that accurate curves cannot be produced. For the latter codes it seems that this likelihood described above is less than for Code 3. (The results agree with those for System 1 detection in Section 4.1, where schemes using Codes 2 and 4 produced very long error bursts.) Given correct decision-directed cancellation, the performance of the system is similar to that of QPSK, since the set of possible values of $\{p_i'\}$ for the four possible values of q_i', are all in one of the two complex number plane diagrams of Figure 4.2.1. The cost ranking exercise in such a case, (see Figure 5.1.1), becomes equivalent to threshold detection. This can also be seen in Graph 5.1.1, which indicates that the performance of the pseudo-nonlinear equaliser may well be asymptotic to that of QPSK, at high signal to noise ratios.

APPROXIMATE AVERAGE NUMBER OF BIT ERRORS PER BURST AT GIVEN BER		
1×10^{-2}	1×10^{-3}	1×10^{-4}
150	170	200

TABLE 5.1.1: Error Burst Characteristics for Pseudo-Nonlinear Equaliser Detection for Coded 8PSK, using Code 3

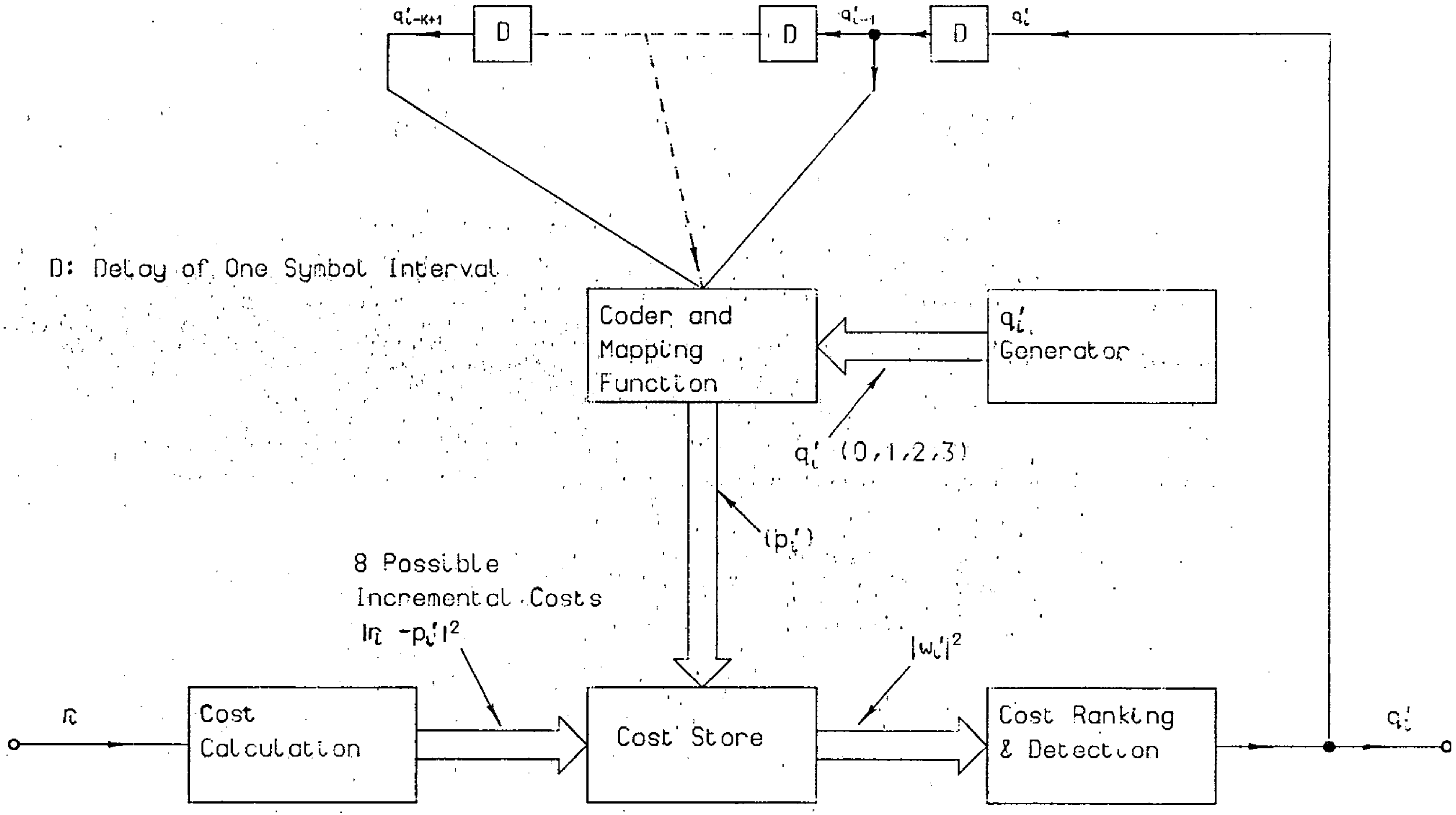
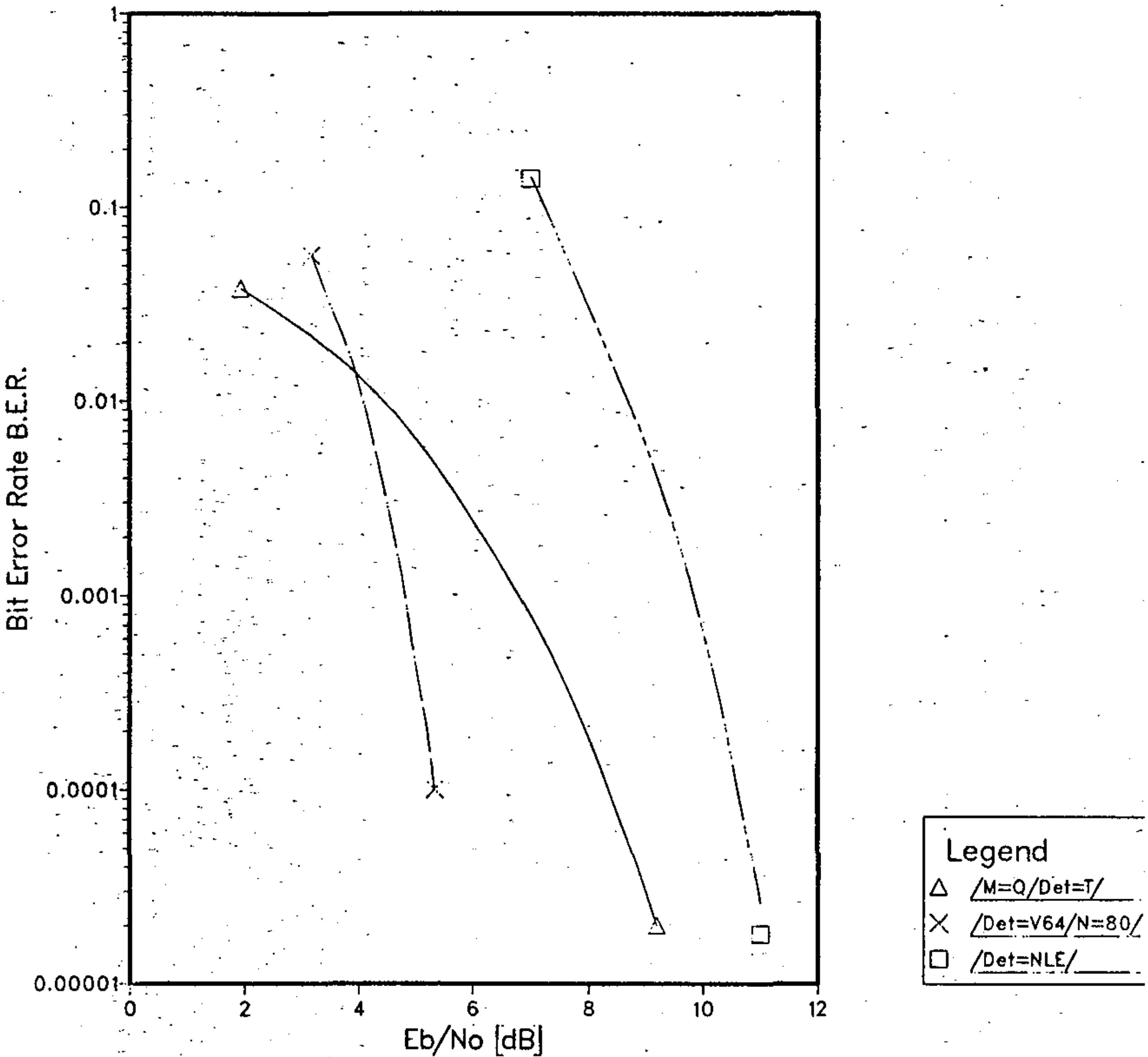


Figure 5.1.1 Pseudo-Nonlinear Equaliser

Graph 5.1.1 Pseudo-Nonlinear Equaliser



COMMON ATTRIBUTES
/M=8/C=3/

5.2 INVERSE CODER; A FEEDFORWARD TECHNIQUE

This system is an attempt to find the detector for coded 8PSK with the minimum dependence on previous detection decisions, (in contrast to the pseudo-nonlinear equaliser of Section 5.1). It was thought that such a detector could form the basis of a two-stage detection process where the initial (soft) detected data, provided by a simple detector with minimum dependence on previous detection decisions, is improved upon by a more sophisticated second-stage detector. It was hoped that such a detector would not have the error burst problems of the detectors of Chapter 4, and would require a second-stage detector which is relatively simple compared with the Viterbi Algorithm detector.

Appendix A6 deals at length with the procedure for determining the feedforward, (tapped delay line), inverse of a feedback-free convolutional encoder. References 77 and 78 provide much of the theory which is used in Appendix A6. Appendix A6 provides an inverse coder of this type for Code 1, given below in matrix form. See Appendix A6 for a description of this matrix presentation, where the matrix elements are polynomials in the delay operator D.

$$G^{-1}(D) = \begin{bmatrix} D^2 & 1+D+D^2 \\ 1+D & D \\ D^2+D^3 & 1+D^3 \end{bmatrix} \qquad (5.2.1)$$

Figure 5.2.1 is a diagram of the inverse coder. The input symbols, $c_i'(1)$, $c_i'(2)$, and $c_i'(3)$ are binary valued and the binary output symbols $q_i'(1)$ and $q_i'(2)$ are uniquely related to the four-level detected data symbol q_i', by the Gray Code Mapping of Table 2.1.1. The values of

$c_i'(1)$,$c_2'(2)$ and $c_i'(3)$ are determined as follows. Each received sample r_i, is tested to find the point p_i' in the complex number plane, of the 8 possible points (see Figure 2.5.4), which is nearest to r_i. In this way the sequence which is an estimate of the code sequence $\{c_i\}$ at the transmitter, (that is the symbols $\{c_i'\}$), is produced, by mapping the chosen values of $\{p_i'\}$ onto the $\{c_i'\}$, (see Figure 2.5.4). Each code symbol c_i' is mapped onto the vector of binary symbols $[c_i'(1),c_i'(2),c_3'(3)]$. The relationship between c_i' and this vector was given in Equation 5.1.2.

Graph 5.2.1 gives the results of computer simulation tests on this system for coded 8PSK using Code 1. This is a graph of bit error rate (BER) as the signal to noise ratio, E_b/N_O, is varied. E_b is the average energy transmitted per data bit and $N_O/2$ is the two-sided power spectral density of the additive white Gaussian noise. (See Appendix A5 for more details of the simulation techniques. Appendix A8 defines the notation used to describe the schemes for which results are given in Graph 5.2.1.) The accuracy of the results at a BER of 1 in 10^3 is of the order of ±0.2dB. The degradations in tolerance to noise that the system suffers at a BER of 1 in 10^3, in comparison with threshold detected QPSK and (64 vector) Viterbi-detected coded 8PSK using Code 3, are respectively, 6.2dB and 8.6dB. The error burst characteristics, in terms of the average number of bit errors per burst at various BERs are given in Table 5.2.1. The definition of an error burst is given in Appendix A5. The number of errors per burst at low BERs tends to an average of 5 bits per burst. This low value can be explained with reference to Figure 5.2.1. For example, a false estimate of $c_i'(3)$ will affect the detected symbols $q_i'(2)$,$q_{i+2}'(1)$,

$q'_{i+3}(1)$ and $q'_{i+3}(2)$. Errors in $c'_i(1)$ and $c'_i(2)$ due to isolated single noise spikes affect the output symbols over a shorter period of time. Therefore if all errors in the $\{c'_i\}$ were spaced far enough apart, the average number of bit errors per burst must be less than 8, since such errors can only affect the $\{q'_i\}$ over a maximum of four consecutive symbol intervals.

Tests have been conducted to find the average error rate in the $\{p'_i\}$, at various values of E_b/N_O. The results are given in Table 5.2.2. The thresholds used to measure the number of so-called boundary crosses, given a symbol error, are shown in Figure 5.2.2, which includes an example of a double boundary cross. Clearly the error rate in the $\{p'_i\}$, even at quite high signal to noise ratios, is very high. For example at the signal to noise ratio, E_b/N_O=5.5dB, the bit error rate for Viterbi detection for coded 8PSK using Code 1, is 2 in 10^4, at which point nearly 15% of the $\{p'_i\}$ are in error. From Table 5.2.2 it is also clear that the vast majority of the errors, in all cases, are single boundary crosses, (that is $c'_i=|c_i \pm 1|$MODULO-8). Therefore this is the error-type on which a detector should place most of its efforts.

In conclusion, the inverse coder detection scheme produces detected values wherein the lengths of the error bursts are minimised. Unfortunately the performance of the detector is inferior, even in comparison with the pseudo-nonlinear equaliser of Section 5.1. Therefore it is unlikely that, at typical values of E_b/N_O, this scheme used as the initial detector in a two-stage detector as described earlier, would provide a low enough bit error rate to be of sufficient use to the second stage detector.

APPROXIMATE AVERAGE NUMBER OF BIT ERRORS PER BURST, AT GIVEN BER		
5×10^{-2}	1×10^{-2}	2×10^{-3}
6.6	5.4	5.1

TABLE 5.2.1: Error Burst Characteristics for Inverse Coder Detection of Coded 8PSK, Using Code 1

E_b/N_O (dB)	ERROR RATE IN THE $\{p_i'\}$	TOTAL NUMBER OF TRANSMITTED DATA SYMBOLS q_i	% OF ERRORS IN THE p_i', FOR GIVEN NUMBER OF BOUNDARY CROSSES			
			1	2	3	4
3	0.28	9×10^4	97	2.6	0.32	0.08
4.6	0.194	9×10^4	99.22	0.74	0.02	0.02
5.5	0.149	5×10^5	99.69	0.3	0.01	0
9.5	0.022	9×10^4	100	0	0	0
13.5	3×10^{-4}	4×10^5	100	0	0	0

TABLE 5.2.2: Error Statistics for the Estimated Values $\{p_i'\}$

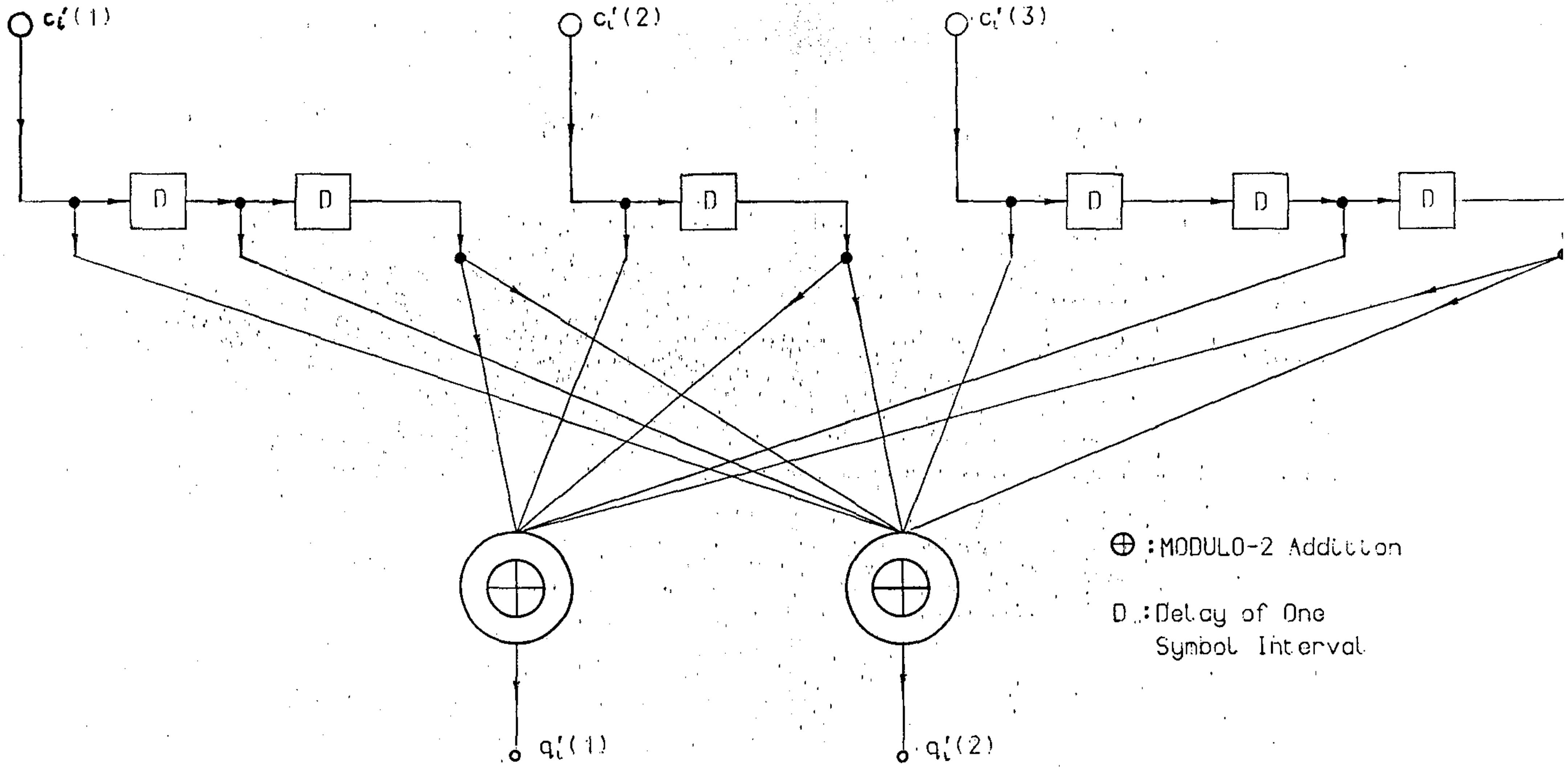


Figure 5.2.1 Inverse Coder Implementation for Code 1

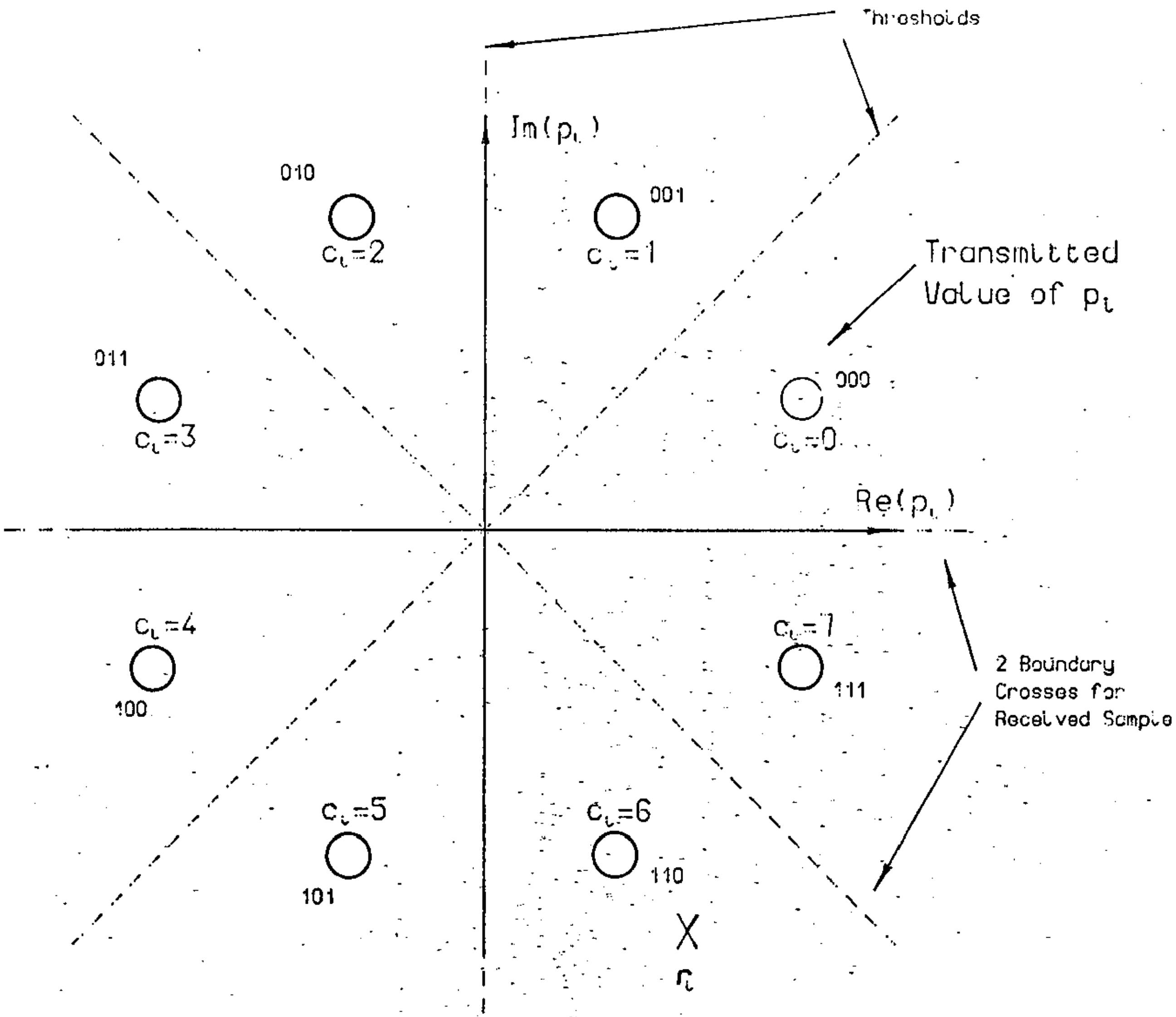
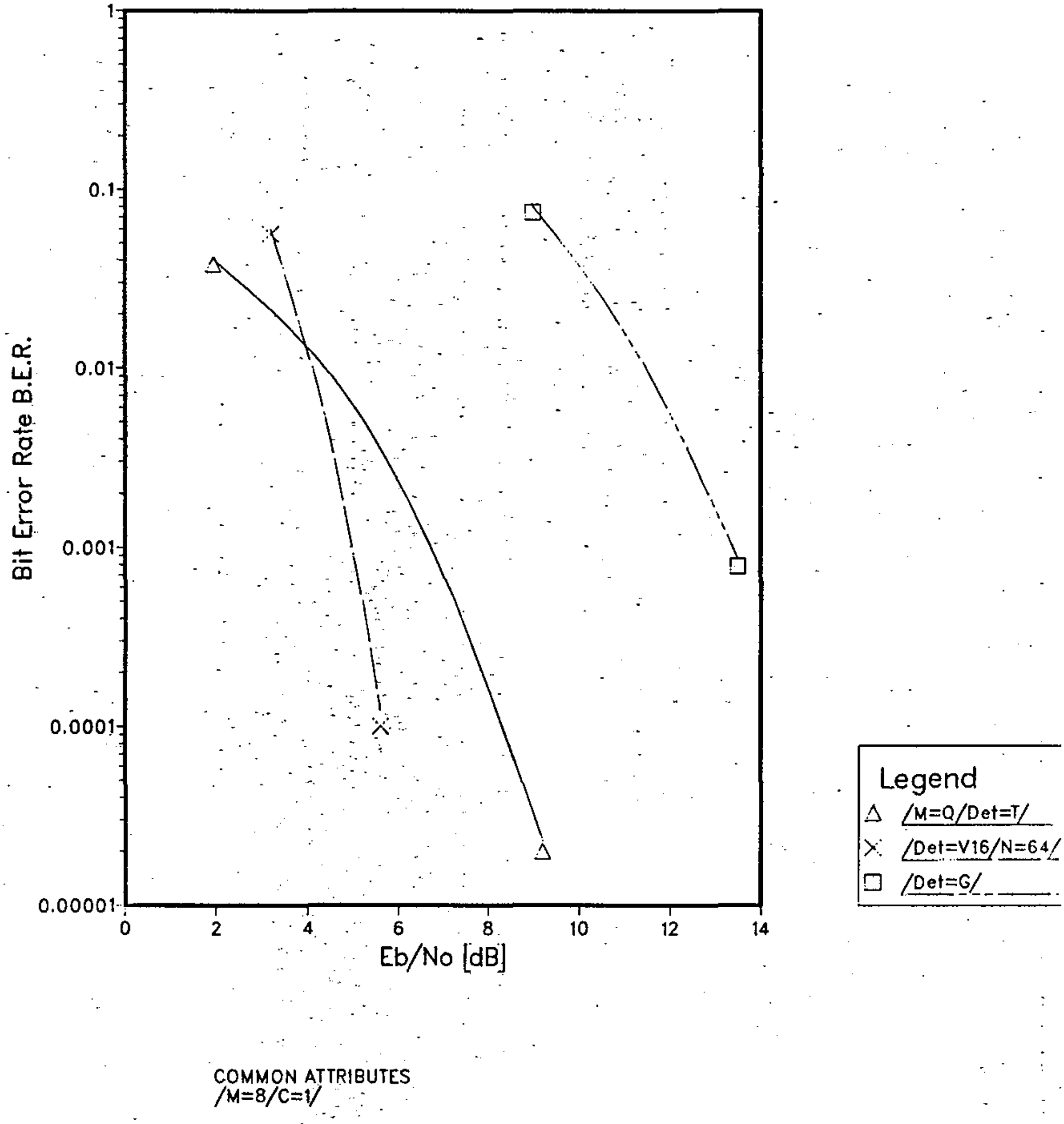


Figure 5.2.2 Thresholds used to Provide a Measure of The Number of Boundary Crosses, Given an Error in p_i'

Graph 5.2.1 Feedforward Inverse Coder Detection



5.3 STATE REDEFINITION TECHNIQUES

Following on from the state redefinition described for the pseudobinary Viterbi detector, (Section 4.3), this section introduces two state redefinition techniques which are based, respectively, on the symbol values involved in the original state definition, and on the original states themselves. For coded 8PSK modulation, the original definition of the state of a stored vector, is the particular combination of the values of the symbols $q'_{i-k+1}, q'_{i-k+2}, \ldots, q'_{i-1}$, in the vector at time t=iT. k is the constraint length of the code. Both approaches involve the regrouping of subsets of the set of all originally defined states into redefined states. The first approach is derived from the pseudobinary Viterbi detector, whereas the second approach is not a pseudobinary technique. It is to be noted that the resulting states do not form true Finite-State machines.[72]

The first approach uses a non-unique mapping of the four-level symbols which define the state in the original definition, onto binary (recoded) symbols. In this sense the scheme is similar to the pseudobinary Viterbi detector. The difference is that the technique maps two possible values of the four-level symbol onto one of the recoded symbols, and the other two possible values of the four-level symbol onto the other recoded symbol. Since the pseudobinary Viterbi detector's mapping is based on incremental costs, (Section 4.3), its mapping rule does not involve the same four-level symbol values every time. Given the mapping rule, the detector operates as does the pseudobinary Viterbi detector, allowing two expanded vectors per vector Q'_{i-1}, at time t=iT. The determination of the two expanded vectors to be discarded is explained later. As for the pseudobinary Viterbi detector, the number of stored vectors is reduced from 4^{k-1} to 2^{k-1}. This technique is called the First Approach state redefinition. Both schemes for coded 8PSK using Code 1, (4 stored vectors), and for coded 8PSK using Code 3, (8 stored vectors), were tested. The four-level symbol to binary symbol mapping rules are given in Tables 5.3.1 and 5.3.2, respectively. For the constraint length k=4 code, (Code 3), the mapping operates on the three symbols q'_{i-3}, q'_{i-2} and q'_{i-1} at time t=iT, whereas for the k=3 code, (Code 1), the mapping operates on the two symbols q'_{i-2} and q'_{i-1} at time t=iT. Clearly, in both cases, these are the symbols which define the state of a vector in the original definition.

The Second Approach state redefinition directly maps subsets of states as originally defined onto redefined states. In contrast to the schemes using the First Approach state redefinition, the detector, given the redefined states, operates as does the Viterbi detector of Section 3.2, not as the pseudobinary Viterbi detector of Section 4.3. Four second Approach State redefinitions were used, two for Code 1 and two for Code 3. Tables 5.3.3 to 5.3.6 define the redefinitions by listing the subsets of states as originally defined, regrouped into redefined states. The states as originally defined, within each redefined state, are given by the values of the four-level symbols $q'_{i-k+1}, q'_{i-k+2}, \ldots, q'_{i-1}$, where the left-most value is that of q'_{i-k+1} and the right-most value is that of q'_{i-1}, (see Tables 5.3.3 to 5.3.6).

These six state redefinitions have a number of important characteristics. Short sections of the code trellis diagrams for the two First Approach redefinitions are given in Figures 5.3.1 and 5.3.2. These diagrams are essentially graphs of the redefined state of a stored vector, (vertical axis), as it varies with time in symbol intervals, (horizontal axis). Only one symbol interval is included in these figures. The states at time t=iT are given by each combination of the recoded values of the vector elements $q'_{i-k+1}, q'_{i-k+2}, \ldots, q'_{i-1}$. Each line in the diagram is for one of the 2^k expanded vectors. For a given vector at time t=iT, the diagram gives the possible values of its (redefined) state at time t=(i+1)T. The two arrows on each line signify that each line is for two expanded vectors of an original vector, (that is, for two different values of q'_i). These lines with two arrows are called parallel transitions in Reference 19. Since the diagrams are for pseudobinary schemes, the two chosen expanded vectors for each vector at time t=iT are determined as follows. The costs of the two expanded vectors of each parallel transition are ranked, and the one with the lowest cost is chosen. In order to optimise performance where such parallel transitions occur in the code trellis diagram, it is essential that the values of the complex numbersp'_i for the two expanded vectors of one parallel transition, are as far apart as possible in the complex number plane.[19] (p'_i is a possible value of the received sample r_i, in the absence of noise. The procedure for determining p'_i was described in Section 5.1.) If the two values of p'_i are as far apart as possible, the two chosen expanded vectors derived from one vector at time t=iT are assumed to be those, of the four possible ones, with the lowest costs. For example if $c'_i=0$ for one expanded vector of a parallel transition, then the other expanded vector of the same parallel transition should be such that $c'_i=4$, (see Figure 2.5.4). The state redefinitions of Tables 5.3.1 and 5.3.2 are such that this is the case. Clark and Cain[19] note that parallel transitions in the code trellis diagram limit the maximum coding gain to 3dB, (given the arrangement just described for the values p_i'). It was as an attempt to develop state redefinitions with code trellis diagrams which do not have parallel transitions, that the Second Approach state redefinition technique was tried. In order to avoid parallel transitions in the code trellis diagram the following rule, concerning the regrouping of states as originally defined into redefined states, is applied. States as originally defined are grouped into redefined states, such that no two states as originally defined in a given redefined state have the same combination of the values of $q'_{i-k+1},q'_{i-k+2},\ldots,q'_{i-2}$. The state redefinition of Table 5.3.3 is effectively that for states as originally defined, for a code with constraint length k=2, whereas the code used, (Code 1), is a code where k=3. In such a regrouping 4^{k-2} redefined states are produced, each containing four states as originally defined and each of these four states as originally defined has a different value of the symbol q'_{i-k+1}, so that all possible values of q'_{i-k+1} are included in each redefined state. Such regroupings effectively delete q'_{i-k+1} from the sequence of symbols defining the state. In general, by including all combinations of the values of the symbols q'_{i-k+1},q'_{i-k+2}, $\ldots,q'_{i-k+j}$, in each redefined state, where $k \geq j+2$, the resulting states are those for a constraint length-(k-j) code, and there are 4^{k-1-j} states. This is a technique, usually termed reduced-state Viterbi detection, which is used on channels with intersymbol interference to reduce detector complexity.

Given the state redefinition, the detector operates in the following manner. (In the case of the First Approach detectors, this procedure takes place after choosing the two expanded vectors of each original vector, as described earlier.) For each redefined state, the detector chooses the lowest-cost expanded vector which has this redefined state. The calculation of the costs of the expanded vectors is fully explained in Chapters 3 and 4. It is clear that the procedure is exactly as that for the corresponding Viterbi or pseudobinary Viterbi detector, except that the redefined states are used.

Graphs 5.3.1 and 5.3.2, for Codes 1 and 3 respectively, give the results of computer simulation tests for schemes using the state definitions given in Tables 5.3.1 to 5.3.6. These are graphs of bit error rate (BER) as the signal to noise ratio, E_b/N_0 is varied. E_b is the average energy transmitted per data bit and $N_0/2$ is the two-sided power spectral density of the additive white Gaussian noise. (See Appendix A5 for more details of the simulation techniques. Appendix A8 gives the notation used to describe the variants of the schemes which were tested by computer simulation.) The accuracy of the results is of the order of ±0.6dB at bit error rates (BER) in the region of 1 in 10^3. This low accuracy is largely due to the long error bursts occurring in many cases. The error burst characteristics, in terms of the average number of bit errors per burst at various BERs, are given in Table 5.3.7. The definition of an error burst is given in Appendix A5. From Graph 5.3.1 it is clear that all the state redefinitions produce very poor detectors. The reduced-state Viterbi scheme, (Rec=1a), is better than the scheme using the Rec=1b state redefinition, and the pseudobinary scheme using the Rec=Pb1a state redefinition is the best overall. Table 5.3.7 shows that the number of errors per burst for all the schemes is very large, which is the same as for System 1 detection for Code 1, (see Table 4.1.1).

Graph 5.3.2 is the equivalent of Graph 5.3.1, for Code 3. In agreement with Table 4.1.3, the numbers of errors per burst in Table 5.3.7 for the schemes using Code 3, are considerably smaller than those for the schemes using Code 1. In particular the pseudobinary scheme where the Rec=Pb3 redefinition is used begins to gain over threshold detected QPSK below a BER of 1 in 10^3. Clearly, the technique of ensuring that the expanded vectors of parallel transitions have values of p_i' spaced as far apart as possible in the complex number plane, does in fact pay-off. Tests were conducted with Code 3 using First Approach State redefinitions which did not have this characteristic, at a few values of E_b/N_0. In all cases the tolerance to noise was considerably inferior.

From the results of Graphs 5.3.1 and 5.3.2 it is clear that these state redefinition techniques produce a very inferior performance for a relatively small reduction in complexity, compared with Viterbi detection. It is clear that reduced-state Viterbi detection for coded systems, (Reç=la), does not perform nearly as well as the same technique applied to channels with intersymbol interference.[64,65] This is because in the latter case the detector ignores only those components of the channel sampled impulse response which are negligible, compared with the main components. In a convolutionally coded system, all symbols involved in the original definition of a state have equal significance, and the removal of any one symbol from the definition of a state inevitably leads to a large performance degradation. Comparing these schemes with pseudobinary Viterbi detection, only the scheme using the Rec=Pb3 redefinition performs better. The advantage that the pseudobinary Viterbi detector has over most of these schemes, is that the state redefinition is based on the selection of those two expanded vectors of a given vector with the lowest costs, which at least involves a measure of the likelihood of each expanded vector. State redefinition techniques based on the possible values of the data symbols are not based upon the distances in unitary vector space between sequences of coded and mapped symbols $\{p_i'\}$, which in fact determine the likelihoods of the possible sequences. The exceptions are the pseudobinary First Approach state redefinitions, which consider this to the extent that the expanded vectors of parallel transitions have values of p_i' spaced as far apart as possible in the complex number plane. In conclusion, such state redefinitions are not viable techniques for reducing the complexity of the detector for coded 8PSK modulation.

4-LEVEL VALUE OF q_j'	RECODED BINARY VALUE OF q_j'
0	0
1	0
3	1
2	1

TABLE 5.3.1: State Redefinition Mapping Pb1a for the First Approach (Pseudobinary) Technique for Code 1

4-LEVEL VALUE OF q_j'	RECODED BINARY VALUE OF q_j'
0	0
1	1
3	0
2	1

TABLE 5.3.2: State Redefinition Mapping Pb3 for the First Approach (Pseudobinary) Technique for Code 3

REDEFINED STATE DESIGNATION	SUBSETS OF ORIGINAL STATES REGROUPED INTO A REDEFINED STATE	
0	[0 0]	[3 0]
	[1 0]	[2 0]
1	[0 1]	[3 1]
	[1 1]	[2 1]
2	[0 3]	[3 3]
	[1 3]	[2 3]
3	[0 2]	[3 2]
	[1 2]	[2 2]

TABLE 5.3.3: Non-Pseudobinary Second Approach State Redefinition 1a for Code 1

REDEFINED STATE DESIGNATION	SUBSETS OF ORIGINAL STATES REGROUPED INTO A REDEFINED STATE	
0	[0 0]	[3 3]
	[1 1]	[2 2]
1	[0 1]	[3 2]
	[1 3]	[2 0]
2	[0 3]	[3 0]
	[1 2]	[2 1]
3	[0 2]	[3 1]
	[1 0]	[2 3]

TABLE 5.3.4: Non-Pseudobinary Second Approach State Redefinition 1b for Code 1

REDEFINED STATE DESIGNATION	SUBSETS OF ORIGINAL STATES REGROUPED INTO A REDEFINED STATE
0	[0 0 0] [0 2 0] [1 3 0] [0 1 0] [1 0 0] [1 2 0] [0 3 0] [1 1 0]
1	[0 0 1] [0 2 1] [1 3 1] [0 1 1] [1 0 1] [1 2 1] [0 3 1] [1 1 1]
2	[0 0 3] [0 2 3] [1 3 3] [0 1 3] [1 0 3] [1 2 3] [0 3 3] [1 1 3]
3	[0 0 2] [0 2 2] [1 3 2] [0 1 2] [1 0 2] [1 2 2] [0 3 2] [1 1 2]
4	[3 0 0] [3 2 0] [2 3 0] [3 1 0] [2 0 0] [2 2 0] [3 3 0] [2 1 0]
5	[3 0 1] [3 2 1] [2 3 1] [3 1 1] [2 0 1] [2 2 1] [3 3 1] [2 1 1]
6	[3 0 3] [3 2 3] [2 3 3] [3 1 3] [2 0 3] [2 2 3] [3 3 3] [2 1 3]
7	[3 0 2] [3 2 2] [2 3 2] [3 1 2] [2 0 2] [2 2 2] [3 3 2] [2 1 2]

TABLE 5.3.5: Non-Pseudobinary Second Approach State Redefinition 3a for Code 3

REDEFINED STATE DEFINITION	SUBSETS OF ORIGINAL STATES REGROUPED INTO A REDEFINED STATE
0	[0 0 0] [2 0 0] [3 1 0] [1 0 0] [0 1 0] [2 1 0] [3 0 0] [1 1 0]
1	[0 0 1] [2 0 1] [3 1 1] [1 0 1] [0 1 1] [2 1 1] [3 0 1] [1 1 1]
2	[0 0 3] [2 0 3] [3 1 3] [1 0 3] [0 1 3] [2 1 3] [3 0 3] [1 1 3]
3	[0 0 2] [2 0 2] [3 1 2] [1 0 2] [0 1 2] [2 1 2] [3 0 2] [1 1 2]
4	[0 3 0] [2 3 0] [3 2 0] [1 3 0] [0 2 0] [2 2 0] [3 3 0] [1 2 0]
5	[0 3 1] [2 2 1] [3 2 1] [1 3 1] [0 2 1] [2 2 1] [3 3 1] [1 2 1]
6	[0 3 3] [2 3 3] [3 2 3] [1 3 3] [0 2 3] [2 2 3] [3 3 3] [1 2 3]
7	[0 3 2] [2 3 2] [3 2 2] [1 3 2] [0 2 2] [2 2 2] [3 3 2] [1 2 2]

TABLE 5.3.6: Non-Pseudobinary Second Approach State Redefinition 3b for Code 3

SCHEME	APPROXIMATE AVERAGE NUMBER OF BIT ERRORS PER BURST, AT GIVEN BER			
	3×10^{-2}	1×10^{-2}	3×10^{-3}	1×10^{-3}
/C=1/Det=V4/Rec=Pb1a/	280	360	-	400
/C=3/Det=V8/Rec=Pb3/	50	46	40	30
/C=1/Det=V4/Rec=1a/	300	310	580	380
/C=1/Det=V4/Rec=1b/	970	>1000	>1000	>1000
/C=3/Det=V8/Rec=3a/	56	60	43	35
/C=3/Det=V8/Rec=3b/	75	75	71	69

TABLE 5.3.7: Error Burst Characteristics for the Schemes Using State Redefinition Techniques for Coded 8PSK, Using Codes 1 and 3

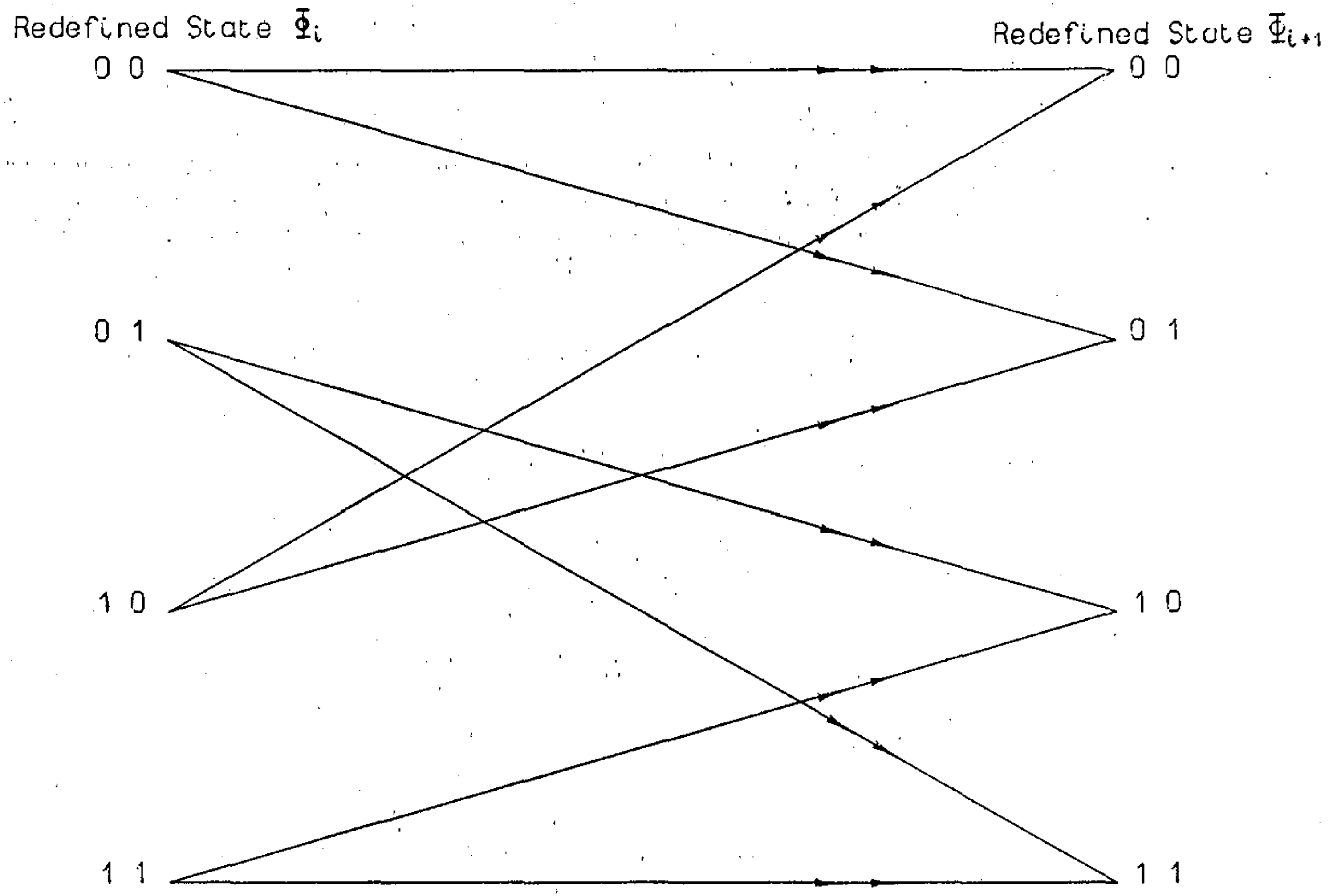


Figure 5.3.1 A Section of The Redefined-State Code Trellis Diagram for System Pb1a (Code 1)

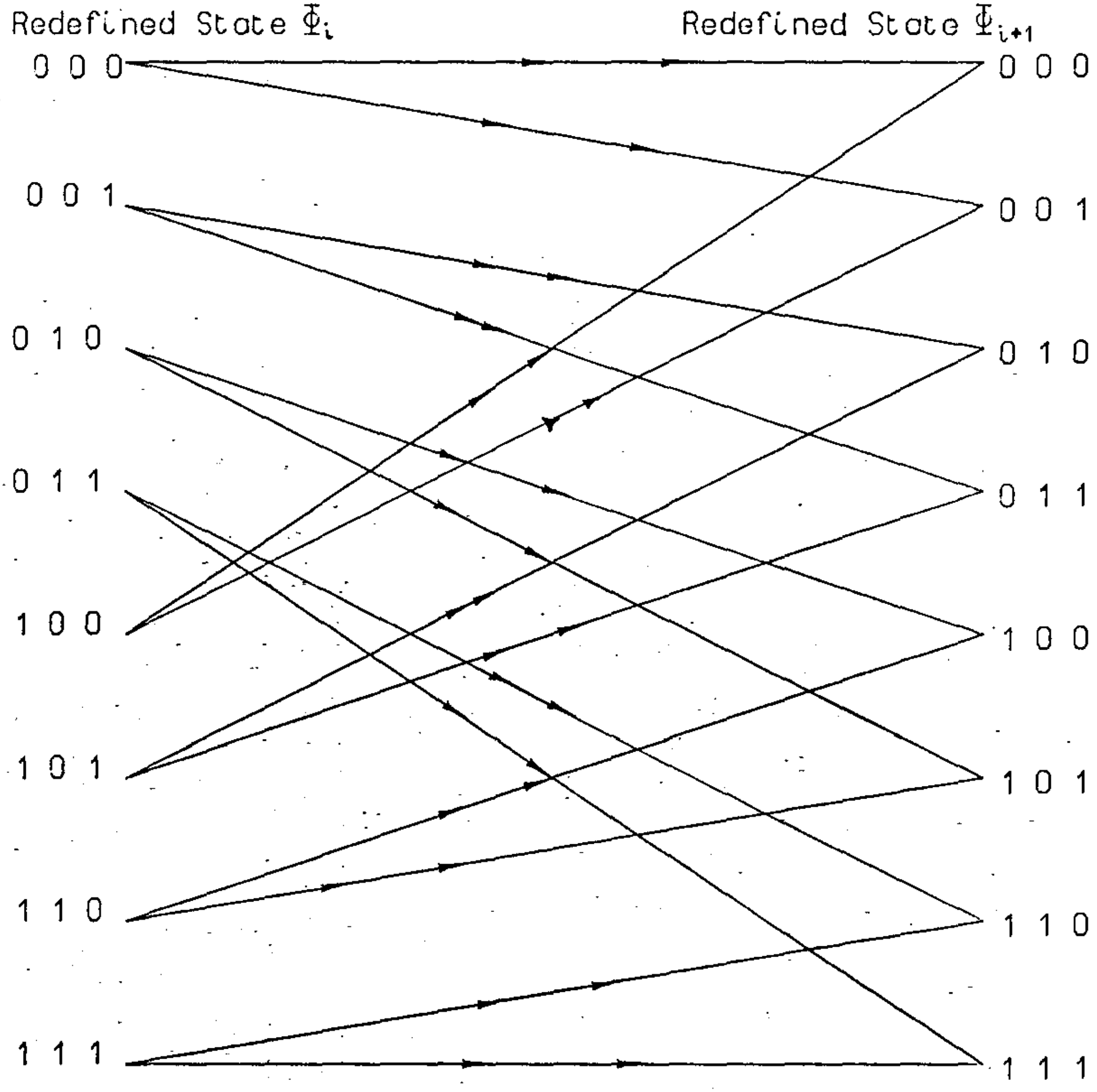
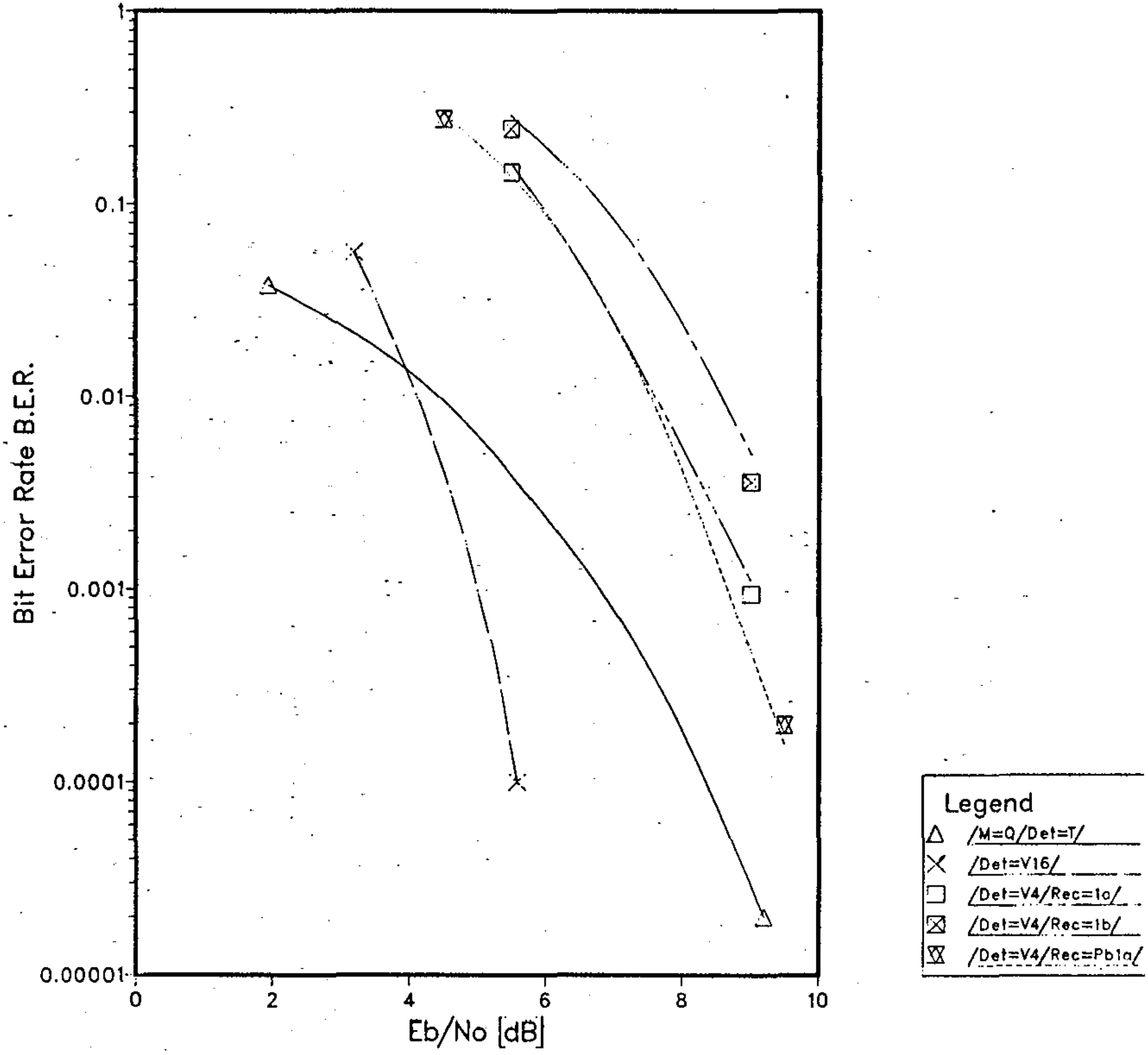


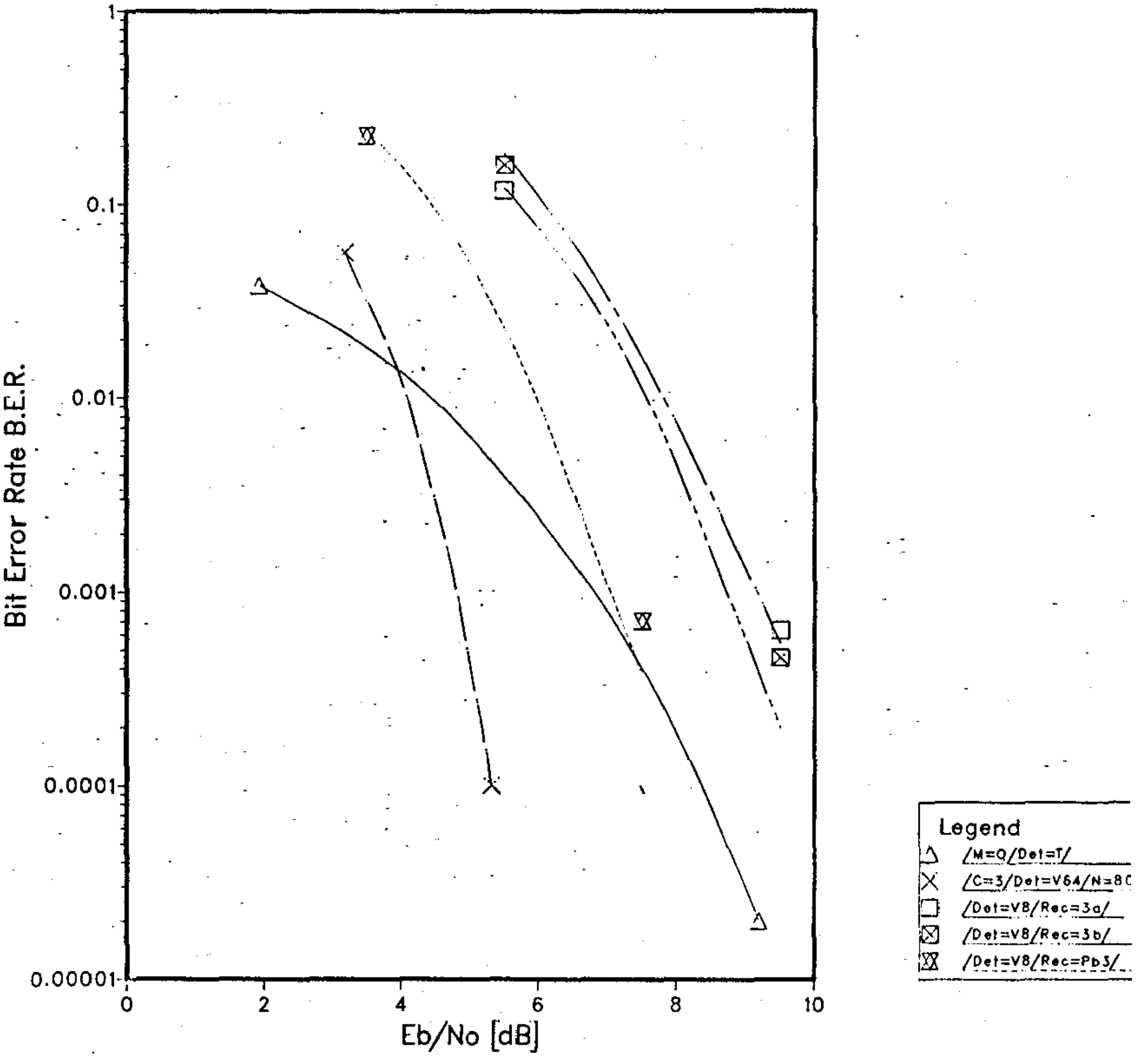
Figure 5.3.2 A Section of The Redefined-State Code Trellis Diagram for System Pb3 (Code 3)

Graph 5.3.1 Redefined State Viterbi Detection. Code 1



COMMON ATTRIBUTES
/M=8/C=1/N=64/

Graph 5.3.2 Redefined–State Viterbi Detection. Code 3



COMMON ATTRIBUTES
/M=8/C=3/N=80/

5.4 SOFT-DECISION SYNDROME DECODING

Syndrome decoders were among the first considered for application to convolutionally encoded systems, simply because at the time Maximum Likelihood detection was considered too complex.[19] In Section 5.2 it was noted that the inverse coder, (a feedforward filter), could be used as the first part of a system using two detectors, (a dual-detector system). The inverse coder would produce a sequence of initial (soft) detected data, and this would be improved upon by a more sophisticated second-stage detector. It was hoped that this system would not have some of the error burst problems of the systems of Chapter 4, and would require a second-stage detector which is relatively simple compared with the Viterbi Algorithm detector. Also, in Section 5.2 it was shown that most errors in the received samples $\{r_i\}$ are where r_i is nearer to a point p_i' which is either directly clockwise or anticlockwise from the transmitted point p_i, in the complex number plane. (Such errors are called single boundary crosses as in Section 5.2.) It was felt that a syndrome decoder could exploit this latter point. This is discussed more fully later.

Initial dual-detector studies concentrated on the inverse coder of Section 5.2 producing a sequence of soft-detected symbols. The second detector would then permutate reasonably short blocks of the inverse coder output sequence. The cost of each block of permutated data symbol values, $q'_{i-\ell+1}, q'_{i-\ell+2}, \ldots, q'_i$, is determined by calculating the squared unitary distance between the received samples $r_{i-\ell+1}$, $r_{i-\ell+2}, \ldots, r_i$, and the complex numbers $\{p_j'\}$ which result from coding the $\{q_j'\}$ above, and mapping the resulting code symbols $\{c_j'\}$ onto the complex number plane. Figure 2.5.4 defines the mapping. The unitary distance measure is defined in Appendix A7. These costs include that for the non-permuted block of possible data symbol values. The resulting costs are ranked and the value of $q'_{i-\ell+1}$ in the permutated block of possible data symbol values with the lowest cost, is taken to be the (hard) detected data symbol. The use of the inverse coder to produce a sequence of soft decisions should reduce the number of permutations which need to be considered, as long as the bit error rate in the inverse coder output sequence is not too high. This is because in such a case, the sequence at the output of the inverse coder will include only a few errors in the $\{q'_i\}$. For the calculated costs to be reasonable measures of the likelihoods of the permutated blocks of possible data symbol values, the blocks must be reasonably long. For a given block of possible data symbol values with ℓ components at the output of the inverse coder, the total number of permutated blocks of possible data symbol values to be considered must be reduced from the maximum of 4^{ℓ}, which would be prohibitive for even quite small values of ℓ. Attempts to restrict the number of such permutated blocks by allowing only a few changes in the blocks of possible data symbol values failed, because such minor changes could produce quite major changes in the complex numbers $\{p'_i\}$. Conversely, a block of permutated data symbol values where quite a few of the component values are changed, may produce only a few changes in the complex numbers $\{p'_i\}$. In addition the poor performance of the inverse coder, (a bit error rate approaching 0.5 at typical signal to noise ratios), drastically limits the performance of the second detector.

The study then considered the use of table look-up syndrome decoding as a basis for the detector.[19] The idea is very similar to that for the scheme just described, in that an initial selection of the most likely data symbols is made, before any intensive processing is undertaken.

In the former case this initial selection is provided by the sequence of possible data symbol values at the output of the inverse coder. In this case a sequence of syndrome symbols, (see Appendix A6), is used to select a list of possible changes to the code symbols $\{c_i'\}$. (The stored sequences of symbols which produce these changes are called error vectors.) The syndrome symbols are binary-valued. In addition, the fact that most errors in the received samples are single boundary crosses, (Section 5.2), means that a considerable number of possible error vectors can be deleted from the list for a particular syndrome sequence. A single boundary cross is where the point p_i' which is nearest to the received sample r_i, is either directly clockwise or directly anticlockwise, from the point p_i actually transmitted, (see Figure 2.5.4). Also, for reasonably high signal to noise ratios, there are occasions when errors in the received samples become more seldom. In such low-noise periods the Viterbi Algorithm, which performs the same operations whatever the noise level is, will clearly perform little better than much simpler detectors, such as the inverse coder. Therefore a detector which adapts the amount of processing undertaken, to suit the noise level, could be an advantage. A table look-up syndrome decoder could do this to the following extent. When the received samples over a period of time are the same as those transmitted, the syndrome sequence will be all-zero, indicating that no errors in the received samples have occurred.[19] In such cases the inverse coder can be used as the detector as in Section 5.2. Since the processing load for such a system varies from symbol interval to symbol interval, buffer stores must be provided to store received samples $\{r_i\}$ at the input, and detected symbols $\{q_i'\}$ at the output, of the detector. This added complexity is probably not compensated for by an equivalent reduction in the detector's complexity since, from Table 5.2.2, errors in the received samples are quite frequent even at reasonably high signal to noise ratios.

A short description of the relevant theory is now given, followed by a description of the system as implemented.

The notation of Appendix A6 will be used. This notation differs from that used in the other chapters of this thesis, and in the remainder of Chapter 5. Sequences of binary-valued symbols (having the positive values O or 1), are presented as polynomials in the delay operator D. The coder is described as a finite matrix whose elements are polynomials in the delay operator D, as is the circuit which produces the sequence of syndrome symbols, (called the syndrome-former). This change in notation simplifies the description of the detector. The syndrome-former $H^T(D)$, which generates the syndrome sequence, is by definition the null-space of the code generated by the coder $G(D)$[77], and is such that

$$G(D)H^T(D) = 0 \qquad (5.4.1)$$

For Code 1,

$$G(D) = \begin{bmatrix} D & 1+D+D^2 & 0 \\ 1+D^2 & D^2 & D \end{bmatrix} \qquad (5.4.2)$$

$H^T(D)$ is not unique as noted in Appendix A6. Equation 5.4.1 is used to generate a syndrome-former $H^T(D)$ for Code 1.

$$H^T(D) = [D+D^2+D^3, D^2, 1+D+D^4] \qquad (5.4.3)$$

The operation of the detector is as follows. The possible value of received sample r_j in the absence of noise, p_j', which is nearest to

r_j in the complex number plane, is found. p'_j is mapped onto the vector of binary code symbols $[c''_j(1),c''_j(2),c''_j(3)]$. (This mapping is the inverse of the mapping which at the transmitter converts the vector of code symbols $[c_j(1),c_j(2),c_j(3)]$ onto the complex number p_j, given in Figure 2.5.4.) The sequences of these code symbols are given by the vector of polynomials in the delay operator D, $C''(D)=[C''_1(D),C''_2(D),C''_3(D)]$ where $C''_\ell(D) = c''_1(\ell)+c''_2(\ell)D+\ldots+c''_i(\ell)D^{i-1}$, at time $t=iT$. The sequence of code symbols given by the vector $C''(D)$ may not be one that can be generated by the coder. This is because noise may change some of the values of the transmitted complex numbers $\{p_i\}$, such that some of the values of the binary code symbols $\{c''_j(\ell)\}$ are not the same as the corresponding values at the transmitter. These changes in the values of the binary code symbols are given by the three-component vector of polynomials $E(D)=[E_1(D),E_2(D),E_3(D)]$ where $E_\ell(D)=e_1(\ell)+e_2(\ell)D+\ldots+e_i(\ell)D^{i-1}$ and $e_j(\ell)$ is binary-valued. The code generated at the transmitter is $C(D)=[C_1(D),C_2(D),C_3(D)]$ where $C_\ell(D)=c_1(\ell)+c_2(\ell)D+\ldots+c_i(\ell)D^{i-1}$. $C(D)$ and $C''(D)$ are related by Equation 5.4.4

$$C''(D) = C(D) \oplus E(D) \qquad (5.4.4)$$

where $\oplus$ denotes MODULO-2 addition. The task of the detector is simply to determine $E(D)$, since if $E(D)$ is known, $C(D)$ can be determined. To this end, the detector uses the syndrome-former $H^T(D)$ to determine the sequence of binary-valued syndrome symbols, $\beta(D)$, from the three-component vector $C''(D)$.

$$\beta(D) = C''(D)H^T(D) \qquad (5.4.5)$$

In Appendix A6, it is shown that $\beta(D)$ is also given by

$$\beta(D) = E(D)H^T(D) \qquad (5.4.6)$$

$E(D)$ cannot be uniquely determined from Equation (5.4.6), given $\beta(D)$. Equation (5.4.6) defines a set of possible vectors $\{E(D)\}$ given $\beta(D)$. The detector's task is to produce an estimate of $E(D)$. From this, Equation (5.4.4) can be used to produce an estimate of $C(D)$, called $C'(D)$, which is hopefully the same as $C(D)$. The inverse coder $G^{-1}(D)$ is then used to produce the two-component vector of polynomials $Q'(D)$.

$$Q'(D) = C'(D)G^{-1}(D) \qquad (5.4.7)$$

$Q'D() = [Q_1'(D), Q_2'(D)]$ where $Q_\ell'(D) = q_1'(\ell) + q_2'(\ell)D + \ldots + q_i'(\ell)D^{i-1}$. The $\{q_j'(\ell)\}$ have the possible values 0 or 1. The output of the inverse coder at time $t = iT$ is the two-component vector $[q_j'(1), q_j'(2)]$. This is uniquely related to the four-level detected data symbol q_j' by the Gray code mapping of Table 2.1.1. Clearly it would be possible to pass $C''(D)$ directly through the inverse coder, and then convert the resultant two-component vector of polynomials in D into detected data which are hopefully equivalent to the transmitted data. It was decided to do the correction before the inverse coder, converting $C''(D)$ into $C'(D)$, since in comparing possible sequences in terms of costs as described later, (soft-decision detection), a method which permutates the inverse coder output sequence requires a coder at the receiver, in order to determine the code sequences for the permutated data sequences. The block diagram of the implementation is given in Figure 5.4.1. Short blocks of code symbols are permutated. Such a block is defined as a vector $[C_1''(D), C_2''(D), C_3''(D)]$ as before where the vector's elements are now taken to be the truncated polynomials, $C_\ell''(D) = c''_{i-\ell_e+1}(\ell)D^{i-\ell_e} + \ldots + c_i''(\ell)D^{i-1}$,

where ℓ_e is an integer. The stored syndrome symbols provide the address of a look-up table which consists of a list of blocks of possible error symbols. Such a block, called an error vector, is the vector $[E_1(D),E_2(D),E_3(D)]$ where the vector's elements are now the truncated polynomials, $E_\ell(D)=e_{i-\ell_e+1}(\ell)D^{i-\ell_e}+\ldots+e_i(\ell)D^{i-1}$. The error vectors are stored in order of likelihood (see later). The vector $[C''_1(D),C''_2(D),C''_3(D)]$ is permutated by adding, MODULO-2, $[C''_1(D),C''_2(D),C''_3(D$ to one of the stored error vectors $[E_1(D),E_2(D),E_3(D)]$.
The result of such an addition is called a permutation. The costs of the resulting permutations are determined using the stored incremental cost look-up tables. An incremental cost is the squared unitary distance between a received sample r_j, and a possible value of r_j in the absence of noise. There are eight such possible values of r_j, (see Table 2.5.4). Appendix A7 defines the unitary distance measure. The detector determines the permutation with the lowest cost. The values of the binary symbols $e_j(1)$,$e_j(2)$, and $e_j(3)$ in the error vector which produces the lowest-cost permutation are added to the corresponding symbols $c''_j(1)$,$c''_j(2)$, and $c''_j(3)$, to give the corrected code symbols $c'_j(1)$,$c'_j(2)$ and $c'_j(3)$. With correct detection $c'_j(\ell)=c_j(\ell)$, for $\ell=1,2$ and 3. The value of j is detector-dependent and will be defined later. The corrected code symbols are fed to the inverse coder, whose output is the detected data sequence. The three symbols $e_j(1)$,$e_j(2)$ and $e_j(3)$ are called the correction symbols.

The error vector tables are produced by one of two methods, the latter method being used in practice. The first method uses the following equation to estimate the likelihood, (probability), of various error vectors.

$$L = [\prod_{j=1}^{n_e} \rho_s \rho_O(i_j)](1-\rho_s)^{\ell_e - n_e} \quad , \text{ for } 1 \leq n_e \leq \ell_e \qquad (5.4.8)$$

ρ_s is the error rate in the complex numbers $\{p_i'\}$ at a given value of E_b/N_O, $\rho_O(i_j)$ is the proportion of all errors in the $\{p_i'\}$ which are single (i_j=1), double (i_j=2), triple(i_j=3), or quadruple (i_j=4), boundary crosses at the same value of E_b/N_O, (see Section 5.2). ℓ_e is as defined earlier and n_e is the number of the $\{p_i'\}$ which are changed, given the error vector. E_b/N_O is the signal to noise ratio where E_b is the average energy transmitted per data bit, and $N_O/2$ is the two-sided power spectral density of the additive white Gaussian noise. Equation 5.4.8 assumes that errors in the $\{p_i'\}$ are independent. The values of ρ_s and $\rho_O(i_j)$ are taken from Table 5.2.2 at E_b/N_O=5.5dB. For a given combination of the syndrome symbols, the error vectors are listed in order of their values of L, that with the highest value of L at the top of the list. The higher L is, the more likely the error vector is. A minimum value of L is given to restrict the size of the look-up tables. This method of generating the error vector tables is optimal, under the assumed conditions. The problem is, that as ℓ_e increases, the computational effort to produce the tables becomes prohibitive. Since most errors in the $\{p_i\}$ consist of single boundary crosses, (see Table 5.2.2), the following technique for producing the look-up tables was used in practice. Only error vectors which produce changes in the vector $[C_1''(D),C_2''(D),C_3''(D)]$ such that the changed values of the $\{p_i'\}$, (see earlier), are directly clockwise or directly anti-clockwise from the unchanged values of the $\{p_i'\}$, are included in the look-up tables. Such changes to the $\{p_i'\}$ were called single boundary crosses in Section 5.2. L is now redefined as the number of the $\{p_i'\}$ which are changed, because of the changes in the vector $[C_1''(D), C_2''(D), C_3''(D)]$. (L is now equal to n_e in Equation 5.4.8). Computer simulation tests showed that the two methods of producing the look-up tables yielded similar performance results.

A problem occurs in that the combination of the values of, for example $e_j(1), e_j(2)$, and $e_j(3)$, which produce a single boundary cross in the complex number p_j', are dependent on the values of $c_j''(1), c_j''(2)$ and $c_j''(3)$. The look-up tables must store all combinations of the values of $e_j(1), e_j(2)$ and $e_j(3)$ which produce a single boundary cross in the value of p_j'. Clearly this requires extra storage capacity. A test is required, given $c_j''(1), c_j''(2)$, and $c_j''(3)$, to find the values of $e_j(1)$, $e_j(2)$ and $e_j(3)$, which produce a single boundary cross in the value of p_j'. This adds to the complexity of the detector.

Two different detectors are considered. The first is a syndrome resetting detector[19] and the number of symbols of the syndrome sequence which are stored, L_s, is the same as ℓ_e. Once the correction symbols $e_j(1), e_j(2)$ and $e_j(3)$ have been chosen a sequence of ℓ_e binary symbols is added, MODULO-2, to the stored syndrome symbols. This operation is called syndrome resetting. This sequence of binary symbols constitute the syndrome symbols which would be stored in the syndrome register if the change in C"(D) caused by $e_j(1), e_j(2)$, and $e_j(3)$, is the only error in C"(D) which affects the syndrome symbols presently stored. Clearly the change to C"(D) is assumed to be correct. Syndrome resetting removes the effects of corrected errors in the stored syndrome symbols. For this detector, the value of j for the correction symbols $e_j(1), e_j(2)$, and $e_j(3)$, is $(i-\ell_e+1)$. Given correct syndrome resetting, (which occurs if the changes made to C"(D) are correct), the resultant syndrome symbols in the syndrome register, are those for the case where the $\{e_j(1)\}$,$\{e_j(2)\}$, and $\{e_j(3)\}$, for $j<(i-\ell_e+1)$, are all zero. The other scheme is called definite decoding[19] and involves no feedback of possible syndrome symbols. The stored error vectors have 4 more symbols than does the syndrome sequence, $(\ell_e=L_s+4)$. This increase in ℓ_e is because, in contrast to the case where syndrome resetting is used, the resultant syndrome symbols in the syndrome register after a change to C"(D), are not those for the case where the $\{e_j(1)\}$,$\{e_j(2)\}$, and $\{e_j(3)\}$ for $j<(i-\ell e+1)$ are all zero. In fact the syndrome symbols are still those for the $\{e_j(1)\}$,$\{e_j(2)\}$ and $\{e_j(3)\}$ which are required to correct C"(D) to give C(D). In this case the syndrome symbol at time $t=(i-L_s+1)T$ is a function of the values $\{e_j(1)\}$,$\{e_j(2)\}$, and $\{e_j(3)\}$, for $j=(i-L_s-3)$, $(i-L_s-2)$,...,$(i-L_s+1)$, from Equations 5.4.3 and 5.4.6. Therefore possible values of the correction symbols over this time period must be included in the error vectors, so that $\ell_e=L_s+4$. Clearly the look-up tables are considerably larger for definite decoding, than for the syndrome resetting case, (if L_s is fixed). The correction symbols $e_j(1)$,$e_j(2)$,$e_j(3)$, at time $t=iT$, are those where $j=i-L_s+1$.

The results of computer simulation tests on the syndrome resetting and definite decoding detectors are given in Graphs 5.4.1 and 5.4.2 respectively. These are graphs of bit error rate (BER) as the signal to noise ratio, E_b/N_0 is varied. (See Appendix A5 for more details of the simulation techniques. Appendix A8 gives the notation used to describe the variants of these detectors which were tested by computer simulation.) The accuracy of the results for Graphs 5.4.1 and 5.4.2 are respectively ±0.5dB and ±0.2dB, for BERs in the region of 3 in 10^3.

(The difference is due to the large error bursts for most of the systems of Graph 5.4.1).

Table 5.4.1 outlines the error burst characteristics for the syndrome resetting variants, in terms of the average number of bit errors per burst at various BERs. Appendix A5 defines an error burst. It is clear that none of the schemes both restricts the burst size and provides a performance which is degraded by less than 3dB in tolerance to noise, compared with the 16-vector Viterbi detector of Chapter 3. As L_s increases, the number of permutations to be considered, and the total required storage capacity, increase dramatically. This is outlined in Table 5.4.2. Clearly, the total storage requirement rises steeply with L_s and E_m. (E_m was called n_e earlier in this section.) Also, as both L_s and E_m rise, performance does not improve over that of the scheme where L_s=7 and E_m=4. From Table 5.4.1 most schemes have a very large number of errors per burst. It is useful to consider the schemes where L_s=9. For E_m=2 it is clear that the number of errors per burst is not much greater than that for Viterbi detection, (Table 3.2.2), whereas for E_m=4 the number of errors per burst is large, and increases as the noise level decreases. An analysis of the error bursts of the schemes where L_s=9, shows that the low number of errors per burst for low values of E_m is due to the fact that few syndrome resettings occur after an initial wrong correction is made. After an initial correction error, the binary symbols added to the contents of the syndrome register are incorrect so that the resulting contents of the syndrome register are incorrect. During the next symbol interval the syndrome symbols will address the wrong table of error vectors which may well yield further incorrect values for $e_j(1)$,$e_j(2)$ and $e_j(3)$.

Clearly this problem is liable to perpetuate. For low values of E_m, once an initial error has been made, few of the binary symbols added to the contents of the syndrome register in subsequent syndrome resetting operations are non-zero, either because there are no error vectors in the addressed table, or because the chosen error vector is such that $e_j(\ell)=0$, for $\ell=1,2,3$. In such cases the incorrect syndrome symbols in the register are shifted out very quickly, so that correct operation resumes. From References 19 and 79, long error bursts may be due to the following point. If L_s is too small there may not always be a path, (the result of a number of syndrome resettings over future symbol intervals), back to the all-zero syndrome sequence from any given syndrome sequence, even if no further errors in the $\{p_i'\}$ occur.[19] Clearly the number and lengths of such paths, (if they exist), also affect the likelihood of resuming correct decoding. The error vector table chosen in the next symbol interval after the syndrome has been reset, falsely or correctly, is always such that the sequence $C'(D)$ which is produced, is such that $C'(D)H^T(D)=0$, whatever error vector, $E(D)$, is chosen.[19] This means that there is no way of testing $C'(D)$ to see if a false syndrome resetting has occurred.

Table 5.4.3 gives the error burst characteristics for the definite decoding variants of Graph 5.4.2. The numbers of errors per burst are very similar to those of the inverse coder, (Table 5.2.1), but the tolerances to noise of these schemes are very inferior to those of the syndrome resetting variants. This is because, at a given value of L_s the syndrome resetting detector needs to consider far fewer possible error vectors than the definite decoding scheme, (since $\ell_e=L_s$ in the former case). The resulting sequences of complex numbers $\{p_i'\}$, (see earlier), are distanced quite far apart in the unitary vector space. This means that the costs to be ranked are reasonable measures of likelihood. On the other hand, in the definite decoding case, many more error vectors are stored per look-up table, and many produce permutations with similar costs. Therefore the cost in this case is less reasonable as a measure of likelihood. A particularly interesting case where for both schemes the cost measure for a given value of L_s is degraded, is the following. Consider the case outlined in Figure 5.4.2. (Here the code symbol $c_i=2^2c_i(1)+2^1c_i(2)+2^0c_i(3)$, as in Section 2.5.) A single boundary cross has occurred but the received sample r_i is closer to the point p_i where $c_i=4$, than to the point p_i where $c_i=6$. The former point is that value of p_i' which is chosen after a double boundary cross, (as described earlier), while the latter point is the transmitted value of p_i. The correct permutation is that where $c_i''=5$ is changed to $c_i'=6$, but since r_i is closer to the point where $c_i=4$, the detector may change $c_i''=5$ to $c_i'=4$. Clearly, this can only occur if the list of error vectors contains an error vector which can produce this latter change. As L_s is increased, the likelihood that the list of error vectors contains an error vector producing the latter change, decreases. In addition to the above, there may be occasions when the correct error vector is not included in the look-up table, because it involves double, triple, or quadruple, boundary crosses. Also, for a given value of L_s, there are a finite number of non-zero error vectors for the all-zero syndrome sequence, which these schemes clearly cannot correct. The latter problems are secondary to the first problem stated above, which in various analyses of computer simulation tests,has been the major cause of detection errors.

Clearly both schemes do not provide a viable alternative to the Viterbi detector, in terms of a trade-off between complexity and tolerance to noise. Clark and Cain note that soft-decision techniques are not usually used in look-up table schemes, because of the large look-up table size.[19] This has indeed been the case for both schemes. Syndrome decoding schemes which use the Viterbi algorithm have been put forward,[80-83] but in this case a large saving in complexity, over the Viterbi Algorithm detector, cannot be achieved.

The advantage of using a technique which adapts to the prevailing noise level, as noted earlier, is exploited in the system of Section 6.2.

SCHEME (Syndrome Resetting)	APPROXIMATE AVERAGE NUMBER OF BIT ERRORS PER BURST, AT GIVEN BER			
	6×10^{-2}	2×10^{-2}	3×10^{-3}	1×10^{-3}
$/L_s=7/E_m=3/$	261	438	-	-
$/L_s=7/E_m=4/$	119	106	117	120
$/L_s=7/E_m=5/$	255	348	290	250
$/L_s=8/E_m=4/$	270	300	-	-
$/L_s=9/E_m=2/$	12	10	-	-
$/L_s=9/E_m=3/$	25	22	20	-
$/L_s=9/E_m=4/$	171	169	180	-
$/L_s=10/E_m=5/$	80	75	62	65

TABLE 5.4.1: Error Burst Characteristics for Soft-Decision Table Look-Up Syndrome Decoding of Coded 8PSK, for Code 1, Using Syndrome Resetting

L_s	7	7	9	9	10	11
E_m [Only single boundary Crosses)	2	4	2	4	4	4
Total Number of Error Vectors	210	3990	351	12,825	20,685	31,713
Average Number of Error Vectors Per Table	6.6	125	0.7	25	20.2	15.5
Total Storage Requirement (k bits)	4.41	83.79	9.477	346.275	620.55	1046.79

TABLE 5.4.2: Outline of the Look-Up Table Storage Requirements for Various Configurations of the Syndrome Resetting Detector, for Coded 8PSK Using Code 1.

SCHEME (Definite Decoding)	APPROXIMATE AVERAGE NUMBER OF BIT ERRORS PER BURST, AT GIVEN BER		
	2×10^{-2}	1×10^{-2}	1×10^{-3}
$/L_s=5/E_m=1/$	6.2	5.8	-
$/L_s=7/E_m=2/$	6.5	5.3	-
$/L_s=8/E_m=2/$	6.6	5.9	5.0
$/L_s=8/E_m=4/$	6.4	5.4	5.1
$/L_s=9/E_m=3/$	6.5	5.7	5.3
$/L_s=9/E_m=4/$	6.4	5.8	5.4

TABLE 5.4.3: Error Burst Characteristics for Soft-Decision Table Look-up Syndrome Decoding of Coded 8PSK, for Code 1, Using a Definite Decoding Scheme

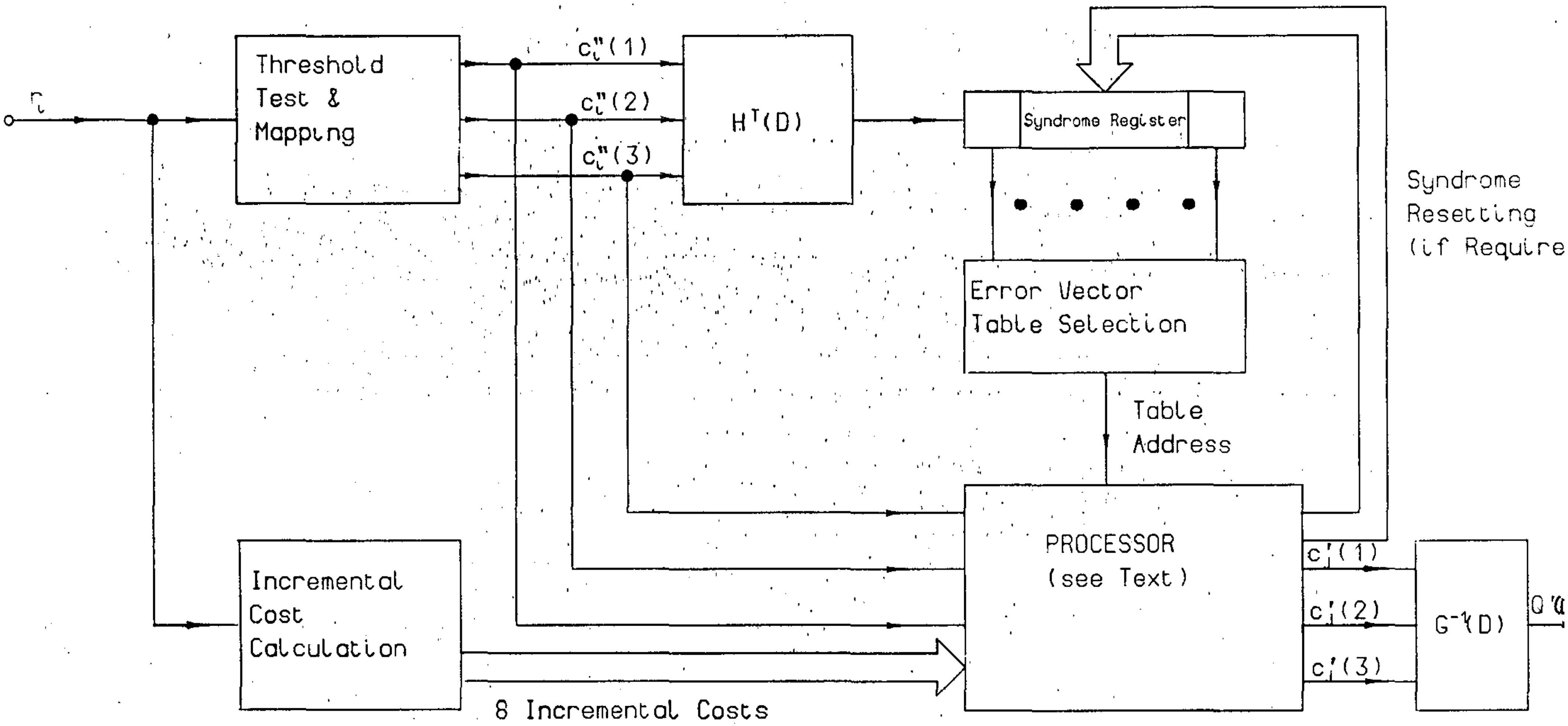


Figure 5.4.1 Soft-Decision Syndrome Detector Block Diagram

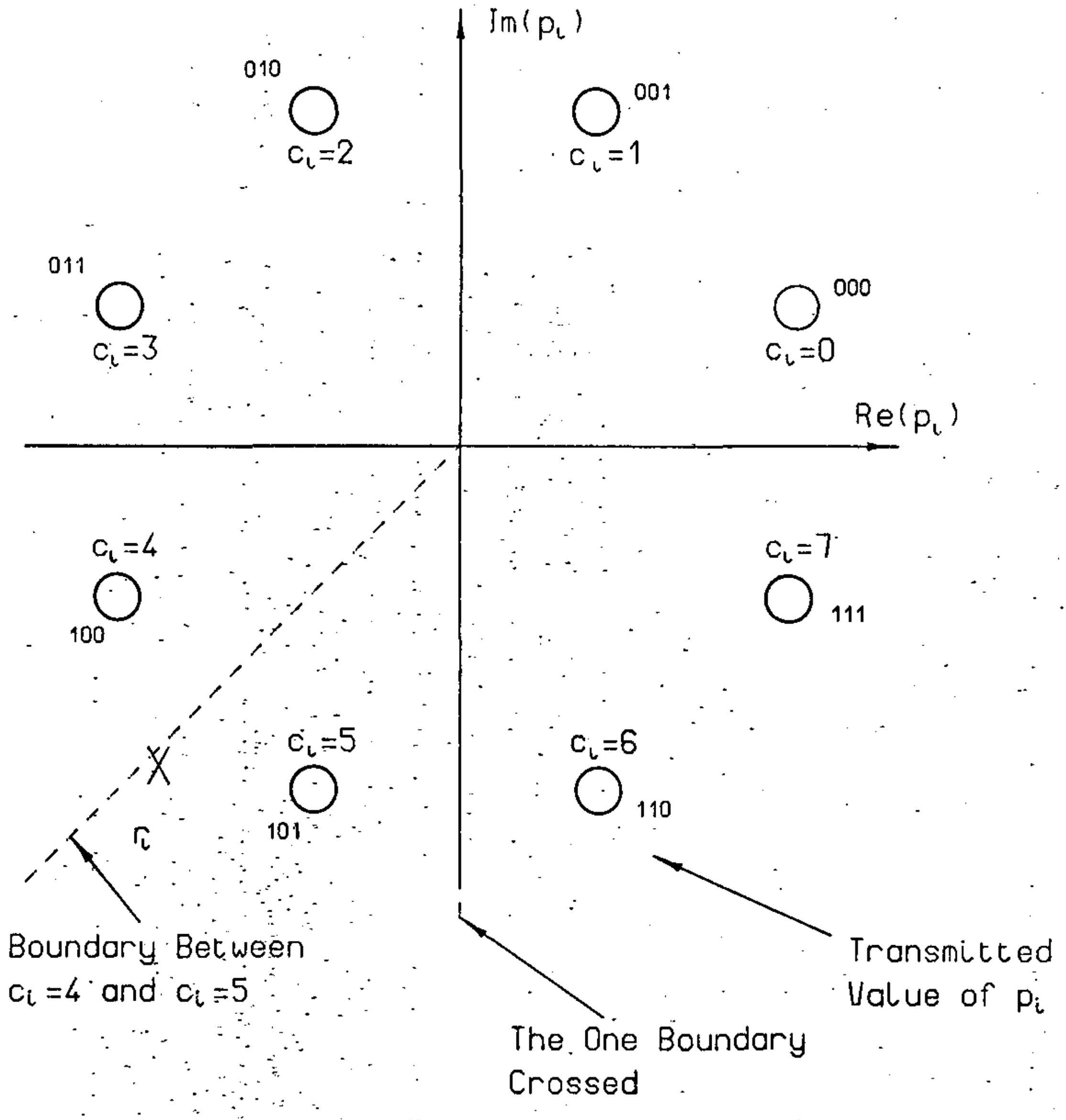
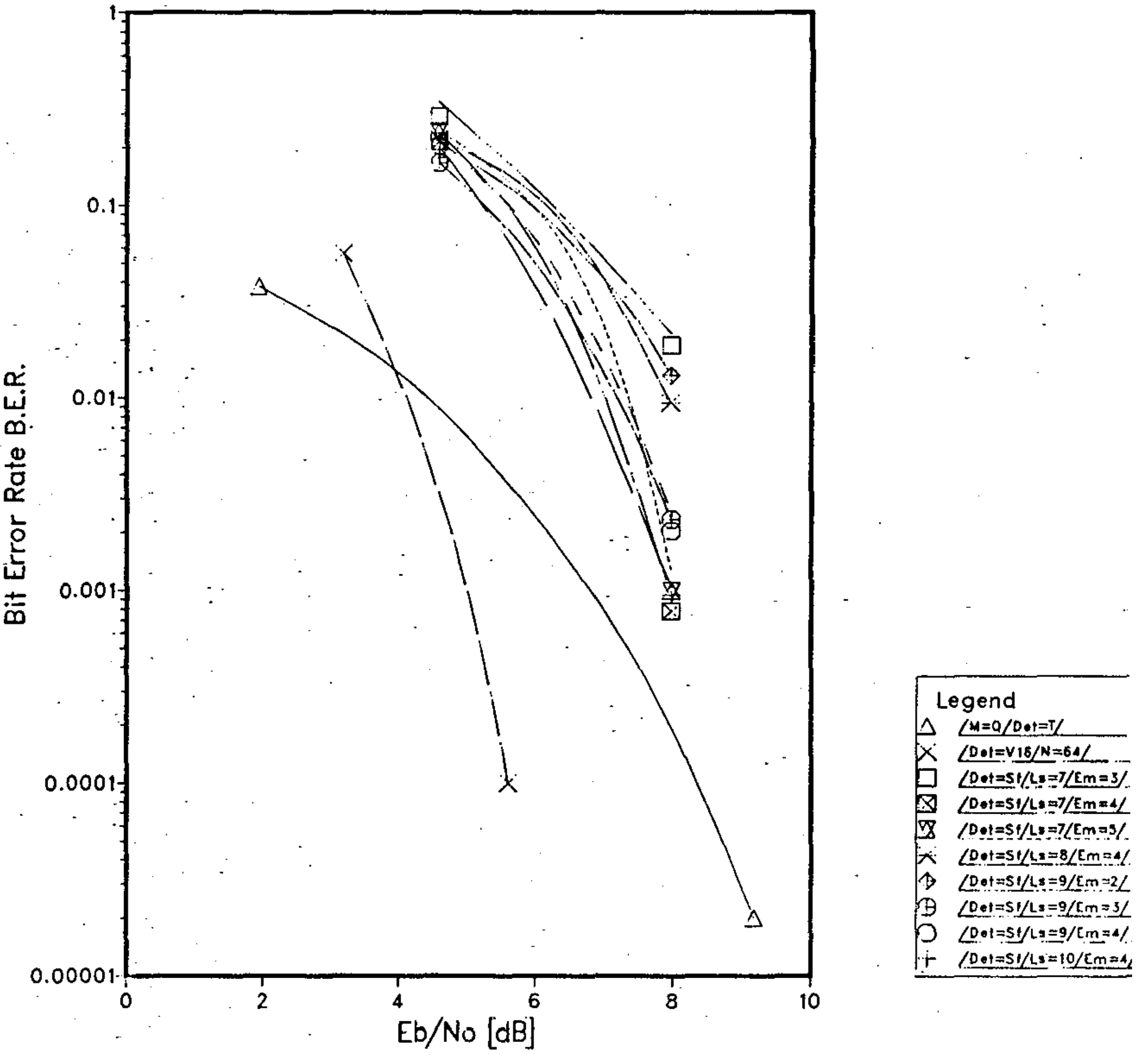


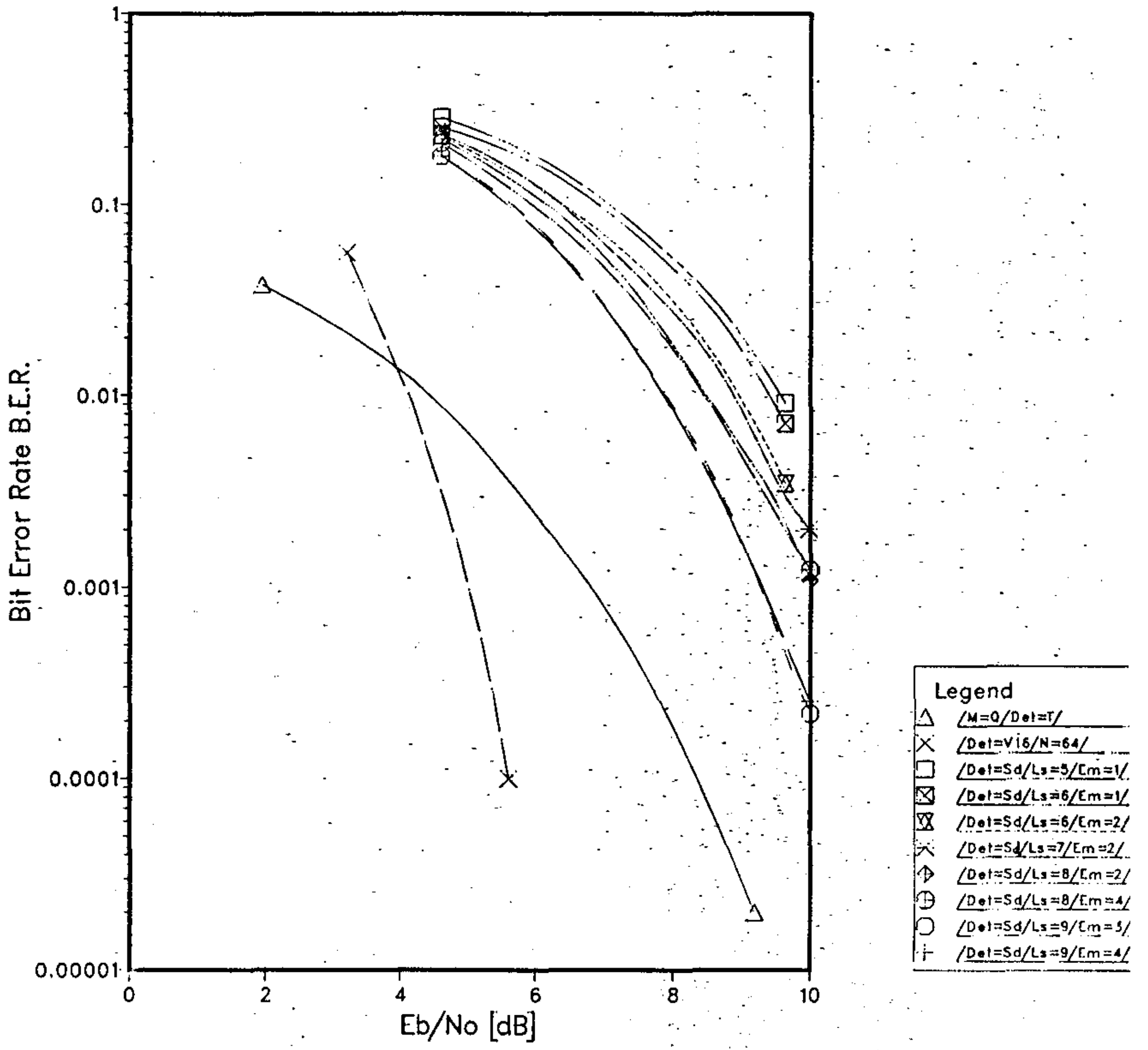
Figure 5.4.2 Complex Plane Representation of a Single Boundary Cross which Appears in The Costs as a Double Boundary Cross

Graph 5.4.1 Soft Decision Feedback Syndrome Decoding [Syndrome Resetting]



COMMON ATTRIBUTES
/M=8/C=1/

Graph 5.4.2 Soft Decision Feedforward Syndrome Decoding
[Definite Decoding]



COMMON ATTRIBUTES
/M=8/C=1/

CHAPTER 6

NOISE-ADAPTIVE (BUFFERED-DATA) DETECTION FOR CODED 8PSK

The two detectors to be described in this Chapter differ from the detectors of Chapters 3 and 4, in that the operations to be undertaken to produce one detected data symbol, vary from detected data symbol to detected data symbol. The price to be paid for this varying processing load is that buffer stores must be provided to store the received samples $\{r_i\}$ and the detected data symbols $\{q_i'\}$. When a suitable store for the $\{r_i\}$ is provided, the probability of losing samples because of buffer store overflow is low, at times when the number of operations per detected data symbol is high. The buffer store of detected data is used to ensure a continuous, constant-rate, stream of detected data symbols. The advantage of such schemes is that the number of operations performed per detected data symbol can suit the prevailing noise level, so that more operations are performed as the instantaneous noise level increases. The Viterbi Algorithm detector performs the same operations in every symbol interval, whatever the noise level is. Clearly, this is wasteful during low-noise periods. A suitable noise-adaptive detector of the type described above, may provide substantial reductions in detector complexity, for only small reductions in tolerance to noise.

Table A8.1 defines the notation which is used to describe the schemes tested by computer simulation, in Section 6.2.

6.1 SEQUENTIAL DECODING FOR CODED 8PSK

The two basic sequential decoding techniques are briefly outlined, together with some more recent variants. The possible application of sequential decoding to coded 8PSK is then analysed. (No computer simulations of sequential decoding were undertaken.)

Sequential decoding was initially introduced by Wozencraft, but the most widely used algorithm is due to Fano.[19,74] The Fano Algorithm will be described,[19] followed by the conceptually simpler Stack Algorithm.[74]

While the Viterbi detector stores all vectors which could conceivably be the Maximum Likelihood vector, sequential decoders essentially consider only one vector at any time. The one vector is that which "appears" to be most probable.[19] The decoder is allowed to back-up in time and change symbol values in this stored vector, (termed a back-up search). A metric associated with the stored vector is used to decide whether a back-up search is required. The metric used is not equivalent to the cost used in a hard-decision Viterbi detector unless the vectors being compared contain the same number of symbols. In particular the metric is biased to favour longer vectors. This ensures that the detector will tend to favour long vectors, so that the stored vector, over a reasonably long time span, tends to become longer, towards the end of the transmitted code sequence. Because of the bias, a vector which may be retained by a Maximum Likelihood detector, might not be considered by the sequential detector. The metric, here termed W_i' at time iT, is outlined in References 19 and 74. The bias term is usually chosen to ensure that, over a reasonably short length of time, the metric of the correct vector increases, while the metrics of incorrect vectors decrease. The metrics used here are based on the inner product of the received sequence of code symbols, and possible forms of the sequence in the absence of noise, rather than on Euclidean or Unitary distance. Thus the more positive the metric, the greater the likelihood that the corresponding possible sequence is correct. Usually the metrics take on only integer values, being associated with the Hamming distance. For coded 8PSK, the latter is the number of differences between the values of the binary code symbols $\{c_i(1)\}$,$\{c_i(2)\}$, and $\{c_i(3)\}$, in two sequences. (The convolutional code is described in Section 2.5.) Two incremental metrics are defined. A small positive incremental metric is added to the metric of a possible code sequence if the values of $c_i(\ell)$ in this sequence and the sequence of code symbols actually received are the same, (where ℓ=1,2, or 3). A larger negative incremental metric is added, if the values of $c_i(\ell)$ are not the same.

An efficient sequential decoding algorithm must be able to quickly detect a generally decreasing metric associated with an incorrect stored vector, so that the required back-up search to find the correct vector is not too computationally intensive. A running threshold metric, Γ, is stored which may be raised or lowered by increments Δ, where Δ is called the threshold spacing. When the stored metric falls below the current value of Γ, it indicates that a back-up search may be needed. The principle rule for the Fano Algorithm is that the decoder neither extends a stored vector by appending symbols, nor moves back along a stored vector by deleting symbols, if the stored metric is less than the current value of Γ. The stored vector is expanded, (see Section 3.2), and the associated metrics are determined. The decoder usually selects the expanded vector with the highest metric as the new stored vector. If the metric of the new stored vector is greater than $\Gamma+\Delta$, Γ is raised by Δ. When the metric of the chosen expanded vector is less than Γ, the decoder attempts to move backwards along the current vector to produce a vector with a metric which is greater than Γ.

When such a vector is found, the value of the most recent data symbol in the stored vector is changed to that which gives the next highest metric, (a lateral move). Forward decoding is then resumed. If no such value of this data symbol exists, because the current value is that producing the lowest metric, a further backwards move is made. When no vector with a metric greater than Γ is found, Γ is lowered by Δ and forward decoding is resumed as before. The full set of rules is given in Table 6.1.1, (from Reference 19). A forward move implies the selection of the most likely expanded vector as above. A backward move simply involves deleting the most recent data symbol value in the vector. A lateral move implies changing the value of the most recent data symbol in the stored vector, to the value associated with the next highest metric. Reference 19 details the algorithm more fully.

In the basic Stack Algorithm, an ordered list or stack of previously examined vectors of possible data symbols is kept. Each stack entry contains the vector along with its metric (which is usually the same as that used in the Fano Algorithm). The vector with the largest metric is stored at the top of the stack. This vector is expanded and the associated metrics are determined as for the Fano Algorithm. The original top vector is deleted and the remaining vectors, along with the expanded vectors, are reordered in the stack according to their metrics. When the vector at the top of the stack reaches the end of the transmitted code sequence, the top vector gives the detected data sequence. The reordering of the stack, which may be thousands of vectors in length, is very time consuming. The Stack-Bucket Algorithm by Jelinek[19,74] requires no reordering. The range of possible metrics is quantised into a number of fixed intervals called buckets, each of which is assigned a number of storage locations.

When a particular vector is expanded, the vector is deleted from its bucket and the expanded vectors are inserted as the top entries in the buckets associated with their metric values. The top vector in the top, (highest metric), non-empty bucket is expanded. The disadvantage is that the best vector is not always that expanded. A very good, (high likelihood), vector is expanded, which may be the best vector. Most practical implementations of the Stack Algorithm use this approach.[74]

A number of problems arise out of the two basic approaches, some of which are at least partly remedied by the modified schemes to be described. The number of computations required in increasing the number of symbols in the (best) stored vector is a random variable. Therefore input and output buffer stores are required to store the received samples, and the detected data symbols, respectively. Under severe noise conditions the number of computations rises dramatically. In the case of the Fano Algorithm the number and lengths of the required back-up searches increase, while equivalently in the case of the Stack Algorithm, the vector at the top of the stack will contain fewer data symbols. Long searches may cause the input buffer store to overflow causing the complete loss, (erasure), of large blocks of data. In the case of the Fano Algorithm such overflows are made less likely by restricting the back-search depth, (defined in terms of a maximum number of symbols in the stored vector which may be deleted), or by using an alternative and simpler detector when the input buffer store is full. A technique of quick threshold-loosening is also used to reduce the number of short back-up searches.[19] Goodman et al[84-86] use the algebraic properties of codes to simplify an essentially Fano-type algorithm. All short back-up searches are replaced by a direct mapping operation which finds the vector at the minimum distance from the received sequence, (where minimum distance is defined in Appendix A3). When a longer back-up search is required, the algorithm points out the most likely vector elements where the wrong symbol value may have been chosen. For the Stack Algorithm, a technique termed the Multiple Stack Algorithm, (MSA), guarantees that no buffer store overflows occur.[87] The size of the first stack is limited to Z_1 vectors. Decoding proceeds as for the original algorithm. If the first stack does not fill before the end of the transmitted code sequence is reached, the algorithm produces the same sequence of detected data as the original algorithm. If the first stack fills, the top Γ vectors are transferred to a second stack with Z storage locations, where $Z<<Z_1$. Decoding proceeds in the second stack. If the end of the transmitted code sequence is reached before the second stack fills, the top vector in the second stack is stored as a tentative decision. Decoding continues in the first stack, (which now has Γ empty locations), until the end of the transmitted code sequence is reached, if possible. If so, the metric of the top vector in the first stack is compared with the metric of the tentative decision from the second stack. The vector with the highest metric gives the detected data. If the first stack should fill again, a new second stack is formed using the Γ top vectors in the first stack. If the second stack should also fill, a third stack of the Γ top vectors from the second stack is formed, where the third stack also has Z storage locations. Additional stacks of Z storage locations are formed until a tentative decision is made. The decoder always compares each new tentative decision with the previous one, retaining the vector with the highest metric. The algorithm terminates

when the end of the transmitted code sequence is reached in the first stack. In addition a computational time limit is given. If this is exceeded, the best tentative decision is taken to be the detected data sequence. With a reasonable computational time limit, at least one tentative decision is always made. The Stack Algorithm in particular, but also Fano-type algorithms to a lesser extent, are really only suited to block data transmission with guard bands of non-data symbols inserted inbetween blocks. This is because back-up searches could theoretically extend back to the first transmitted code symbol, so that detection is only possible, en-bloc , when all the transmitted data has been fully processed. This precludes symbol by symbol detection. Such data blocks are terminated by (k-1) zeroes where k is the code constraint length, (see Section 2.5), in order to yield a code sequence with a final zero-valued code symbol which is clearly known at the receiver. In such cases, the computational time limit for the MSA would refer to the time required to decode one block or frame of data. In the case of block transmission, resynchronisation after an erasure, (a buffer store overflow), is simplified, since the whole data block is discarded. In a retransmission (ARQ) scheme this discarded data is not lost. In non-blocked data transmission, resynchronisation upon an erasure can be a major problem, involving the loss of a large number of data symbols.[19]

A discussion of sequential coding characteristics with reference to the coded 8PSK scheme, (Section 2.5), follows. Comparisons with Viterbi detection are also included, with reference to the same modulation scheme. The codes used in sequential decoding schemes can have very long constraint lengths, (often 50 or more symbols), since decoding speed is largely independent of code constraint length.[19] Such codes are impractical for Viterbi detection since a large number of stored vectors would be required. In addition, the processing load in the case of the sequential decoder is noise-adaptive. The algorithm tends to do more work when the noise level is high, (such as during a burst of noise). The Viterbi Algorithm performs the same operations whatever the prevailing noise level is, and is therefore wasting effort during low noise periods. It was noted in Chapter 1 that a basic aim of this study is to achieve a performance which is as close as possible to the best available tolerance to noise, at the lowest feasible level of complexity, at a bit error rate in the region of 1 in 10^4. A characteristic of optimal free-distance convolutional codes, (such as Codes 1 to 4 of Table 2.5.1), is that the promised asymptotic coding gain is achieved only at very low error rates.[19] At higher error rates, such as within the range noted above, the coding gain is much lower. Therefore, as the constraint length k increases, the incremental coding gain per increment in k, (for optimal free-distance codes within the above error rate range), is very small. This tends to diminish this advantage of sequential decoding, for coded 8PSK. Whether or not the noise-adaptive characteristic is an advantage is influenced by the possibility of buffer store overflow. The latter is influenced by the code properties and by the signal properties, (in particular the mapping function). Taking these in turn, a particularly important code characteristic for sequential decoders is the code distance profile,[73] which is itself a function of the code's column distance function (CDF).[73] The CDF, $d_c(n)$, is defined in terms of the minimum Hamming distance between all pairs of code sequences with n symbols, which differ in the earliest symbol, as given below.

$$d_c(n) = \min_{\{q^1\}\neq\{q^2\}} d_H(\{c^1\},\{c^2\}) \qquad (6.1.1)$$

$\{q^i\}$ and $\{c^i\}$, i=1,2 are respectively, the n-symbol data and code sequences, and $d_H(.,.)$ is the Hamming distance between the sequences of symbols in the brackets. The distance profile is the vector

$$D_p = [d_c(1),d_c(2),\ldots,d_c(k)] \qquad (6.1.2)$$

where k is the code constraint length. This is a measure of the rate of growth of the CDF with time. Rapid initial growth in the values of the elements of D_p ensures fast sequential decoding, (a low number of usually short back-up searches), and low erasure probability.[73,74] Good codes for sequential decoding also have large minimum free distances for maximising the asymptotic coding gain.[73,74] The codes of Table 2.5.1 are optimal with respect to their minimum free distances.[12] This may well imply that a sequential decoder, employed in the detection of data coded using one of these codes, would be prone to buffer store overflow, (or, in the case of the MSA, would yield a poor tolerance to noise because of a computational time limit which is too low). This is because their distance profiles are not optimised. The mapping function onto the complex number plane also causes problems. Sequential decoding theory usually assumes that the stored metrics are determined using the Hamming distance. This cannot be used for this coded scheme, because large unitary distances are not equivalent to large Hamming distances, (see Figure 2.5.4 and Table 2.1.1). The unitary distance measure is optimal for these codes, (see Appendix A7). In Section 4.2 it was noted that all four expanded vectors of a given vector produce values of p_i' in the complex number plane, (after coding and mapping), which all belong to one of the two sets of Figure 4.2.1. It was noted that for an arbitrary stored vector the likelihood that one of these values p_i' is that closest to the received sample, is about ½. The result is that quite a few of the data symbol values in two vectors may be different, while the metrics of the vectors in the absence of noise may be very similar. This is because the distance between the corresponding sequences of the $\{p_i'\}$, is small. Clearly, the minimum distance properties of the code will eventually ensure a reasonably large distance between the two vectors, but this may only occur after quite a long span of the $\{p_i'\}$. It will be easy for a sequential decoder to advance quite a long way with a vector containing the wrong data symbol values before the error is detected. In such cases long back-up searches are required, increasing the erasure probability. In addition the channel error statistics of Table 5.2.2 indicate that the sequential decoder may well need to back-up quite often, since the error rate in the received samples is significant, even at reasonably high signal to noise ratios.

Clearly, since the unitary distance measure is optimal in this situation, soft-decision decoding is imperative. Many comparisons of Viterbi and sequential decoding tend to be biased since they assume the Hamming distance measure for the Viterbi scheme,[74,87] whereas the Viterbi detector can gain nearly 3dB in tolerance to noise at high signal to noise ratios by using soft-decision metrics, even when the quantisation is quite coarse.[19] This gain is achieved with little increase in complexity.[19] On the other hand, the complexity increase for a sequential decoder using soft decisions is considerable, principally because it requires so much buffer store capacity, (together with the required control circuitry). In addition, quick threshold-loosening cannot be used in soft-decision sequential decoders, and sequential decoders are very sensitive to AGC, (automatic gain control), inaccuracies, which affect the threshold settings for determining the soft-decision metrics.[19] Therefore, soft decision decoding is not advised for sequential decoders.

Many comparisons of Viterbi and sequential decoding are based on the number of "computations" required, where one computation is defined as those operations required to increase by one the value of i, of the most recent data value q_i', in each stored vector (where there is only one such vector in the sequential decoder).[74,87] A measure of the relative effort required to undertake such a computation, and the relative ease of performing operations in parallel, are not fully considered. Because of this the speed gains claimed for sequential decoders are somewhat biased, since such a computation is often more complex for the sequential decoder.[74]

In addition, as noted earlier, sequential decoding is really only advisable for block data transmission using ARQ, (automatic retransmission request), techniques. It was noted that the data blocks require (k-1) redundant zero-valued symbols appended to the end of the blocks to terminate them.[19] Since very long constraint length codes are used when sequential decoders are implemented, these redundant symbols can constitute a considerable proportion of the total number of symbols in the data block, giving an undesirable overhead of symbols carrying no information. Since the constraint lengths are so much shorter for the codes used when Viterbi detection is implemented, there is only a very small overhead of code symbols in a blocked data scheme. Finally, if the sequential decoder is to be transparent, (that is, seen as a "black box" with a constant-rate sequence of input samples and a constant-rate sequence of detected output data), a delay of possibly hundreds of data symbols is required to ensure that the detected data stream is continuous, even when very long back-up searches are underway.[19]

In conclusion sequential decoding, although offering a number of possible advantages, (especially the noise-adaptive characteristic), is not considered further for this coded 8PSK scheme. The major problems involve the signal characteristics which could lead to long back-up searches, and the need to use soft-decision metrics. Section 6.2 introduces a noise-adaptive detector where the maximum computational effort, in contrast to the sequential decoder, is fixed, and at a level which is only marginally greater than that of the Viterbi detector. Because of this, and despite the signal characteristics, the potential buffer store overflow problems are much less severe in this case. Since it is very heavily based on the Viterbi Algorithm, it carries with it many of the attendant advantages that Viterbi detection has in comparison with sequential decoding, (as outlined above).

Rule	Conditions		Action	
	Previous Move	Comparisons	Final Threshold	Move
1	Forward or Lateral	$W'_{i-1}<\Gamma+\Delta, W'_i \geq \Gamma$	Raise (if possible)	Forward
2	Forward or Lateral	$W'_{i-1} \geq \Gamma+\Delta, W'_i \geq \Gamma$	No change	Forward
3	Forward or Lateral	Any W'_{i-1}, $W_i<\Gamma$	No change	Backwards
4	Backwards	$W'_{i-1}<\Gamma$, any W'_i	Lower	Forwards
5	Backwards	$W'_{i-1} \geq \Gamma$, any W'_i	No change	Lateral if possible otherwise backwards

TABLE 6.1.1: Sequential Decoding Rules for the Fano Algorithm

6.2 NOISE-ADAPTIVE VITERBI-TYPE DETECTOR[93]

In Reference 76, for channels with intersymbol interference, the authors propose that a decision can be made between two possible sequences of received samples in the absence of noise, once the distance between them exceeds the minimum distance d_{min}, (equivalent to d_{free} for coded systems). d_{min} and d_{free} are defined in Appendix A3. These values are calculated using the unitary distance measure defined in Appendix A7. They contend that the probability of discarding the correct sequence, when a number of sequences are to be decided between, is upper-bounded by $\alpha Q(d_{min}/\sqrt{2N_O})$ where α is a constant, $\alpha \geq 1$, and $Q(.)$ is the error-function[1]. Vermeulen[88], considering a similar proposal for channels involving intersymbol interference, defines the probability of error as being wide-sense asymptotically optimal, (wsao), when;

$$\lim_{E_b/N_O \to \infty} \frac{\log P(e)}{\beta \log P_{opt}(e)} = 1 \qquad (6.2.1)$$

$P_{opt}(e)$ is the bit error probability for Maximum Likelihood detection, $P(e)$ is the bit error probability in the proposed scheme, and β is a constant, $\beta \geq 1$. E_b/N_O is the signal to noise ratio. E_b is the average energy transmitted per data bit, and $N_O/2$ is the two-sided power spectral density of the additive white Gaussian noise. wsao ensures that as the signal to noise ratio, (E_b/N_O), tends to infinity, the additional transmitter power, (in decibels), required to compensate for a degradation in tolerance to noise compared with the optimal tolerance to noise,tends to zero. The above described proposal ensures asymptotic optimality.[76]

Consider the following adaption of the Viterbi Algorithm. Each stored vector is expanded, the relevant costs are calculated, and the

Viterbi selection algorithm is undertaken, as in Section 3.2. The resulting vector with the lowest cost is found as usual. Then, for all the other stored vectors, the distances between the sequence of the $\{p'_i\}$ of the vector with the lowest cost, and the sequences of the $\{p''_i\}$ of the other vectors, are determined as in Equation 6.2.2. The $\{p'_i\}$ and the $\{p''_i\}$ are the sequences of the possible values of the received samples in the absence of noise, for the lowest-cost vector and one of the other stored vectors respectively.

$$d^2 = \sum_{j=1}^{i} |p'_j - p''_j|^2 \qquad (6.2.2)$$

The value of d^2 for each stored vector is compared with $d^2_{free}/4$. If $d^2 \geq d^2_{free}/4$, the vector is discarded. The algorithm continues in this way, the major difference compared with the Viterbi Algorithm being that the number of stored vectors is a variable, less than or equal to 4^{k-1}, where k is the code constraint length. The philosophy of such a scheme is that the Viterbi detector may momentarily reject the correct vector, if the noise produces a received sequence which is nearer to a sequence of the $\{p'_i\}$ which is not the transmitted sequence. This happens if the magnitude of the noise vector is greater than $d_{free}/2$.[88] A scheme that discards sequences of the $\{p''_i\}$ which are distanced $d_{free}/2$ or more away from the current most likely sequence, (the $\{p'_i\}$), will be asymptotically optimal.[88]

Such a scheme is in practice difficult to implement since the determination of d^2 is not simple. As long as a particular vector has the lowest cost, the determination of d^2 simply involves updating stored values of d^2 for the other stored vectors, at the end of each symbol interval. Clearly though, when the lowest-cost vector changes, such stored values of d^2 will be incorrect, and require recalculation which is time-consuming.

Consider the following modification. Instead of comparing distances between possible sequences of received samples in the absence of noise, a scheme could alternatively consider the costs of the stored sequences, compared with the zero-cost attributed to the lowest-cost sequence. Section 3.2 describes the calculation of these costs, which use the unitary distance measure described in Appendix A7. The cost of a vector is a measure of how likely it is that the vector's element values are the same as those of the corresponding transmitted data symbols. A low cost implies high likelihood. Here, vectors are discarded once their costs exceed $(d'/2)^2$ where d' is some constant value. This is not equivalent to the initial proposal, and therefore asymptotic optimality cannot be assumed, but one would expect the results be very similar, since the difference in the costs of two possible code sequences is to some extent a measure of the distance between the two sequences. Since the costs are stored in a conventional Viterbi detector, such a scheme would not involve the calculation of any new values. The only addition would be a test, which discards all stored vectors whose costs are greater than $(d'/2)^2$.

The following section describes in detail a scheme incorporating this technique. The first part of the algorithm decides for a given vector, how many expanded vectors are to be derived from it. These expanded vectors are those, of the four possible expanded vectors, with the lowest costs. The second part of the algorithm is that described above, which discards vectors after the selection procedure, (see Section 3.2), if their costs are greater than a given value.

The two parts of the algorithm are described for a scheme using Code 1, (from Table 2.5.1). Initially, the stored values in the detector are described just prior to the receipt of the sample r_i at time t=iT. The proposed detector contains storage locations for 16 vectors of possible data symbol values and their associated costs, just as in the case of the Viterbi Algorithm detector. The difference is that only k_{i-1} of these locations, where $k_{i-1} \leq 16$ contain vectors which are to take part in the detection process upon the receipt of r_i. The remaining $16-k_{i-1}$ locations are taken to be empty in that they contain no useful information. Each of the k_{i-1} vectors is called a valid vector. The state of a vector at time t=iT is given by the combination of the values of the symbols, $q'_{i-k+1}, q'_{i-k+2}, \ldots, q'_{i-1}$, in the vector, k is the constraint length of the code (=3). The state of a valid vector is called a valid state. In the Viterbi detector, each stored vector has a different state, and all possible states occur among the stored vectors. Clearly in this case if $k_{i-1} < 16$, all possible states do not occur, but the states of the k_{i-1} stored vectors are still all different. Each valid vector has a stored cost $|W'_{i-1}|^2$.

On the receipt of sample r_i, the detector forms a number of possible values of the received sample, using the stored vectors. Each vector Q'_{i-1} is expanded, to give four expanded vectors, by appending one of the four possible data symbol values, q'_i=0,1,2 or 3. The elements of such an expanded vector are coded using the convolutional code described in Section 2.5 to give the vector of binary code symbols $[c'_i(1), c'_i(2), c'_i(3)]$

$$c'_i(j) = \sum_{\ell=1}^{2} \sum_{h=0}^{k-1} q'_{i-h}(\ell) g_h(\ell, j) \qquad (6.2.3)$$

$\oint$ denotes MODULO-2 summation. The $\{g_h(\ell,j)\}$ have the possible values 0 or 1. The two-component vector $[q_i'(1),q_i'(2)]$, where $q_i'(\ell)$, (for $\ell=1$ or 2), has the possible values 0 or 1, is uniquely related to the possible data symbol q_i' by the Gray-code mapping of Table 2.1.1. The vector $[c_i'(1),c_i'(2),c_i'(3)]$ is mapped onto the 8-level symbol c_i'.

$$c_i' = 2^2c_i'(1) + 2^1c_i'(2) + 2^0c_i'(3) \qquad (6.2.4)$$

Since $c_i'(1)$, $c_i'(2)$ and $c_i'(3)$, each have the two possible values 0 or 1, c_i' has one of the eight possible values 0,1,...,7. A possible value of p_i in the absence of noise is given by mapping c_i' onto a complex number p_i'. The mapping onto the complex number plane is given in Figure 2.5.4.

The first part of the algorithm is now outlined, which for a given vector Q_{i-1}', determines the costs of j of its four expanded vectors, where $j\leq 4$. These expanded vectors are called valid expanded vectors. The remaining (4-j) expanded vectors are discarded. The value of j is not the same for every vector Q_{i-1}'. Initially the possible value of r_i in the absence of noise is found which is nearest to the received sample r_i, (using threshold tests). Let this value of p_i be p_i''. For each expanded vector, the difference between the phase angles of p_i'' and p_i' is found, where this is the smaller of the two possible angular differences. (p_i' is the possible value of the received sample in the absence of noise, derived from the expanded vector.) An example is given in Figure 6.2.1. A look-up table could be used to give the angle, given p_i'' and p_i'. These angles are measures of the distances between p_i'' and the values of p_i'. The costs of those expanded vectors is now calculated, whose differences between the phase angles of p_i'' and p_i' are less than or equal to $\Delta\pi/4$, where Δ has one of the four values, 1,2,3 or 4. Clearly if $\Delta=4$, no expanded vectors are discarded. Note that Δ is not necessarily equal to the number of expanded vectors whose costs are calculated. Figure 6.2.2 shows (for $\Delta=2$), that either Δ or $\Delta+1$ expanded vectors' costs are calculated, depending on the values of p_i'. Two methods are used to set the value of Δ. In the first, termed the static expansion limitation method, a constant value for Δ, ($\Delta=1,2,3$ or 4) is stored within the detector. The second is termed the dynamic expansion limitation method, where Δ is set individually for each stored vector Q_{i-1}' in relation to its stored cost $|W_{i-1}'|^2$. Three cost thresholds, cth(1),cth(2) and cth(3) are stored. They are used to ascertain the range of costs, of four ranges in all, into which $|W_{i-1}'|^2$ falls for each vector Q_{i-1}'. The range of costs into which the cost of Q_{i-1}' falls is used to set Δ for this vector.

$$\begin{aligned} |W_{i-1}'|^2 &\leq \text{cth}(1) \; ; \quad \Delta=4 \\ \text{cth}(1) < |W_{i-1}'|^2 &\leq \text{cth}(2) \; ; \quad \Delta=3 \\ \text{cth}(2) < |W_{i-1}'|^2 &\leq \text{cth}(3) \; ; \quad \Delta=2 \\ |W_{i-1}'|^2 &> \text{cth}(3) \; ; \quad \Delta=1 \end{aligned} \tag{6.2.5}$$

Clearly, $\Delta=1$ is for the range of highest costs, (least likely possible data sequences), whereas $\Delta=4$ is for the range of lowest costs, (most likely possible data sequences). Once the set of valid expanded vectors has been determined, their costs are calculated. For each such expanded vector

$$|W_i'|^2 = |W_{i-1}'|^2 + |r_i-p_i'|^2 \tag{6.2.6}$$

where $$|r_i-p_i'|^2 = [\text{Re}(r_i-p_i')]^2 + [\text{Im}(r_i-p_i')]^2 \tag{6.2.7}$$

The cost is based on the unitary distance measure (see Appendix A7). (Clearly the value of $|W'_{i-1}|^2$ is that of the vector from which the expanded vector is derived, and the value of p'_i is that for the expanded vector.) For each combination of the values of the symbols $q'_{i-k+2}, q'_{i-k+3}, \ldots, q'_i$, (the state of a vector at time $t=(i+1)T$), the detector selects the expanded vector with this combination of values, which has the lowest cost. If there are no expanded vectors with a given state, the selection process for this state is not undertaken. After this procedure has been undertaken for all states which occur among the expanded vectors, the selected vector with the lowest cost is found. This cost is subtracted from the costs of all the vectors, to prevent overflow in their stored values. The value of q'_{i-N+1} in the vector with the lowest cost is the detected data symbol. The delay in detection is NT seconds. Clearly, this is equivalent to the Viterbi Algorithm procedure of Section 3.2, ammended by the fact that all possible states do not occur among the expanded vectors at times. The second part of the algorithm is not performed in the Viterbi Algorithm detector.

The second part of the algorithm simply discards those vectors which were selected in the above procedure, whose costs are greater than a value C_m, stored in the detector. The result is a set of k_i stored vectors called valid vectors, $(k_i \leq 16)$, where k_i may not be equal to k_{i-1}. The process continues in this way for received sample r_{i+1}, etc.

The philosophy of this algorithm is as follows. The first part of the algorithm has two characteristics. The first characteristic is that vectors $\{Q'_{i-1}\}$ with low costs are more likely to be the Maximum

Likelihood vector than those with high costs. Therefore Δ is made large for low cost vectors, and small for high cost vectors (in the dynamic expansion limitation method). This ensures that fewer of the expanded vectors of low-cost vectors are discarded than for high-cost vectors. Secondly, the j expanded vectors whose costs are calculated for each vector Q'_{i-1}, are the j expanded vectors of Q'_{i-1} with the lowest costs. The arguments in favour of the second part of the algorithm were considered in depth at the start of this section, where C_m is the maximum cost referred to in that section. Since the number of stored vectors, k_i, and therefore the processing time per detected data symbol, vary from symbol interval to symbol interval, buffer stores are required to hold a number of received samples $\{r_i\}$, and a similar number of detected data symbols $\{q'_i\}$. The operation of such a system is transparent in that continuous, constant-rate, sequences of samples $\{r_i\}$ and detected data $\{q'_i\}$ are sent into, and out of the detector, respectively.

A number of computer simulation tests were undertaken to ascertain values of the parameters Rexp (equivalent to Δ in the static expansion limitation method), C_m and cth(1),cth(2) and cth(3), which yield cost-effective compromises between equipment complexity and tolerance to noise. The criterion used to define these compromises is a degradation in tolerance to noise, compared to Viterbi detection for a system incorporating Code 1, of approximately 0.5dB at a bit error rate (BER), of about 1 in 10^3, for as low a level of equipment complexity as possible.

Initially tests were undertaken for a scheme using Code 1, (Table 2.5.1), using the static expansion limitation method. A number of values of Rexp and the maximum cost C_m, were used at the signal to noise ratios,

E_b/N_0 = 4.6dB and 5dB. (See Appendix A5 for more details of the simulation techniques. Appendix A8 describes the notation used to describe the variants of this detector which were tested by computer simulation.) The results of these initial tests, and similar tests for schemes using the dynamic expansion limitation method and the constraint length k=4 codes, are presented by way of performance comparison tables, and two types of graph providing statistical information. The first type of graph gives the distribution of the number of vectors, k_i, averaged over the transmission of 3×10^5 data symbols, for each value of E_b/N_0, for a number of values of E_b/N_0. This type of graph is called the Type-A distribution in the following. The second type of graph gives a measure of the buffer store requirements. Once the number of stored vectors rises to be greater than or equal to a given value X, there will be X or more stored vectors for j consecutive symbol intervals, where j=1,2,... . When the number of stored vectors falls below X again, j is fixed. One "occurrence" of the fact that X or more vectors have been stored for exactly j symbol intervals, (after which the number of stored vectors fell below X), is said to have taken place. The second type of graph gives the number of such occurrences as X ranges over the values 2 to 4^{k-1}, where k is the code constraint length, and j has the values 1,2,... . This type of graph is called a Type-B distribution in the following.

Table 6.2.1 gives the results of these initial tests at E_b/N_0= 4.6dB. From Section 2.1, the value of p_i at the transmitter is such that $|p_i|^2 = [Re(p_i)]^2 + [Im(p_i)]^2$=4.0. Also included in Table 6.2.1 is a column stating the maximum number of vectors Sv. This simply gives the number of vector storage locations held in the detector.

If at any time the number of valid vectors exceeds Sv, the detector simply retains those vectors with the lowest costs, such that Sv vectors are retained. The results for C_m=120 show that a scheme with Rexp=2 yields results which are very similar to schemes with Rexp>2, but that Rexp=1 is too low. Also from Table 6.2.1, a small degradation in tolerance to noise occurs, if Rexp≳2, when C_m is reduced to 8,6 or 5. For example when Rexp=2 and C_m=5, the degradation in tolerance to noise is less than 0.4dB compared with Viterbi detection, for an average of 9 valid expanded vectors per symbol interval. For C_m=120, and Rexp=1, the degradation in tolerance to noise is somewhat higher at 0.6dB, whilst the average number of expanded vectors per symbol interval is considerably higher at 24. For C_m=4, the degradation rises more sharply. Despite this, it is useful to compare the case where C_m=4 and Rexp=1, with near-maximum likelihood System 1 detection with four stored vectors, (k_1=4). The number of expanded vectors per symbol interval is 16 in the latter case. At a BER of 5 in 10^2 from Graph 4.1.1, the degradation in tolerance to noise with respect to Viterbi detection is about 1.75dB for the System 1 detector. Therefore the detector using the new algorithm where C_m=4 and Rexp=1 gains some 0.4dB in tolerance to noise over System 1 detection with k_1=4, at the same BER, despite requiring on average less than a quarter of the expanded vectors per symbol interval. Clearly this is a very significant improvement. Table 6.2.2 outlines the results at a signal to noise ratio, (E_b/N_0), of 5dB. The results are very similar to those at 4.6dB. From a comparison of the results where C_m=5 in Tables 6.2.1 and 6.2.2 the schemes at E_b/N_0=5dB are degraded more heavily than those at E_b/N_0=4.6dB, but this is attributable to a drop in the average number of valid expanded vectors per symbol interval. Graph 6.2.1 is the Type-A distribution at 4.6dB, for a number of variants of the detectors. As C_m is decreased the curves become more concentrated towards the lower numbers of valid vectors. At a given value of C_m, the curves become more concentrated towards the lower numbers of valid vectors when Rexp is decreased from 2 to 1. The difference in the curves when Rexp is decreased from 4 to 2, at constant C_m, is negligible. For $C_m<6$ the curves have a markedly exponential-like fall-off. Graphs 6.2.2 to 6.2.5 are the Type-B distributions for certain of the schemes of Table 6.2.1. Since the curves fall-off very sharply with time, especially for large X, it is clear that the probability of buffer store overflow will be low, as long as the chosen detector can process a reasonable number of expanded vectors per symbol interval. The approximate size of the buffer store can be obtained by considering the curve for the number of valid vectors which is just higher than the average number of valid vectors, (from Table 6.2.1). If the rate of computation in the detector is adjusted so that this average number of vectors can be processed during one symbol interval, then if more than this number of valid vectors are stored the buffer store will tend to fill, and if less than this number of valid vectors are stored the buffer store will tend to empty. For example take Graph 6.2.2. The average number of valid vectors for this scheme is 2.5 from Table 6.2.1. Therefore, considering the X = 4 curve in Graph 6.2.2, a suitable buffer store would have in the region of 30 to 50 storage locations. This range is also typical of the other schemes of Graphs 6.2.2 to 6.2.5. Graph 6.2.6 is the Type-A distribution at a signal to noise ratio, (E_b/N_O) of 5dB, while Graphs 6.2.7 to 6.2.10 are the Type-B distributions for a selection of schemes at the same signal to noise ratio. The results are similar to those at 4.6dB.

Graph 6.2.11 gives the initial computer simulation results for the dynamic expansion limitation method, for a number of schemes, at a signal to noise ratio, (E_b/N_O), of 5.3dB. From the results of the static expansion limitation method, a value of C_m in the region of 5.0 is seen to be a good compromise between performance and equipment complexity. This value of C_m for a scheme using Code 1 is very nearly equal to $d^2_{free}/2$. Therefore $d^2_{free}/2$, (5.172), was chosen as one of the values of C_m in the tests. The results are given in Table 6.2.3 at the same value of E_b/N_O. The last column, (b/a), gives the average number of valid expanded vectors derived from a single vector. Table 6.2.3 indicates that, as for the static expansion limitation method, the lowest feasible value of the average number of valid expanded vectors derived from a single vector, lies in the region 2.0 to 2.5. Below this the BER tends to rise substantially. In general, for comparable values of the average number of valid expanded vectors derived from a single vector, the static and dynamic expansion limitation methods are very similar. The latter method though, allows values of the average number of valid expanded vectors derived from a single vector within the range 1.5 to 2.5, whereas the static expansion limitation method only allows the discrete values 1.5 and 2.5 (approximately, for random data). Clearly the dynamic expansion limitation method allows more scope for optimisation. Graph 6.2.11 is very similar to the Type-A distribution for the static expansion limitation method, while the Type-B distributions in Graphs 6.2.12

to 6.2.15 suggest that the buffer store requirements are very similar to those for the static expansion limitation method, (20 to 50 storage locations).

Tests were also carried out using the constraint length k=4 code, Code 3, at the signal to noise ratios, (E_b/N_0), 4.6dB, 5dB, and 5.25dB. The results are given in Tables 6.2.4 (4.6dB), 6.2.5 (5dB), and 6.2.6 (5.25dB). The Type-A distributions for a selection of the schemes in the preceeding tables, are given in Graphs 6.2.16, (4.6dB), 6.2.17, (5dB), and 6.2.18, (5.25dB). Table 6.2.5 shows again that reducing the average number of valid expanded vectors derived from a single vector, to a value below 2.0, leads to an increased degradation in tolerance to noise. For values of this measure above 2.0 for C_m=6.344, the degradation in tolerance to noise compared with Viterbi detection for Code 3 is effectively constant. This is so over a considerable range of the average number of valid expanded vectors per symbol interval, (13.3 to over 18.3). Reducing C_m to values below 6.344 leads to a sizeable increase in the degradation in tolerance to noise, but also to a sizeable reduction in the average number of valid expanded vectors per symbol interval. Clearly the potential equipment complexity gains indicated by Tables 6.2.4 to 6.2.6 are considerable, since the corresponding Viterbi detector processes 256 expanded vectors during every symbol interval. For example, the last row in Table 6.2.5 is for a scheme which processes on average only 1/34th of the expanded vectors per symbol interval that the Viterbi detector processes, whilst losing only 0.83dB in tolerance to noise. Also, included, for C_m=4.8, are two tests where the maximum number of vectors Sv is reduced from 64. The results show that the degradation in tolerance to noise is negligible, if Sv is reduced to 16. This is important, since long constraint-length codes require a phenomenal amount of storage capacity in a Viterbi detector. If Sv can be reduced considerably in such cases, the amount of storage capacity required is also reduced considerably, (but with the penalty that a cost ranking process must be introduced when the number of vectors exceeds Sv, as described earlier). From Table 6.2.6 it is clear that even for large values of C_m, the performance of these schemes using Code 3 does not approach the performance of Viterbi detection as closely as certain schemes using Code 1. Graphs 6.2.16 to 6.2.18 indicate that the curves of the Type-A distributions do not have exponential-like fall-offs for typical values of C_m, (although they do for low values of C_m). Also for C_m =6.344, the proportion of the total transmission time becomes negligible well before the maximum number of vectors, (Sv = 64) is reached. This suggests, (as was seen in Table 6.2.5), that Sv can be reduced considerably without impairing performance.

Graph 6.2.19 is the Type-A distribution for a scheme using Code 2, at a number of signal to noise ratios. Table 6.2.7 gives the results, with respect to Viterbi detection for Code 3. The definition of an error burst is given in Appendix A5. From Table 6.2.7 it is clear that the complexity, measured in terms of the average number of valid expanded vectors per symbol interval, reduces as the noise level falls. The scheme is therefore noise-adaptive in the sense that more processing is undertaken when the noise level is high. Table 6.2.8 gives the results for schemes using Code 4 at a signal to noise ratio, (E_b/N_0), of 5.25dB. Again, the BER tends to rise significantly when the average number of valid expanded vectors derived from a single vector falls below about 2.0.

From these initial tests three schemes, two using Code 1 and one using Code 4, were chosen. Full computer simulation tests of these three schemes were undertaken. A scheme using Code 4 was chosen in preference to a scheme using Code 3 because the schemes of Table 6.2.8 perform consistently better than those of Table 6.2.6, both in terms of tolerance to noise and complexity. (Schemes using Code 2 produce results which are very similar to those of schemes using Code 4.) The chosen schemes are outlined in Table 6.2.9. The accuracy of the results is of the order of ±0.25dB over the range of BER, 1 in 10^3 to 1 in 10^4. The results are shown in Graph 6.2.20 in comparison with Viterbi detection for schemes using Codes 1 and 3, and threshold detection for QPSK. It is evident that the degradation in tolerance to noise compared with Viterbi detection, for the noise-adaptive schemes, is low. Tables 6.2.10 to 6.2.12 give the results for the three schemes at various signal to noise ratios. Table 6.2.13 gives results for a less complex scheme using Code 4, than that of Table 6.2.12. From Graph 6.2.20 and Table 6.2.10, the first scheme using Code 1 is only marginally degraded in tolerance to noise compared with Viterbi detection, (0.44dB at a BER of 3 in 10^4). It can be seen that this scheme has an average number of valid expanded vectors derived from a single vector in excess of 2.5. The average number of bit errors per burst is not significantly higher than that for Viterbi detection, (Table 3.2.2). At a signal to noise ratio, (E_b/N_0), of 5.8dB, using the average number of valid expanded vectors per symbol interval as a measure, this scheme is approximately 8.5 times less complex than Viterbi detection, (which processes 64 expanded vectors per symbol interval). From Table 6.2.11, the less complex scheme using Code 1 is some 12 times less complex than Viterbi detection at the same signal to noise ratio. At this signal to noise ratio the degradation in tolerance to noise compared with Viterbi detection is 0.6dB, only marginally greater than that for the other scheme using Code 1. In addition, the average number of bit errors per burst is only marginally greater than in the former scheme. In this case the average number of valid expanded vectors derived from a single vector is in the region 1.7 to 2.2 which, from the initial tests, approaches the lower limit at which the BER rises substantially. Graph 6.2.21 is the Type-A distribution for the first scheme using Code 1 described in Table 6.2.9. The curves become more concentrated towards the lower numbers of valid vectors as the noise level falls. This indicates the noise-adaptive nature of the algorithm in that more vectors are stored, and therefore more processing is required, when the noise level is high. Graphs 6.2.22 and 6.2.23 are the Type-B distributions at two values of BER. Clearly these Type-B distributions are very similar to those presented earlier. The required buffer store size is again in the range, 20 to 50 samples. Graph 6.2.24 is the Type-A distribution for the second scheme of Table 6.2.9 using Code 1, at three values of the BER. The Type-B distributions at two values of the BER are given in Graph 6.2.25 and 6.2.26. These are very similar to those of the first scheme of Table 6.2.9 which uses Code 1.

From Graph 6.2.20, the tolerance to noise of the scheme using Code 4 is approximately equivalent to that of Viterbi detection for a scheme using Code 1, at a BER of 1 in 10^4. In terms of complexity, from Table 6.2.12 at a signal to noise ratio, (E_b/N_0), of 5.5dB, the

the scheme is about 4.7 times less complex than Viterbi detection for Code 1, which is a significant saving. The measure of complexity is, as before, the average number of valid expanded vectors per symbol interval. Compared with the noise-adaptive schemes using Code 1, this scheme is considerably more complex, although it does gain somewhat in tolerance to noise. A comparison is given in Table 6.2.14. Clearly the noise-adaptive schemes using Code 1 are more attractive. Graph 6.2.27 is the Type-A distribution for the scheme of Table 6.2.9 using Code 4. As noted earlier, the curves fall-off very quickly at relatively low numbers of valid vectors. No Type-B distributions for the constraint length k=4 codes have been produced, because of computing restrictions.

Graphs 6.2.28 and 6.2.29 use the third to last and second to last columns of Tables 6.2.10 to 6.2.12 to give measures of system complexity, as the signal to noise ratio is varied. These graphs support the supposition that the processing load reduces as the noise level falls, so that the algorithm is noise-adaptive. Clearly, from Graphs 6.2.28 and 6.2.29, the difference between the two schemes using Code 1, lies solely in the number of valid expanded vectors per symbol interval, and therefore in the number of valid expanded vectors derived from a single vector. An interesting point is that the complexity of the scheme using Code 4 falls off more rapidly than does the complexity of the schemes using Code 1. Therefore at high signal to noise ratios the scheme using Code 4 may compete, both in terms of tolerance to noise and complexity, with the schemes using Code 1.

The following is an analysis of the feasibility of implementing such a system in practice, in the light of the potential savings highlighted in the preceeding sections. Also included is a comparison of the scheme with Viterbi Algorithm and sequential decoding techniques.

In a practical implementation of the algorithm, the rate of operation of the detector is adjusted to handle conditions where the signal to noise ratio has its typical or average value. During high noise periods the input buffer store holding the received samples will gradually fill, and the output buffer store holding the detected data symbols will gradually empty. When the input buffer store is full, the following is implemented. The maximum number of stored vectors, Sv, is reduced to a value such that in succeeding detection processes the input buffer store gradually empties. The chosen value of Sv is clearly influenced by the time required to rank the costs of the vectors, when the number of these exceeds Sv, as well as by the average rate of operation of the detector. Eventually, Sv is reset to its previous value, (when the number of samples held in the input buffer store has reduced sufficiently).

An essential feature of the new algorithm is that its implementation is based firmly on that of a conventional Viterbi Algorithm detector. A store is available which is capable of holding Sv vectors together with their associated costs, where Sv is, for short constraint length codes, equal to the number of vector storage locations in the corresponding Viterbi Algorithm detector. In the vector selection process, the same procedure is carried out as for the Viterbi detector, but modified by the fact that the majority of the vectors are not normally present. The efficiency with which the detector manages the set of unused storage locations is crucial in any attempt to approach the theoretical savings in system complexity. More specifically, the problem lies in minimising the time spent in determining the valid vectors and valid expanded vectors. Also, it is crucial that Viterbi Algorithm processing, (that is, cost calculation, selection of the vectors, lowest cost determination, followed by storage of the selected vectors and costs), is not held up. A short preliminary study of the problem has produced a possible solution, in the form of the block diagrams of Figures 6.2.3 and 6.2.4. These stress the main points pertaining to the implementation problem described above, so that it has been necessary to dispense with some of the details. The figures refer to a scheme using Code 1, where Sv=16.

The proposed implementation separates the determination of the valid expanded vectors, (validity test), from the algorithm which chooses the valid vectors from these valid expanded vectors and determines the detected data symbols, (performed by the Viterbi processor). Figure 6.2.3 is the block diagram of the validity test circuit. Figure 6.2.4 is the block diagram of the Viterbi processor. The two operations can proceed independently, feeding the outputs of the validity test circuit into a buffer store for use by the Viterbi processor. (The way in which this information is used by the Viterbi processor is not included in Figure 6.2.4.)

In Figure 6.2.3 the input to the valid-vector test circuit is one of the possible states of a vector, (given by a four-bit integer). The test determines whether a vector Q'_{i-1} is stored which has this state. If so the valid expanded vectors are determined, using the valid expanded vector test. The latter test can be implemented using a Read Only Memory (ROM). The result of the latter test is fed to the Viterbi processor, and to a counter which designates the data symbol value for the next expanded vector, q_i'. In addition, the result of the valid-vector test is fed back to a counter to designate the next possible state when appropriate.

Figure 6.2.4 is very similar to a conventional implementation of the Viterbi Algorithm.[19] The possible data symbol value, q_i', and the state of the vector Q_{i-1}', called Φ_i, are used to extract the following quantities. (As a reminder, the state Φ_i at time t=iT for Code 1 is given by the combination of the values of the symbols q_{i-2}' and q_{i-1}', in an expanded vector).

(a) The value of q_{i-2}' given by state Φ_i is fed to the main processor.

(b) The value of the state, Φ_{i+1}, and the value of the complex number p_i'. The state Φ_{i+1} is the combination of the values of the possible data symbols q_{i-1}' and q_i'. The determination of p_i' was considered earlier in this section.

(c) The value of the cost $|W_{i-1}'|^2$, of vector Q_{i-1}'. $|W_i'|^2$ is calculated using Equation 6.2.6.

In the main processor, for each possible state Φ_{i+1}, there is a stored cost and a value of q_{i-2}'. This stored cost is the lowest of those costs $|W_i'|^2$ fed so far to the processor, of expanded vectors with the state Φ_{i+1}. The associated value of q_{i-2}' is the value of q_{i-2}' in the valid expanded vector which has this particular cost, and has the given state Φ_{i+1}. When the new value of $|W_i'|^2$ of a valid expanded vector which has state Φ_{i+1} is fed in, it is compared with the processor-stored cost for state Φ_{i+1}. If the new cost is lower than the stored cost, it is stored in place of the latter, and the value of q_{i-2}' in the valid expanded vector which has this new cost is stored in place of the processor-stored value of q'_{i-2}. This procedure continues until all the costs of the valid expanded vectors have been fed to the main processor. The result is the set of selected vectors and costs before the second part of the algorithm, (to discard selected vectors whose costs are greater than c_m), is implemented. The process of comparing costs as they are fed into the main processor is called continuous ranking. A stored value of q'_{i-2} does not of itself define a selected vector, but in conjunction with Φ_{i+1}, (the values of q'_{i-1} and q'_i), the position of the vector in the store is fully defined. Continuous ranking is also used to ascertain the overall minimum cost at time t=iT. Once determined this is added to c_m in order to provide a means of undertaking the second part of the algorithm. In Figure 6.2.4 this is undertaken before the lowest cost is subtracted from all the costs, contrary to the method described earlier. If the cost of a selected vector passes the test the post-detector processor is enabled. Otherwise the post-detection processor is disabled. Its job is to store the valid vectors and associated costs. The proposed arrangement for the storage of the vectors is often termed the Path Memory Traceback[19,89], or Pointer-organised[90] storage method. The contents of a number of vectors will be identical for time t≤jT, for some j. Therefore storing separate vectors is very wasteful of storage capacity. This method does not store separate vectors. For example, if two vectors have become the same for t≤jT, only one set of the possible data symbols for t≤jT, is stored for both vectors. This method also dispenses with the need to change the contents of many storage locations after the selection procedure. The method is described more fully in Reference 19.

Detection now involves the transfer of a block of N' detected data symbols $\{q_j'\}$, for $j=i-N-N'+1, i-N-N'+2, \ldots, i-N+1$, to the output buffer store every N' symbol intervals. Clearly the detection delay is increased by at least N' symbol intervals.[19] In addition, the subtraction of the minimum cost from all $|W_i'|^2$, to prevent overflow, is undertaken in the post-detection processor.

Clearly the proposed implementation is essentially serial so that data transmission of the order of megabits per second is not possible. (This does not preclude the introduction of some measure of parallel processing to increase the operating speed.) At lower data rates, the algorithm seems an ideal application for digital signal processors such as the TMS32010/TMS32020 series.[91,92] Such processors are very efficient at dealing with algorithms of this type, where the required processing changes from symbol interval to symbol interval.

The new algorithm has a number of advantages over sequential decoding, for the present application to coded 8PSK. Clearly the buffer store size requirements are less severe for the new algorithm. 20 to 50 stored samples are typical compared with some hundreds of stored samples for typical sequential decoders, (see Section 6.1). In addition, the Type-B distributions, which were presented to gain measures of the required size of the buffer store, show that buffer store overflow is probably less likely in the new scheme. A major factor in this is that there is an upper-bound to the processing time required per detected data symbol, which is only slightly greater than the time required to yield a detected data symbol in the Viterbi detector. (The overhead in processing time compared to Viterbi detection is due to the need to determine the valid expanded vectors, and due to the required cost ranking, if Sv is set below that required for the Viterbi detector.) Sequential decoders have no such inherent upper-bound, although some techniques, (such as the Multiple Stack Algorithm), do specify a computational time limit. This also means that the range of processing times per detected data symbol is much smaller than for the sequential decoder, which may have to undertake some very long back-up searches. This is essentially because no back-up searches are required in the new scheme, the technique being fundamentally "feedforward". Clearly, without significant restrictions on the value of Sv, long constraint length codes cannot be accommodated by the new technique since the required storage capacity would become prohibitive. This advantage that sequential decoding has is insignificant in the range of BER considered, (1 in 10^3 to 1 in 10^4), since longer constraint length codes do not lead to significant gains in tolerance to noise in this region, (Section 6.1). As noted in Section 6.1, sequential decoding is not really suited to continuous data transmission because of the risk of buffer store overflow. A block transmission scheme with repeated transmission of erased blocks, (automatic retransmission request, ARQ), was considered to be the best method. Clearly, the new method could also be used in a block transmission scheme, but in addition, with an efficient algorithm to prevent buffer store overflow as described earlier, the new scheme can also be used for continuous data transmission.

The conclusion is that the novel technique promises significant reductions in equipment complexity over the corresponding Viterbi Algorithm detector, for negligible losses in tolerance to noise. Also, of the two noise-adaptive schemes considered in this Chapter, the novel technique provides a number of important advantages with only a few, insignificant, disadvantages.

C_m	Rexp	Sv	B.E.R.	DEGRADATION IN TOLERANCE TO NOISE OF VITERBI DETECTION (CODE 1) AT GIVEN BER (dB)	AVERAGE NUMBER OF VALID VECTORS	AVERAGE NUMBER OF VALID EXPANDED VECTORS PER SYMBOL INTERVAL
120	4	16	3.38×10^{-3}	0.0	16	64
120	3	16	3.5×10^{-3}	<0.1	16	56
120	2	16	3.7×10^{-3}	0.1	16	40
120	1	16	1.2×10^{-2}	0.6	16	24
120	4	12	4.5×10^{-3}	0.2	12	48
120	4	8	1×10^{-2}	0.5	8	32
8	4	16	3.4×10^{-3}	<0.1	9.4	37.5
6	4	16	4.33×10^{-3}	0.15	5.2	20.7
6	2	16	4.89×10^{-3}	0.2	5.1	12.9
6	1	16	1.84×10^{-2}	0.8	4.05	5.88
5	4	16	6×10^{-3}	0.3	3.6	14.3
5	2	16	7.5×10^{-3}	<0.4	3.5	9.0
5	1	16	2.7×10^{-2}	1.0	3.15	4.5
4	4	16	1.75×10^{-2}	0.8	2.5	10
4	2	16	2.0×10^{-2}	0.85	2.5	6.47
4	1	16	5.0×10^{-2}	1.35	2.5	3.55

TABLE 6.2.1: Performance Results for Schemes Using the Static Expansion Limitation Method for Code 1 at a Signal to Noise Ratio (E_b/N_O) of 4.6dB

C_m	Rexp	Sv	B.E.R.	DEGRADATION IN TOLERANCE TO NOISE CF VITERBI DETECTION (CODE 1) (dB)	AVERAGE NUMBER OF VALID VECTORS	AVERAGE NUMBER OF VALID EXPANDED VECTORS PER SYMBOL INTERVAL
120	4	16	9.73×10^{-4}	0.0	16	64
120	2	16	1.2×10^{-3}	<0.1	16	40
120	1	16	6.2×10^{-3}	0.7	16	24
5.5	4	16	1.81×10^{-3}	0.2	3.66	14.64
5.5	2	16	2.27×10^{-3}	0.29	3.62	9.22
5.5	1	16	1.34×10^{-2}	1.05	2.95	4.18
5	4	16	2.9×10^{-3}	<0.4	3.03	12.1
5	2	16	3.38×10^{-3}	0.45	3.0	7.7
5	1	16	1.87×10^{-2}	<1.2dB	2.65	3.73
4.6	4	16	4.9×10^{-3}	0.6	2.58	10.33

TABLE 6.2.2: Performance Results for the Schemes Using the Static Expansion Limitation Method for Code 1 at a Signal to Noise Ratio (E_b/N_O) of 5dB

C_m	Sv	cth( )			B.E.R.	DEGRADATION IN TOLERANCE TO NOISE CF VITERBI DETECTION (CODE 1) (dB)	AVERAGE NUMBER OF VALID VECTORS (a)	AVERAGE NUMBER OF VALID EXPANDED VECTORS PER SYMBOL INTERVAL (b)	b/a
		cth(1)	cth(2)	cth(3)					
5.172	16	0.75	2.586	4.422	1.5×10^{-3}	0.4	2.89	8.52	2.95
5.172	16	1.67	2.586	4.422	1.5×10^{-3}	0.4	2.89	8.58	2.97
5.172	16	0	2.586	4.422	1.5×10^{-3}	0.4	2.89	8.49	2.94
5.172	16	0	0	3	1.5×10^{-3}	0.4	2.89	7.4	2.56
5.172	16	0	0	1	1.7×10^{-3}	0.48	2.87	6.99	2.44
5.172	16	0	0	0.5	1.85×10^{-3}	0.51	2.86	6.93	2.42
5.172	16	0	0	0	2.1×10^{-3}	0.55	2.85	6.88	2.41
5.172	16	-1	-1	3	1.7×10^{-3}	0.48	2.86	6.19	2.16
5.172	16	-1	0	1	1.8×10^{-3}	0.5	2.86	6.16	2.15
5.172	16	-1	-1	1	1.89×10^{-3}	0.52	2.84	5.78	2.04
5.172	16	-1	-1	-1	1.1×10^{-2}	1.25	2.39	3.32	1.39
4.5	16	0	0	3	2.9×10^{-3}	0.68	2.25	6.41	2.85
4.5	16	-1	-1	3	3.2×10^{-3}	0.72	2.24	5.21	2.33

TABLE 6.2.3: Performance Results for the Dynamic Expansion Limitation Method for Code 1 at a Signal to Noise Ratio (E_b/N_O) of 4.6dB

C_m	Sv	cth( )			B.E.R.	DEGRADATION IN TOLERANCE TO NOISE CF VITERBI DETECTION (CODE 3) dB	AVERAGE NUMBER OF VALID VECTORS (a)	AVERAGE NUMBER OF VALID EXPANDED VECTORS PER SYMBOL INTERVAL (b)	b/a
		1	2	3					
8.7	64	1.26	4.35	7.44	4.4×10^{-3}	0.2	19.23	46.5	2.42
6.344	64	0.92	3.172	5.425	4.8×10^{-3}	0.25	7.96	20.6	2.59
5.6	64	0.81	2.8	4.79	5.7×10^{-3}	0.3	5.74	15.48	2.7
4.8	64	0.7	2.4	4.1	8.6×10^{-3}	0.45	4	11.44	2.86
4	64	0.58	2.0	3.42	1.6×10^{-2}	0.7	2.85	8.74	3.07

TABLE 6.2.4: Performance Results for the Dynamic Expansion Limitation Method for Code 3 at a Signal to Noise Ratio (E_b/N_O) of 4.6dB

C_m	Sv	cth( )			B.E.R.	DEGRADATION IN TOLERANCE TO NOISE CF VITERBI DETECTION (CODE 3) (dB)	AVERAGE NUMBER OF VALID VECTORS (a)	AVERAGE NUMBER OF VALID EXPANDED VECTORS PER SYMBOL INTERVAL (b)	b/a
		1	2	3					
8.7	64	1.26	4.35	7.44	1.7×10^{-3}	0.34	16.7	39.9	2.39
6.344	64	0.92	3.172	5.425	1.86×10^{-3}	0.37	6.66	17.3	2.6
6.344	64	0.92	4.3	5.425	1.86×10^{-3}	0.37	6.66	18.31	2.75
6.344	64	0.92	3.172	5.895	1.86×10^{-3}	0.37	6.66	18.12	2.72
6.344	64	2.0	3.172	5.425	1.86×10^{-3}	0.37	6.66	17.4	2.61
6.344	64	0	3.172	5.425	1.87×10^{-3}	0.37	6.66	17.18	2.58
6.344	64	0.92	2.0	5.425	1.86×10^{-3}	0.37	6.66	16.69	2.51
6.344	64	0.92	3.172	4.3	1.86×10^{-3}	0.37	6.66	15.5	2.33
6.344	64	0	0	4.3	1.88×10^{-3}	0.37	6.65	14.5	2.18
6.344	64	0	0	3	1.88×10^{-3}	0.37	6.61	13.28	2.01
6.344	64	0	0	1.5	1.95×10^{-3}	0.39	6.5	12.49	1.92
5.6	64	0.812	2.8	4.79	2.32×10^{-3}	0.43	4.79	13.03	2.72
4.8	64	0.7	2.4	4.1	3.77×10^{-3}	0.57	3.36	9.77	2.91
4.8	24	0.7	2.4	4.1	3.85×10^{-3}	0.58	3.36	9.76	2.9
4.8	16	0.7	2.4	4.1	4.1×10^{-3}	0.6	3.33	9.7	2.91
4	64	0.58	2	3.42	8×10^{-3}	0.83	2.43	7.61	3.13

TABLE 6.2.5: Performance Results for the Dynamic Expansion Limitation Method for Code 3 at a Signal to Noise Ratio (E_b/N_O) of 5dB

C_m	Sv	cth( )			B.E.R.	DEGRADATION IN TOLERANCE TO NOISE CF VITERBI DETECTION (CODE 3) (dB)	AVERAGE NUMBER OF VALID VECTORS (a)	AVERAGE NUMBER OF VALID EXPANDED VECTORS PER SYMBOL INTERVAL (b)	b/a
		1	2	3					
9.3	64	1.35	4.65	7.95	9.3×10^{-4}	0.42	18.68	43.99	2.35
9.9	64	1.44	4.95	8.47	9.3×10^{-4}	0.42	22.36	52.45	2.35
12	64	1.74	6	10.26	9×10^{-4}	0.4	36.5	86.4	2.37

TABLE 6.2.6: Performance Results for the Dynamic Expansion Limitation Method for Code 3 at a Signal to Noise Ratio (E_b/N_O) of 5.25dB

E_b/N_O (dB)	B.E.R.	AVERAGE NUMBER OF BIT ERRORS PER BURST	DEGRADATION IN TOLERANCE TO NOISE CF VITERBI DETECTION (CODE 3) (dB)	AVERAGE NUMBER OF VALID VECTORS (a)	AVERAGE NUMBER OF VALID EXPANDED VECTORS PER SYMBOL INTERVAL (b)	b/a
3.75	2.69×10^{-2}	29.8	0.1	13.8	25.9	1.88
4.25	9.16×10^{-3}	26.7	0.13	10.1	19.3	1.91
4.75	1.84×10^{-3}	21.3	0.1	7.4	14.7	1.99
5.35	2.44×10^{-3}	15.3	0.2	5.5	11.4	2.07

TABLE 6.2.7: Performance Results for a Scheme Using Code 2 where C_m=6.344 and cth=3,0,0.

C_m	Sv	cth( )			B.E.R.	DEGRADATION IN TOLERANCE TO NOISE CF VITERBI DETECTION (CODE 3) (dB)	AVERAGE NUMBER OF VALID VECTORS (a)	AVERAGE NUMBER OF VALID EXPANDED VECTORS PER SYMBOL INTERVAL (b)	b/a
		1	2	3					
6.344	64	0.92	3.172	5.425	4.8×10^{-4}	0.25	5.79	14.99	2.59
6.344	64	0	0	1	9.6×10^{-4}	0.44	5.68	11.24	1.98
6.344	64	-1	-1	1	1.86×10^{-3}	0.6	5.59	9.92	1.77
5.6	64	0.812	2.8	4.78	1.33×10^{-3}	0.5	4.14	11.35	2.74
5.6	64	0	0	3	1.35×10^{-3}	0.52	4.12	9.38	2.28
5.6	64	-1	-1	3	2.55×10^{-3}	0.7	4.1	8.17	1.99
4.8	64	0.7	2.4	4.1	5.9×10^{-3}	0.96	2.98	8.8	2.95
4.8	64	0	0	3	5.94×10^{-3}	0.96	2.97	7.67	2.58
4.8	64	-1	-1	3	8.26×10^{-3}	1.32	2.99	6.54	2.19

TABLE 6.2.8: Performance Results for the Dynamic Expansion Limitation Method for Code 4 at a Signal to Noise Ratio (E_b/N_O) of 5.25dB

Code	C_m	Sv	cth( )		
			cth(1)	cth(2)	cth(3)
1	5.172	16	0.75	2.586	4.422
1	5.172	16	-1	-1	1
4	6.344	64	0.92	3.172	5.425

TABLE 6.2.9: Schemes Chosen for Full Simulation Tests

E_b/N_o dB	B.E.R.	AVERAGE NUMBER OF BIT ERRORS PER BURST	DEGRADATION IN TOLERANCE TO NOISE CF VITERBI DETECTION (CODE 1) dB	AVERAGE NUMBER OF VALID VECTORS (a)	AVERAGE NUMBER OF VALID EXPANDED VECTORS PER SYMBOL INTERVAL (b)	b/a
3.25	7.07×10^{-2}	23.6	0.2	6.7	19.2	2.87
3.5	4.78×10^{-2}	21.8	0.23	6.0	17.2	2.87
3.75	2.86×10^{-2}	19.3	0.18	5.4	15.5	2.87
4.0	1.99×10^{-2}	18.9	0.25	4.9	14	2.86
4.25	1.04×10^{-2}	17.3	0.19	4.4	12.7	2.89
4.5	7.59×10^{-3}	18.1	0.29	4.0	11.4	2.85
4.75	3.78×10^{-3}	16.2	0.25	3.6	10.3	2.86
5.0	2.46×10^{-3}	17.8	0.3	3.2	9.5	2.97
5.3	1.5×10^{-3}	15.9	0.4	2.9	8.5	2.93
5.6	3.9×10^{-4}	12.9	0.4	2.6	7.8	3.0
5.8	2.9×10^{-4}	15.5	0.44	2.5	7.4	2.96

TABLE 6.2.10: Performance Results for the Scheme Using Code 1 where C_m=5.172 and cth=4.422, 2.586, 0.75

E_b/N_O (dB)	B.E.R.	AVERAGE NUMBER OF BIT ERRORS PER BURST	DEGRADATION IN TOLERANCE TO NOISE CF VITERBI DETECTION (CODE 1) dB	AVERAGE NUMBER OF VALID VECTORS (a)	AVERAGE NUMBER OF VALID EXPANDED VECTORS PER SYMBOL INTERVAL (b)	b/a
3.25	7.65×10^{-2}	24.7	0.23	6.5	11.5	1.77
3.5	5.22×10^{-2}	22.8	0.26	5.8	10.4	1.79
3.75	3.31×10^{-2}	20.4	0.25	5.3	9.6	1.81
4.0	2.35×10^{-2}	20.4	0.3	4.8	8.8	1.83
4.25	1.27×10^{-2}	18.2	0.3	4.3	8.1	1.88
4.5	9.36×10^{-3}	19.8	0.33	3.9	7.4	1.9
4.75	4.76×10^{-3}	17.4	0.35	3.5	6.8	1.94
5.0	3.19×10^{-3}	18.9	0.4	3.2	6.3	1.97
5.3	1.89×10^{-3}	16.5	0.49	2.8	5.8	2.07
5.6	6.8×10^{-4}	16.4	0.62	2.6	5.4	2.08
5.8	5.26×10^{-4}	18.5	0.62	2.4	5.2	2.17

TABLE 6.2.11: Performance Results for the Scheme Using Code 1 where C_m=5.172 and cth=1,-1,-1

E_b/N_O (dB)	B.E.R.	AVERAGE NUMBER OF BIT ERRORS PER BURST	DEGRADATION IN TOLERANCE TO NOISE CF VITERBI DETECTION (CODE 3) dB	AVERAGE NUMBER OF VALID VECTORS (a)	AVERAGE NUMBER OF VALID EXPANDED VECTORS PER SYMBOL INTERVAL (b)	b/a
3.75	2.62×10^{-2}	28.7	0.13	14.5	37.3	2.57
4.0	1.28×10^{-2}	25.4	0	12.1	31	2.56
4.25	8.01×10^{-3}	27.2	0.1	10.4	26.7	2.57
4.5	3.81×10^{-3}	25.9	0.1	8.9	22.8	2.56
4.75	2.49×10^{-3}	27.1	0.18	7.7	19.8	2.57
5.0	1.17×10^{-3}	27.1	0.25	6.7	17.2	2.57
5.25	4.8×10^{-4}	18.5	0.3	5.8	15	2.59
5.5	3.4×10^{-4}	37.8	0.43	5.2	13.5	2.6
5.8	1.3×10^{-5}	6.5	0.4	4.5	11.9	2.64

TABLE 6.2.12: Performance Results for the Scheme Using Code 4 where C_m=6.344 and cth=5.425,3.172,0.92

E_b/N_O (dB)	B.E.R.	AVERAGE NUMBER OF BIT ERRORS PER BURST	DEGRADATION IN TOLERANCE TO NOISE CF VITERBI DETECTION (CODE 3) dB	AVERAGE NUMBER OF VALID VECTORS (a)	AVERAGE NUMBER OF VALID EXPANDED VECTORS PER SYMBOL INTERVAL (b)	b/a
3.75	9.23×10^{-2}	75.3	0.9	7.6	15.5	2.04
4.0	6.2×10^{-2}	75	0.9	6.4	13.1	2.05
4.25	4.83×10^{-2}	80.1	0.95	5.6	11.5	2.05
4.5	3.3×10^{-2}	86.3	1.0	4.7	9.9	2.11
4.75	1.83×10^{-2}	76.4	0.95	4.0	8.4	2.1
5.0	1.37×10^{-2}	84	1.05	3.4	7.4	2.18
5.25	8.26×10^{-3}	96.2	1.3	3.0	6.5	2.17
5.5	5.9×10^{-3}	107.4	1.2	2.7	5.9	2.19
5.65	3.43×10^{-3}	91.4	1.2	2.4	5.5	2.29

TABLE 6.2.13: Performance Results for the Scheme Using Code 4 where C_m=4.8, and cth=3,-1,-1

DESCRIPTION OF SCHEME					DEGRADATION IN TOLERANCE TO NOISE CF VITERBI DETECTION FOR CODE 1 (AT BER OF 6 in 10^4) (dB)	REDUCTION IN COMPLEXITY [cf 64 EXPANDED VECTORS PER SYMBOL INTERVAL FOR VITERBI DETECTION FOR CODE 1] (AT BER = 6 in 10^4)
Code	C_m	cth( )				
		cth(1)	cth(2)	cth(3)		
1		VITERBI			0.0	1:1
3		VITERBI			-0.25 (Gain)	1:4 (more complex)
1	5.172	0.75	2.586	4.422	0.4	7.8:1
1	5.172	-1	-1	1	0.6	11.8:1
4	6.344	0.92	3.172	5.425	0.1	4:1
4	6.344	-1	-1	3	0.95	>11:1

TABLE 6.2.14: Comparison of Performance and Complexity for the Chosen Configurations

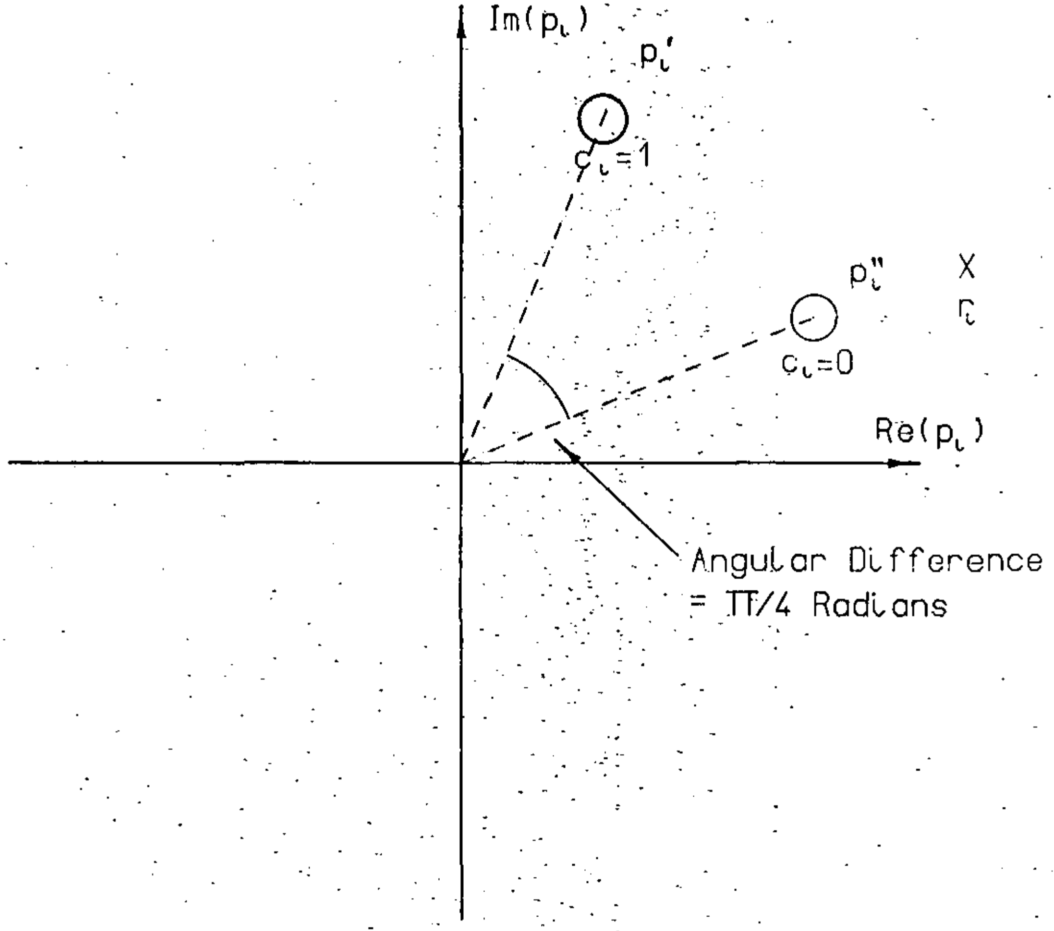
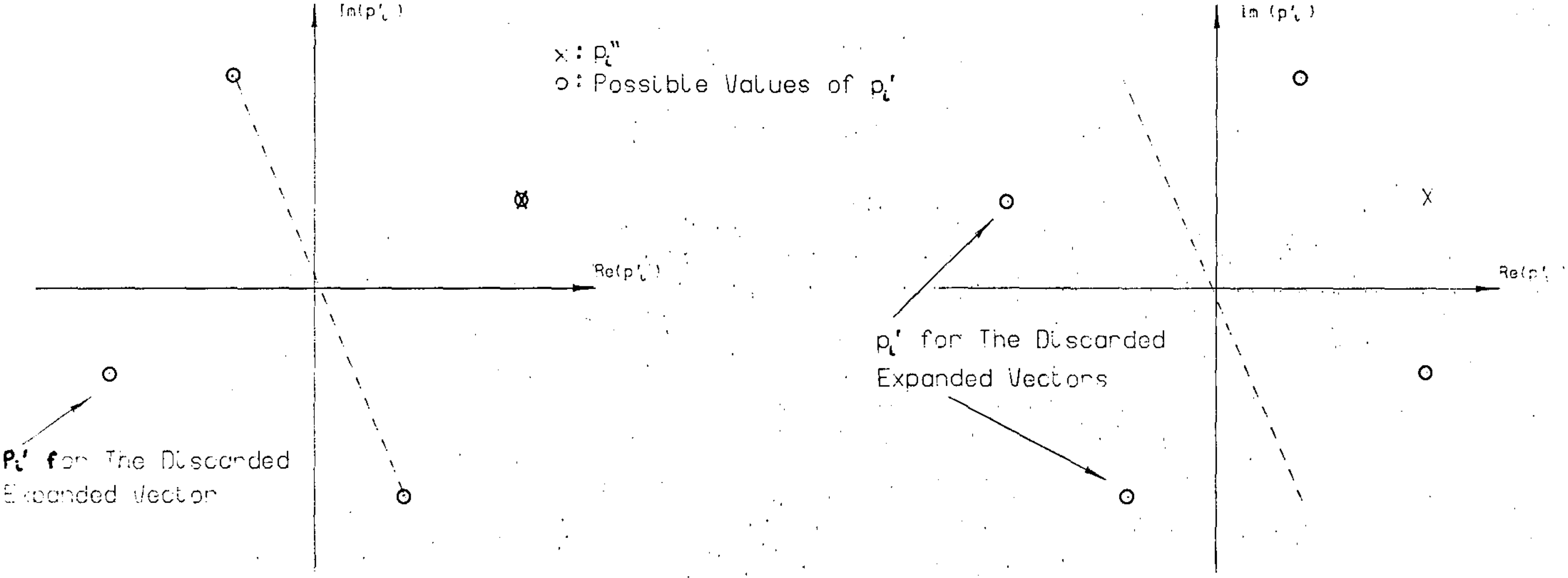


Figure 6.2.1 Determination of The Difference Between The Phase Angles of Points p_i'' and p_i'


Im(p'_i)
x : p_i''
o : Possible Values of p_i'
Re(p'_i)
p_i' for The Discarded Expanded Vector
Im (p'_i)
Re(p'_i)
p_i' for The Discarded Expanded Vectors
X

(a) Case where 3 Costs are Calculated

(b) Case where 2 Costs are Calculated

Figure 6.2.2 Illustration of The Number of Expanded Vectors Derived from One Vector when Δ =2

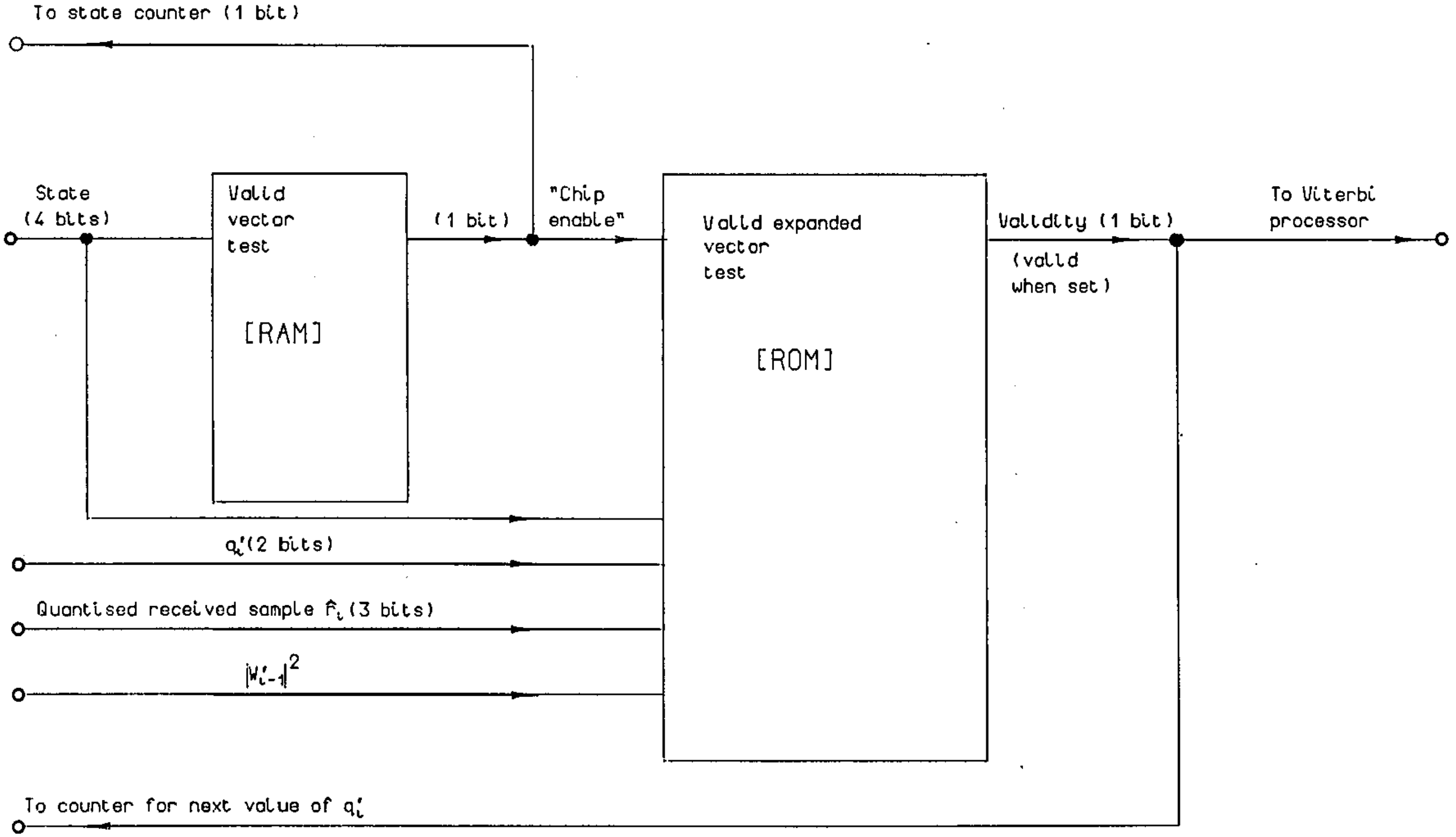
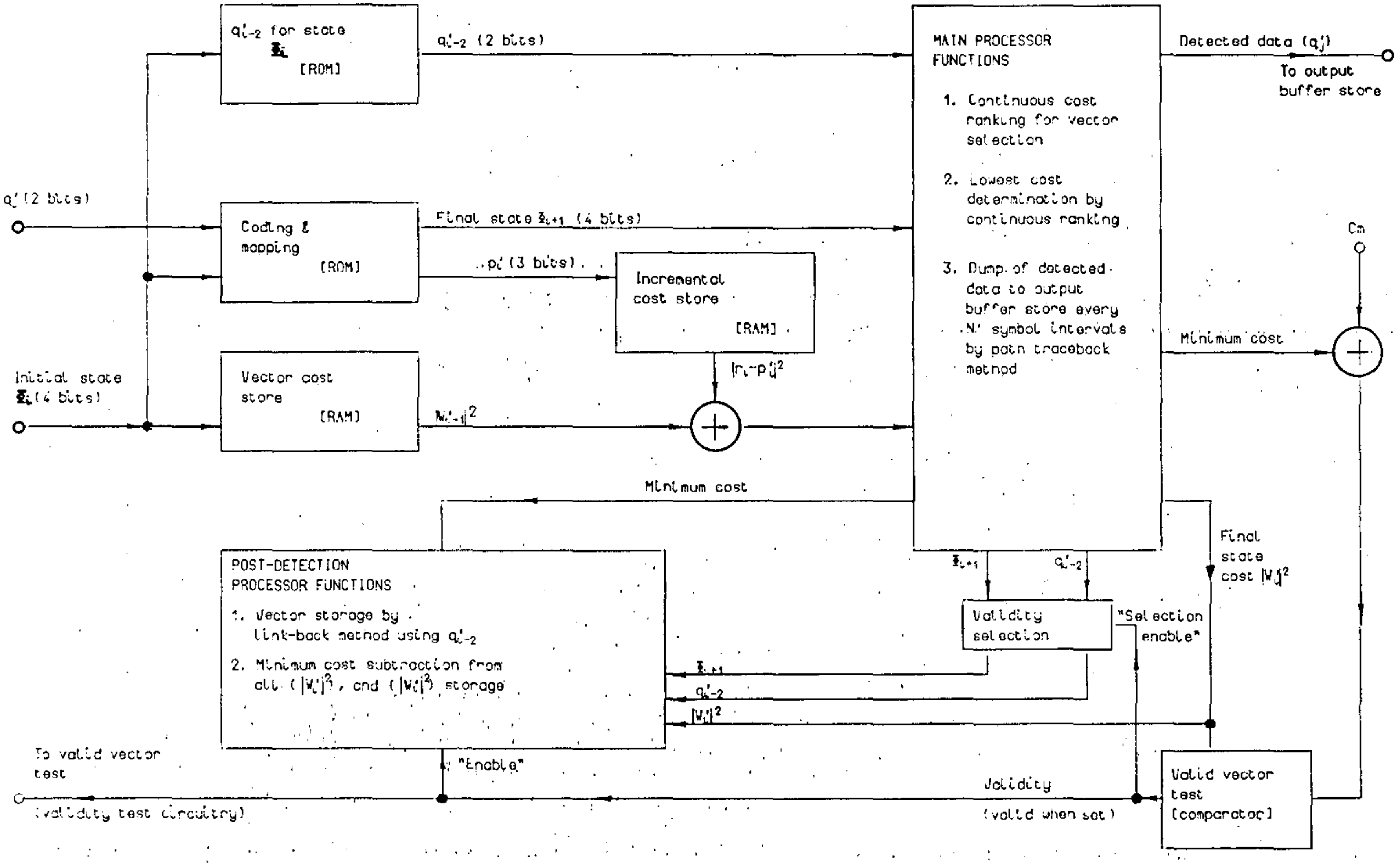


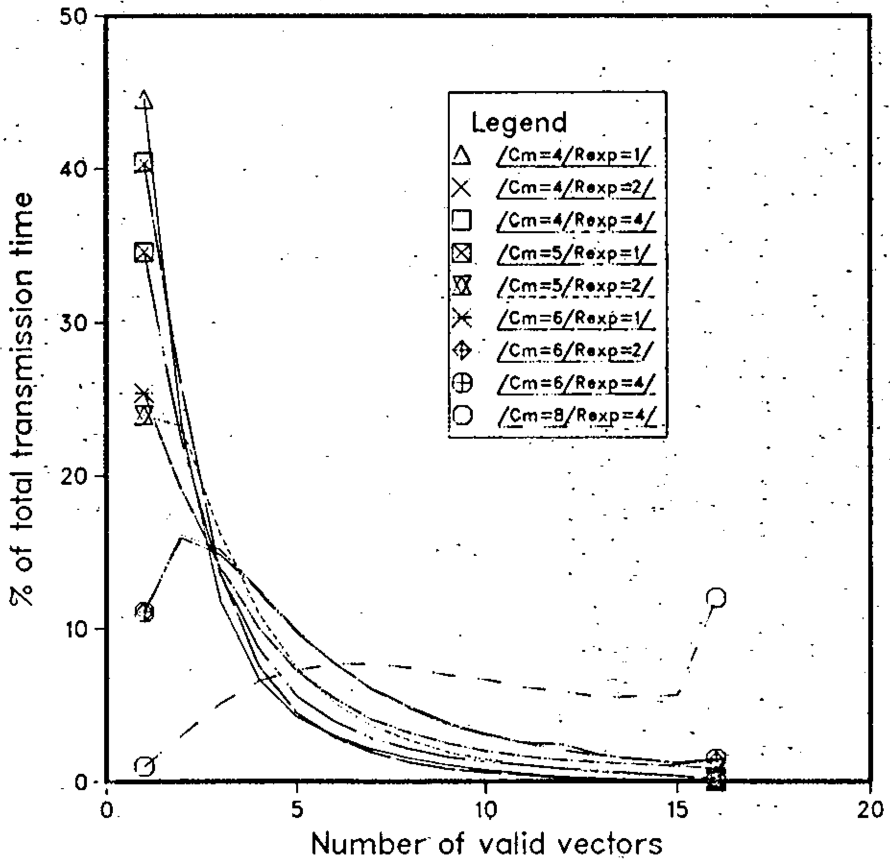
Figure 6.2.3. Validity Test Block Diagram



q'_{i-2} for state Φ_i [ROM]
q'_{i-2} (2 bits)
MAIN PROCESSOR FUNCTIONS
1. Continuous cost ranking for vector selection
2. Lowest cost determination by continuous ranking
3. Dump of detected data to output buffer store every N' symbol intervals by path traceback method
Detected data (q'_j)
To output buffer store
q'_i (2 bits)
Coding & mapping [ROM]
Final state Φ_{i+1} (4 bits)
p'_i (3 bits)
Incremental cost store [RAM]
$|r_i-p'_i|^2$
Cm
Minimum cost
Initial state Φ_i (4 bits)
Vector cost store [RAM]
$|W'_{i-1}|^2$
Minimum cost
POST-DETECTION PROCESSOR FUNCTIONS
1. Vector storage by link-back method using q'_{i-2}
2. Minimum cost subtraction from all ($|W'_i|^2$, and ($|W'_i|^2$) storage
Φ_{i+1}
q'_{i-2}
Validity selection
"Selection enable"
Final state cost $|W'_i|^2$
Φ_{i+1}
q'_{i-2}
$|W'_i|^2$
To valid vector test (validity test circuitry)
"Enable"
Validity (valid when set)
Valid vector test [comparator]

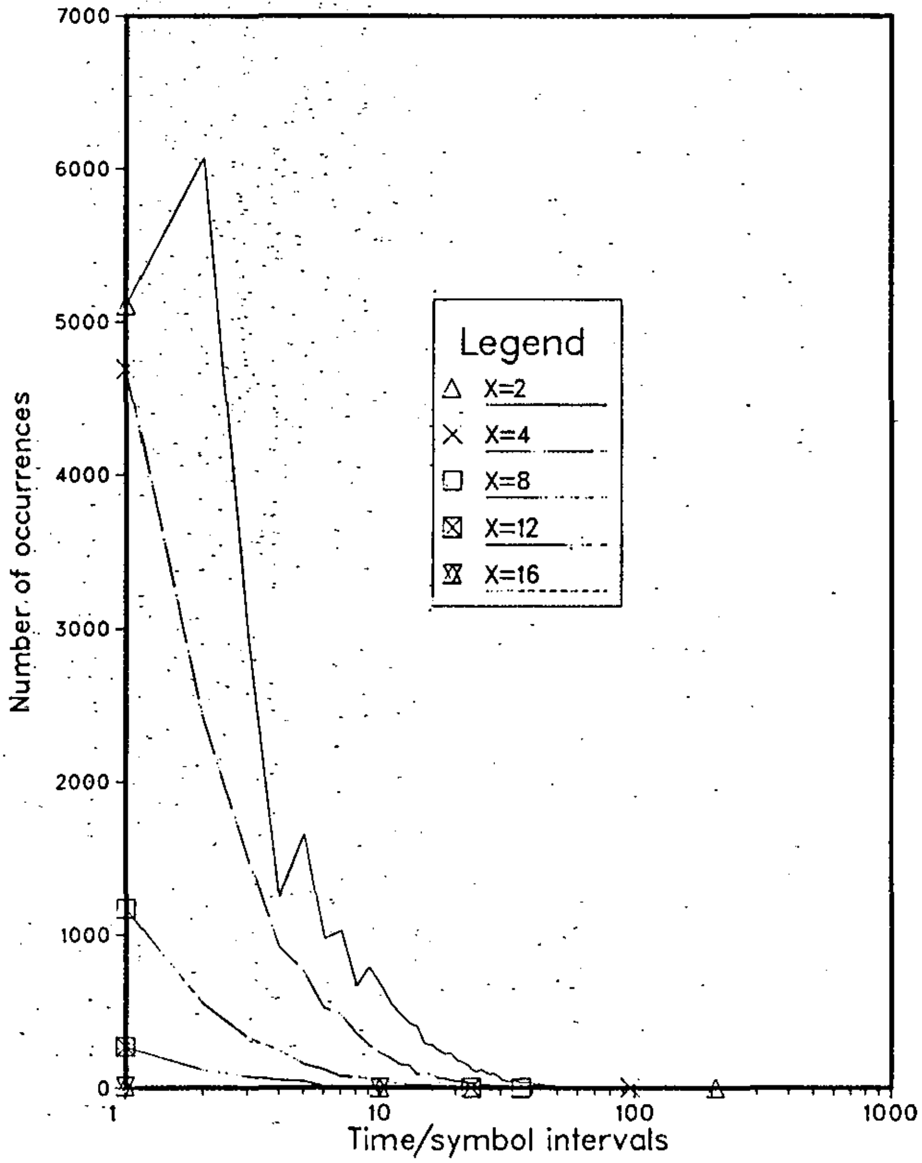
Figure 6.2.4 Viterbi Processor Block Diagram

Graph 6.2.1 Type–A Distribution. Static Expansion Limitation Method. Code 1. Eb/No=4.6dB



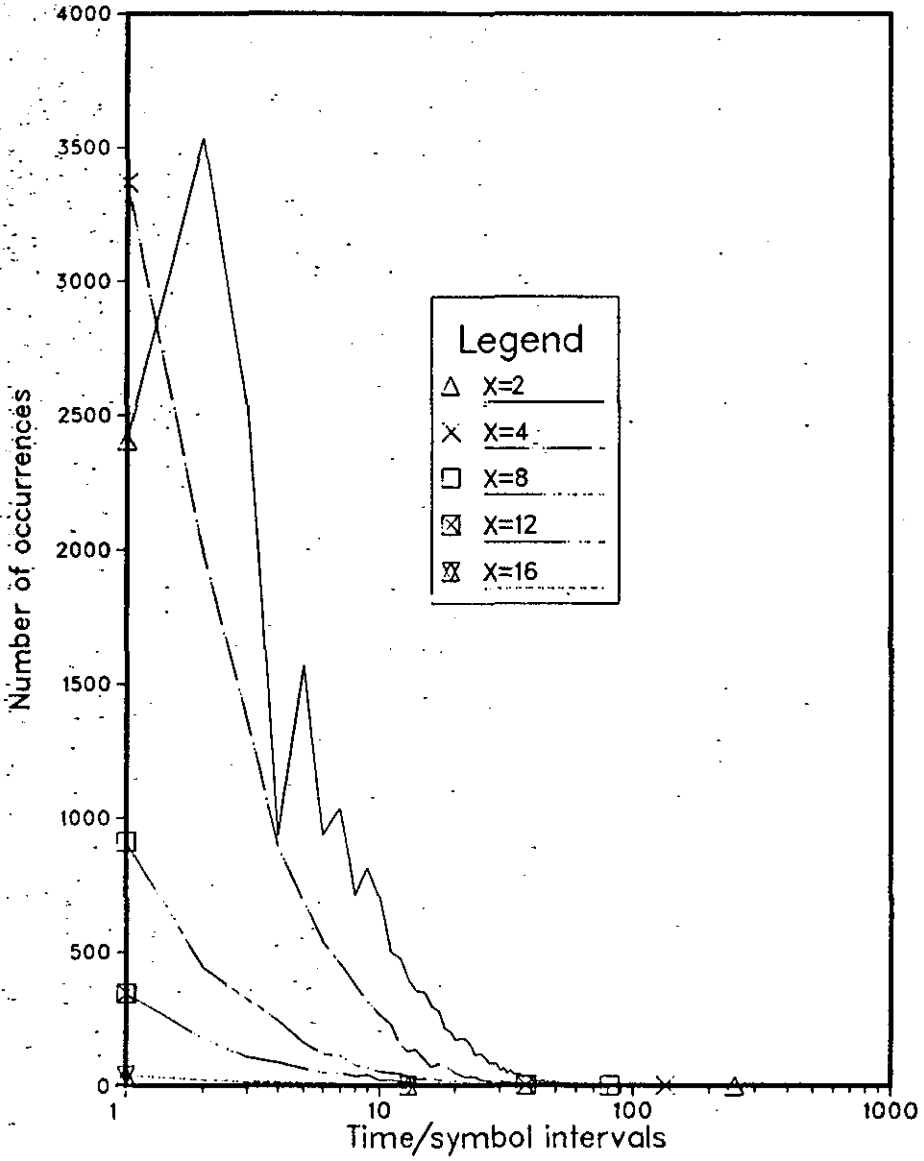
COMMON ATTRIBUTES
/M=8/C=1/Det=V16/N=64/

Graph 6.2.2 Type–B Distribution. Static Expansion Limitation Method. Code 1. Eb/No=4.6dB. BER=0.05



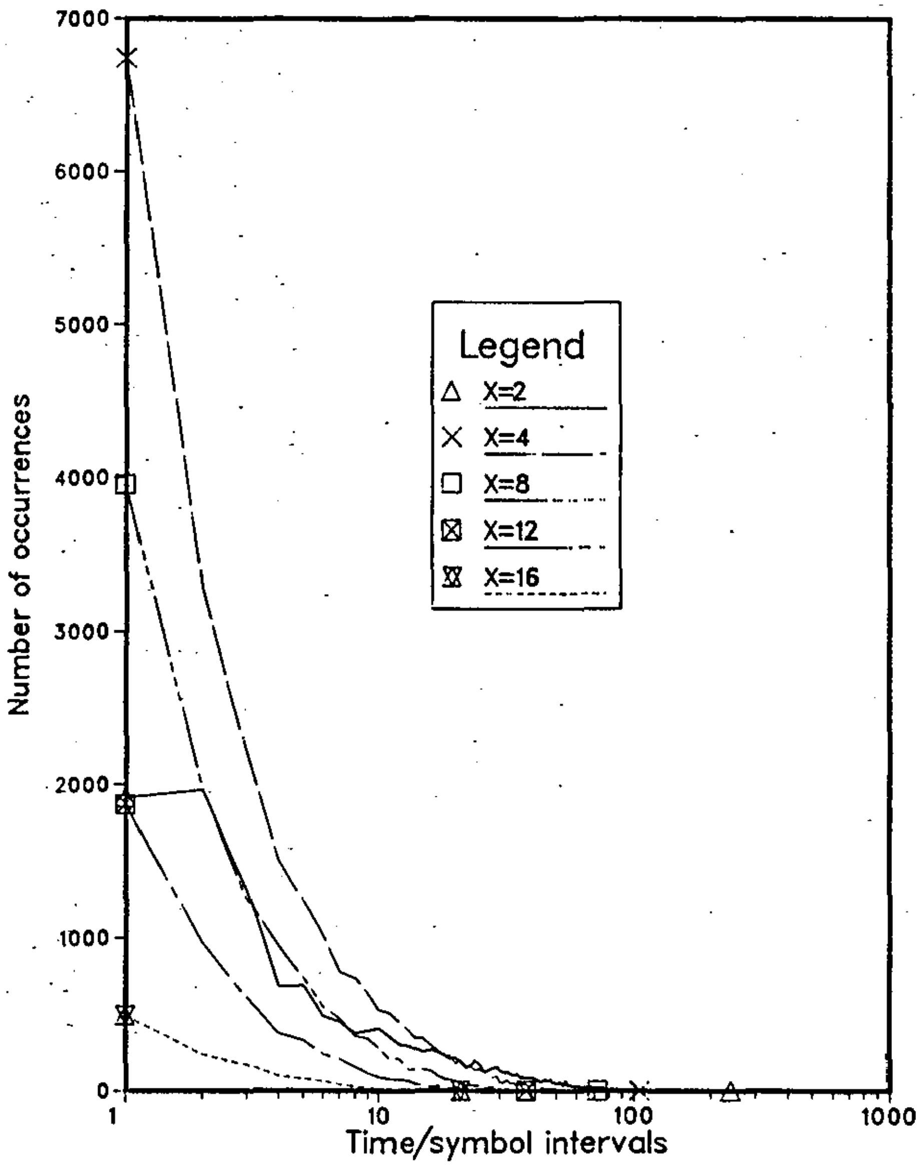
COMMON ATTRIBUTES
/M=8/C=1/Det=V16/N=64/Rexp=1/Cm=4/

Graph 6.2.3 Type–B Distribution. Static Expansion Limitation Method. Code 1 Eb/No=4.6dB BER=0.0175



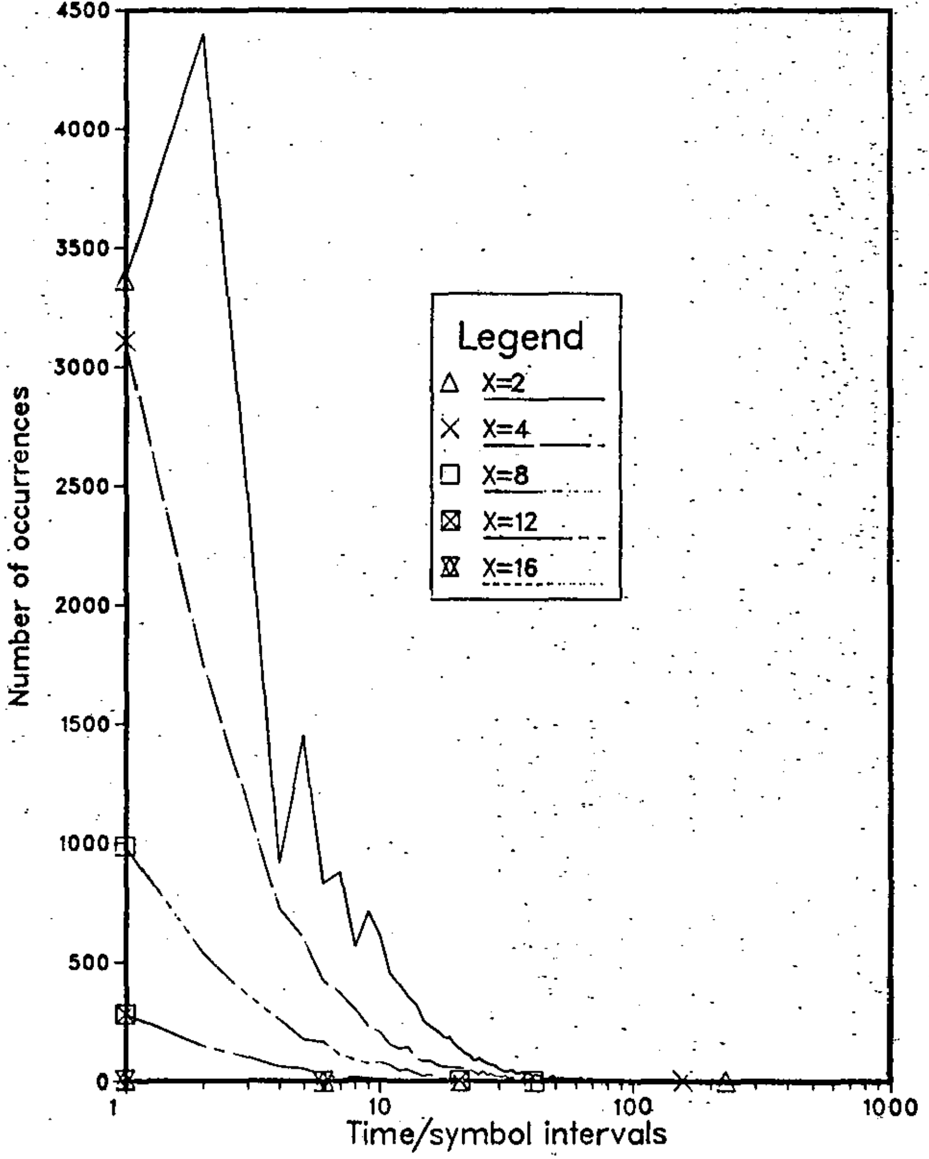
COMMON ATTRIBUTES
/M=8/C=1/Det=V16/N=64/Rexp=4/Cm=4/

Graph 6.2.4 Type–B Distribution. Static Expansion Limitation Method. Code 1 Eb/No=4.6dB BER=0.0184



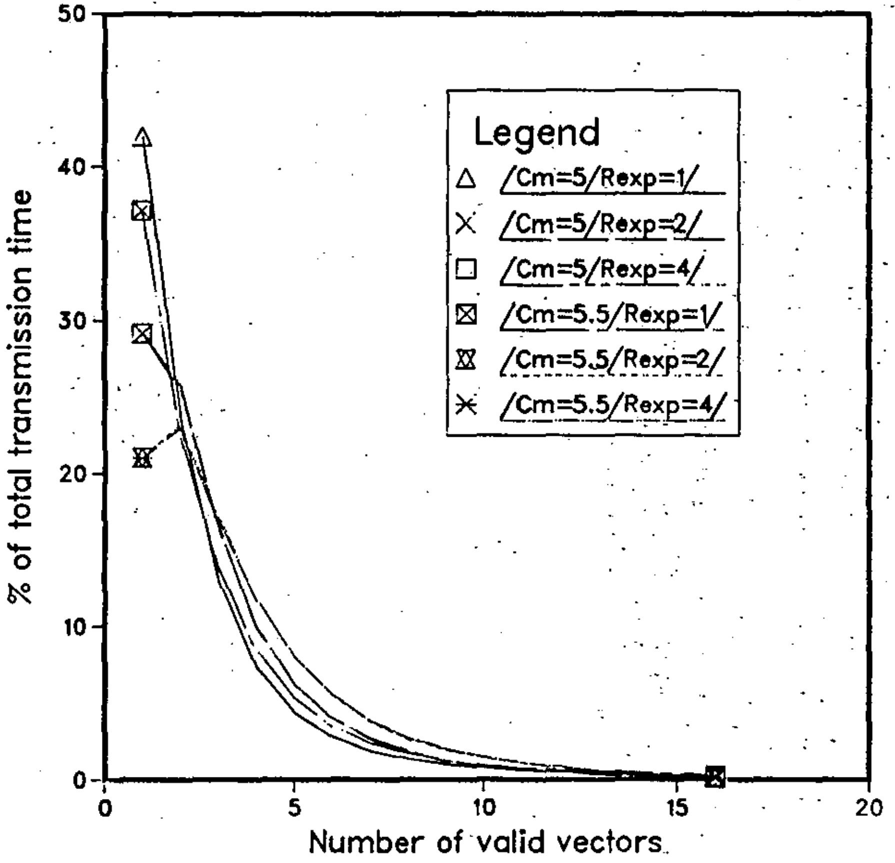
COMMON ATTRIBUTES
/M=8/C=1/Det=V16/N=64/Rexp=1/Cm=6/

Graph 6.2.5 Type–B Distribution. Static Expansion Limitation Method. Code 1 Eb/No=4.6dB BER=0.00433



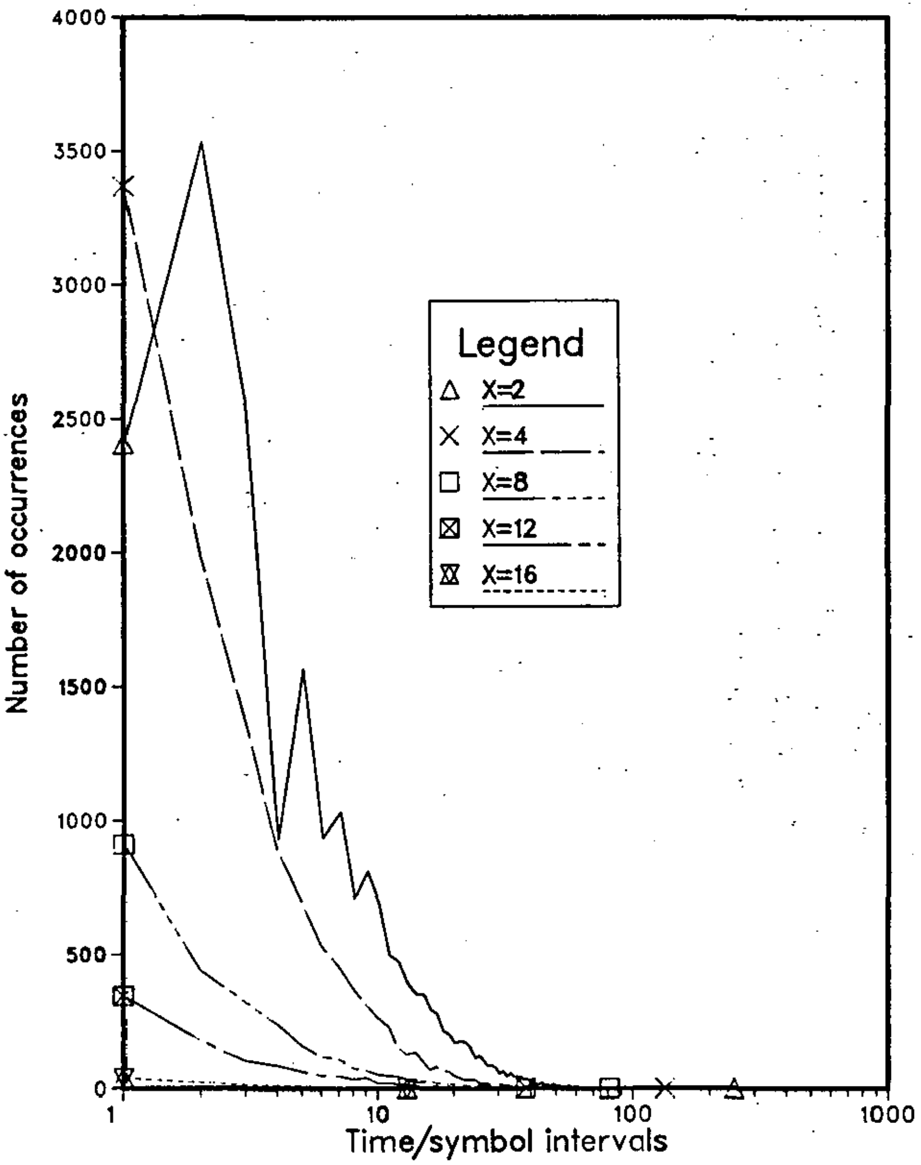
COMMON ATTRIBUTES
/M=8/C=1/Det=V16/N=64/Rexp=4/Cm=6/

Graph 6.2.6 Type–A Distribution. Static Expansion Limitation Method. Code 1 Eb/No=5dB



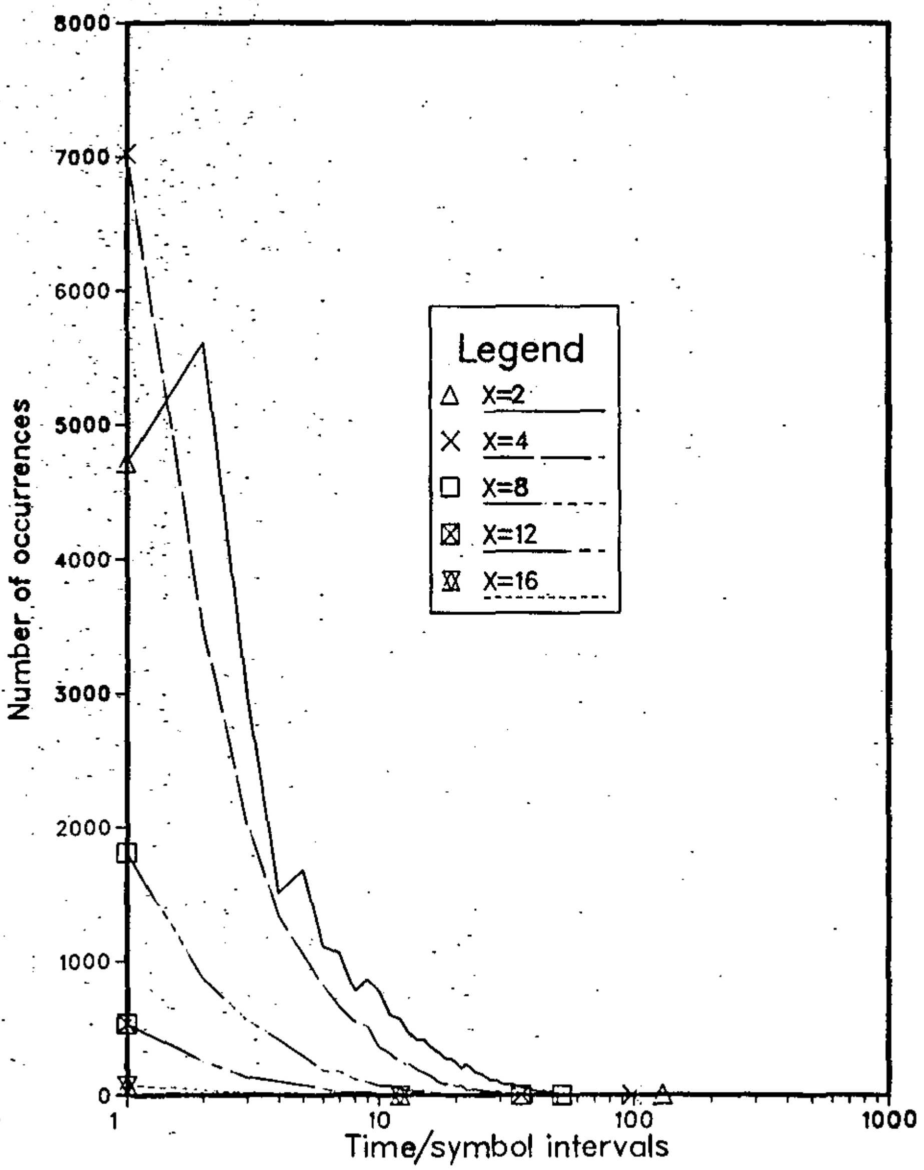
COMMON ATTRIBUTES
/M=8/C=1/Det=V16/N=64/

Graph 6.2.7 Type–B Distribution. Static Expansion Limitation Method. Code 1 Eb/No=5dB BER=0.0187



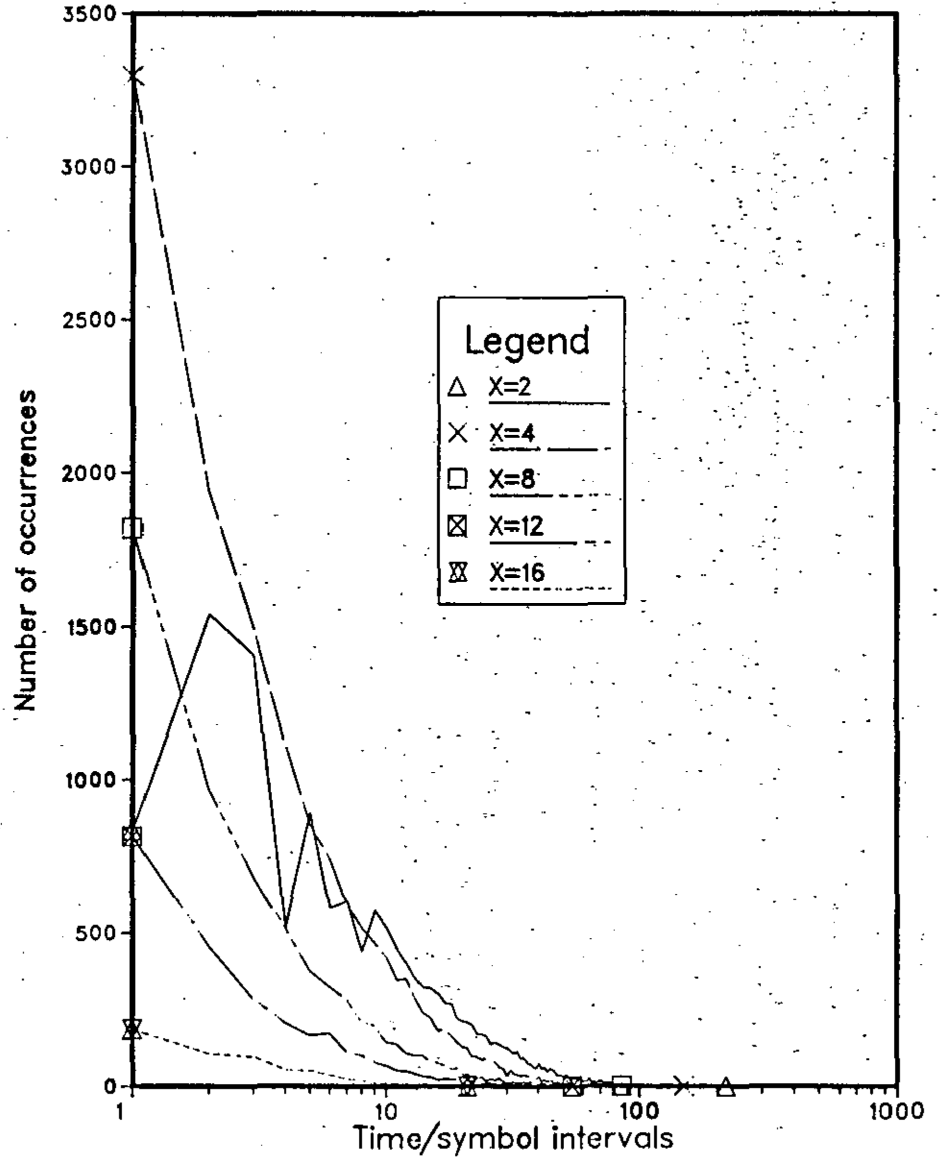
COMMON ATTRIBUTES
/M=8/C=1/Det=V16/N=64/Rexp=1/Cm=5/

Graph 6.2.8 Type–B Distribution. Static Expansion Limitation Method. Code 1 Eb/No=5dB BER=0.0029



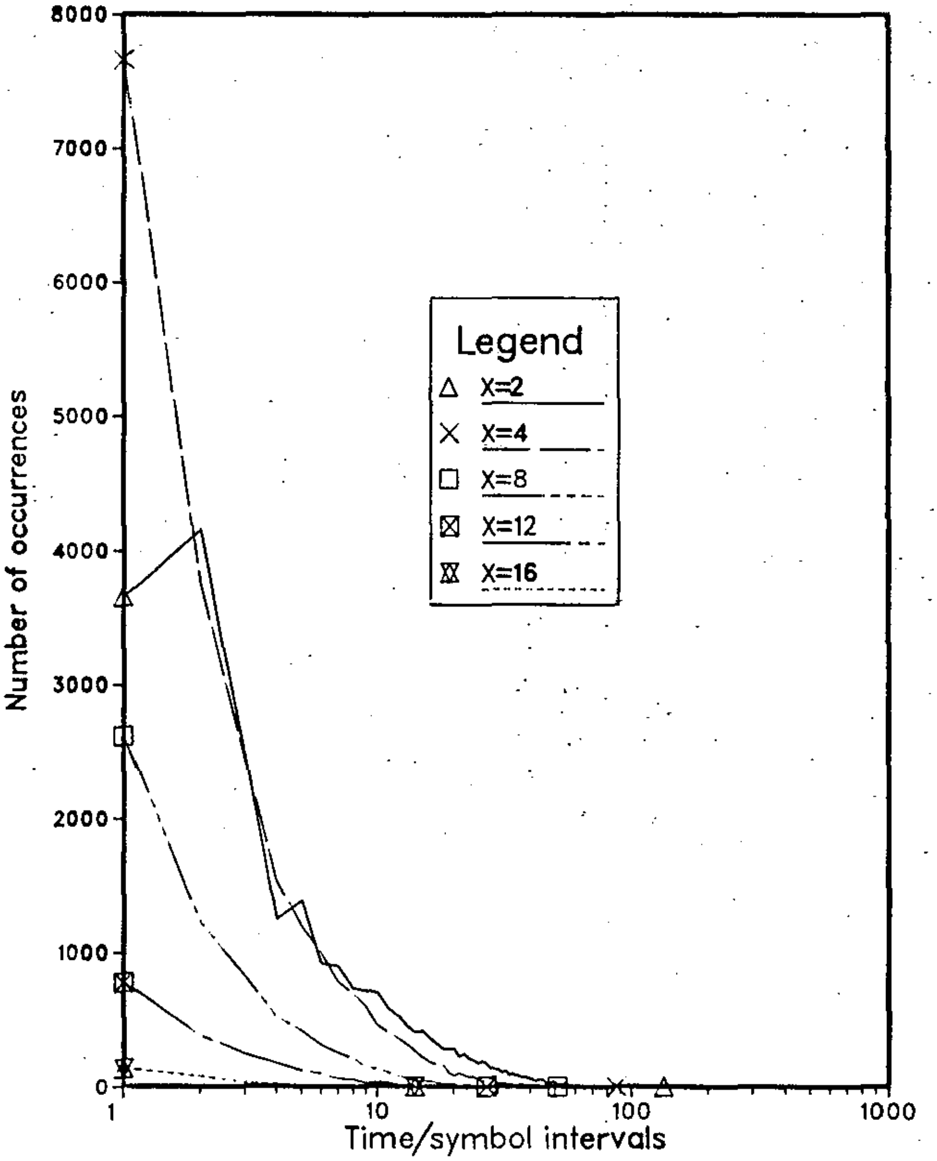
COMMON ATTRIBUTES
/M=8/C=1/Det=V16/N=64/Rexp=4/Cm=5/

Graph 6.2.9 Type–B Distribution. Static Expansion Limitation Method. Code 1 Eb/No=5dB BER=0.0134



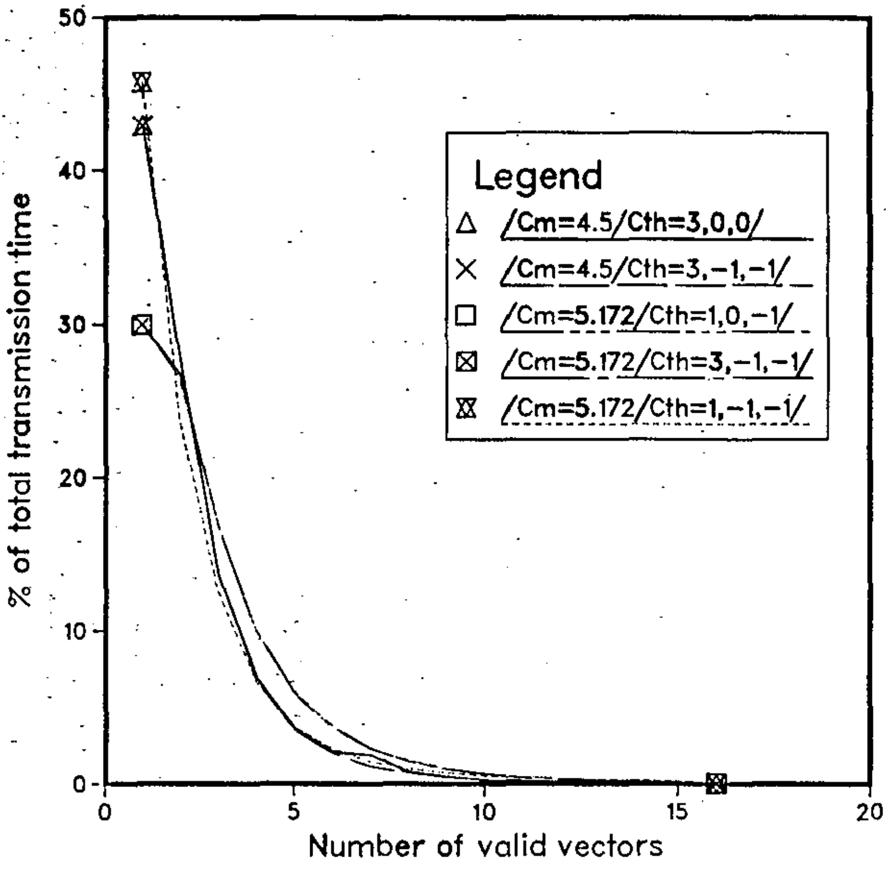
COMMON ATTRIBUTES
/M=8/C=1/Det=V16/N=64/Rexp=1/Cm=5.5/

Graph 6.2.10 Type–B Distribution. Static Expansion Limitation Method. Code 1 Eb/No=5dB BER=0.00181



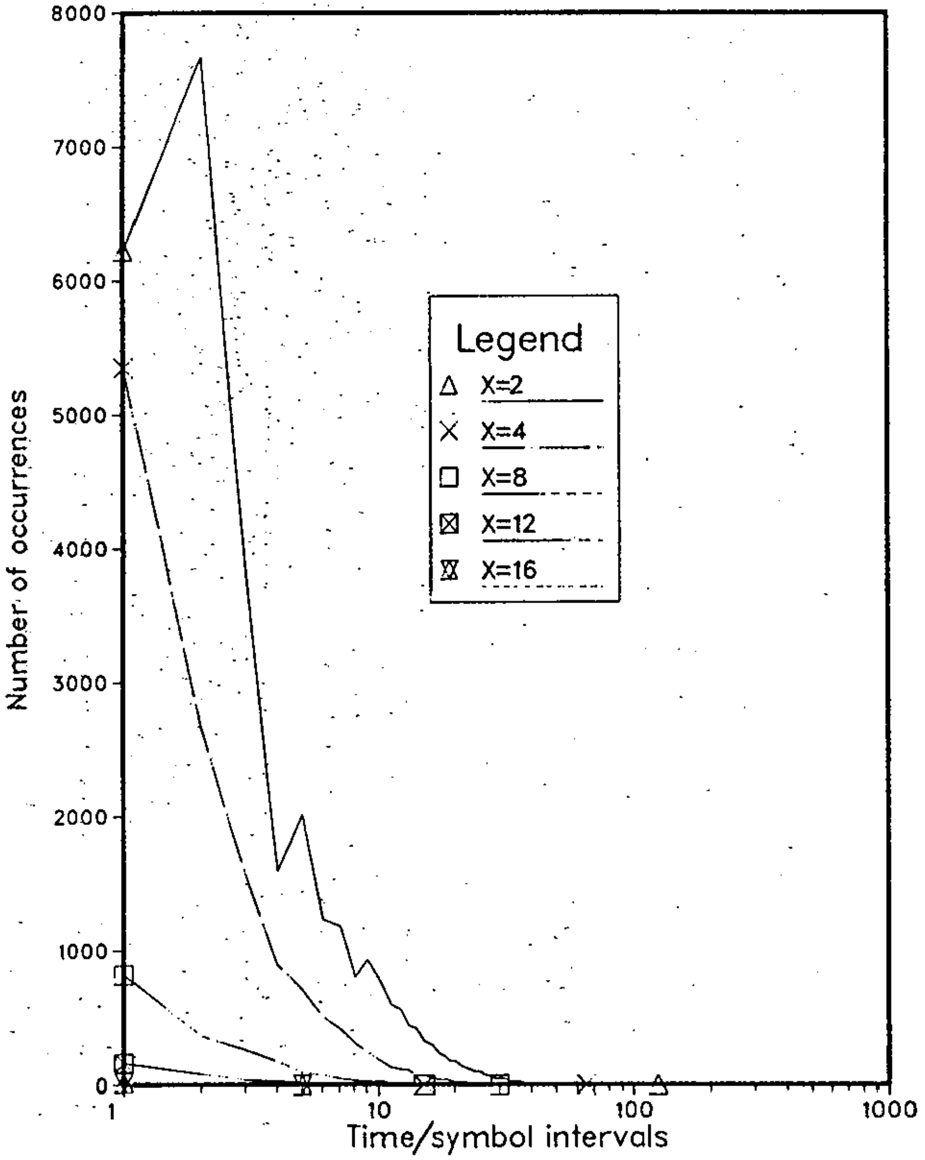
COMMON ATTRIBUTES
/M=8/C=1/Det=V16/N=64/Rexp=4/Cm=5.5/

Graph 6.2.11 Type–A Distribution. Dynamic Expansion Limitation Method. Code 1 Eb/No=5.3dB



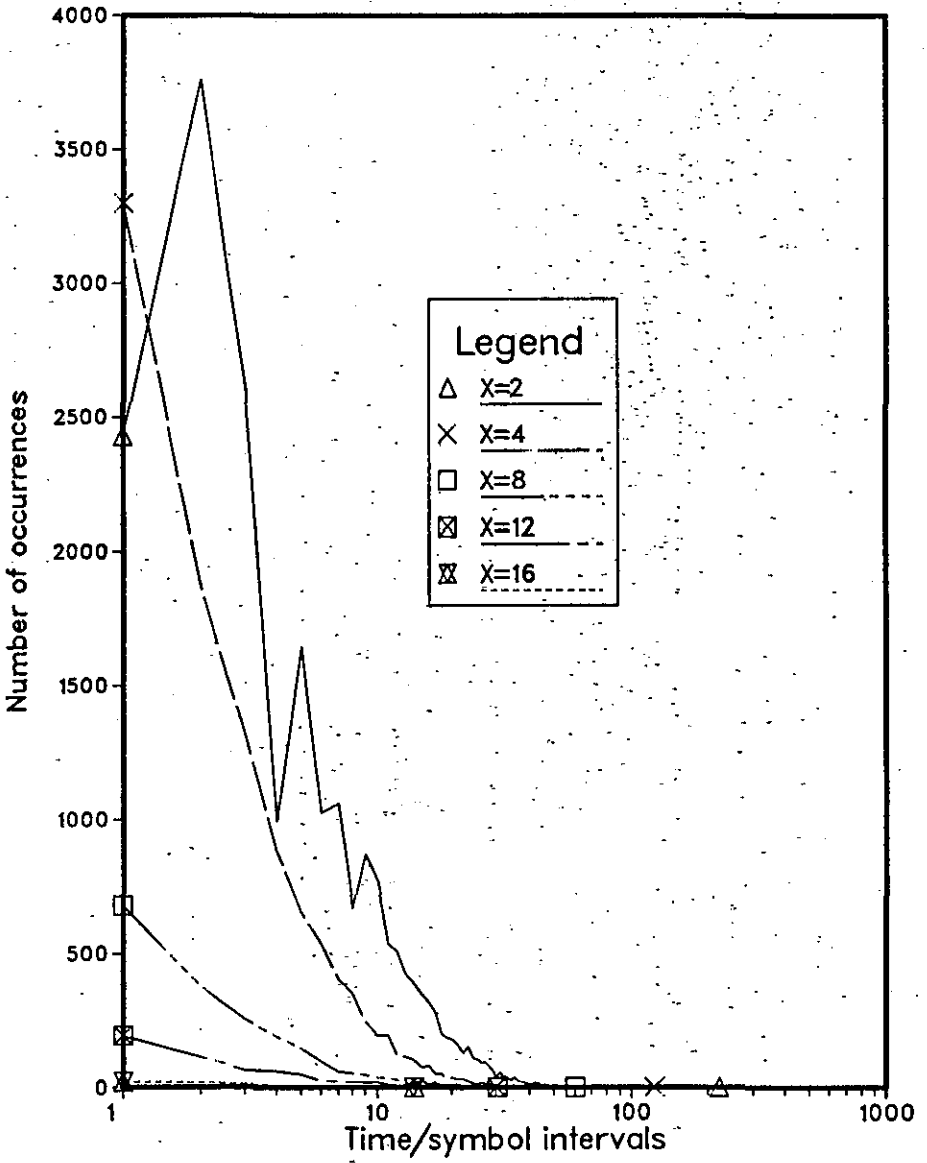
COMMON ATTRIBUTES
/M=8/C=1/Det=V16/N=64/

Graph 6.2.12 Type–B Distribution. Dynamic Expansion Limitation Method. Code 1 Eb/No=5.3dB BER=0.00289



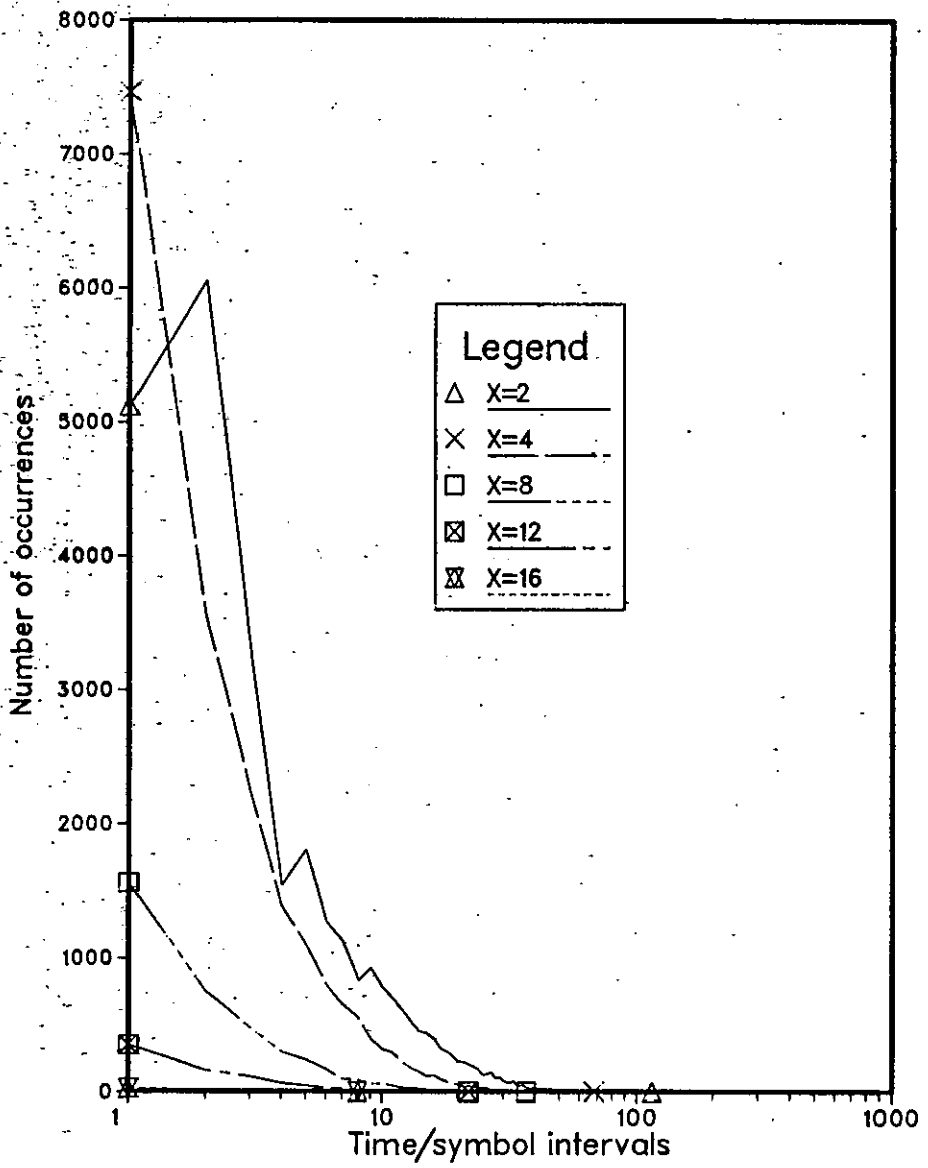
COMMON ATTRIBUTES
/M=8/C=1/Det=V16/N=64/Cth=3,0,0/Cm=4.5/

Graph 6.2.13 Type–B Distribution. Dynamic Expansion Limitation Method. Code 1 Eb/No=5.3dB BER=0.011



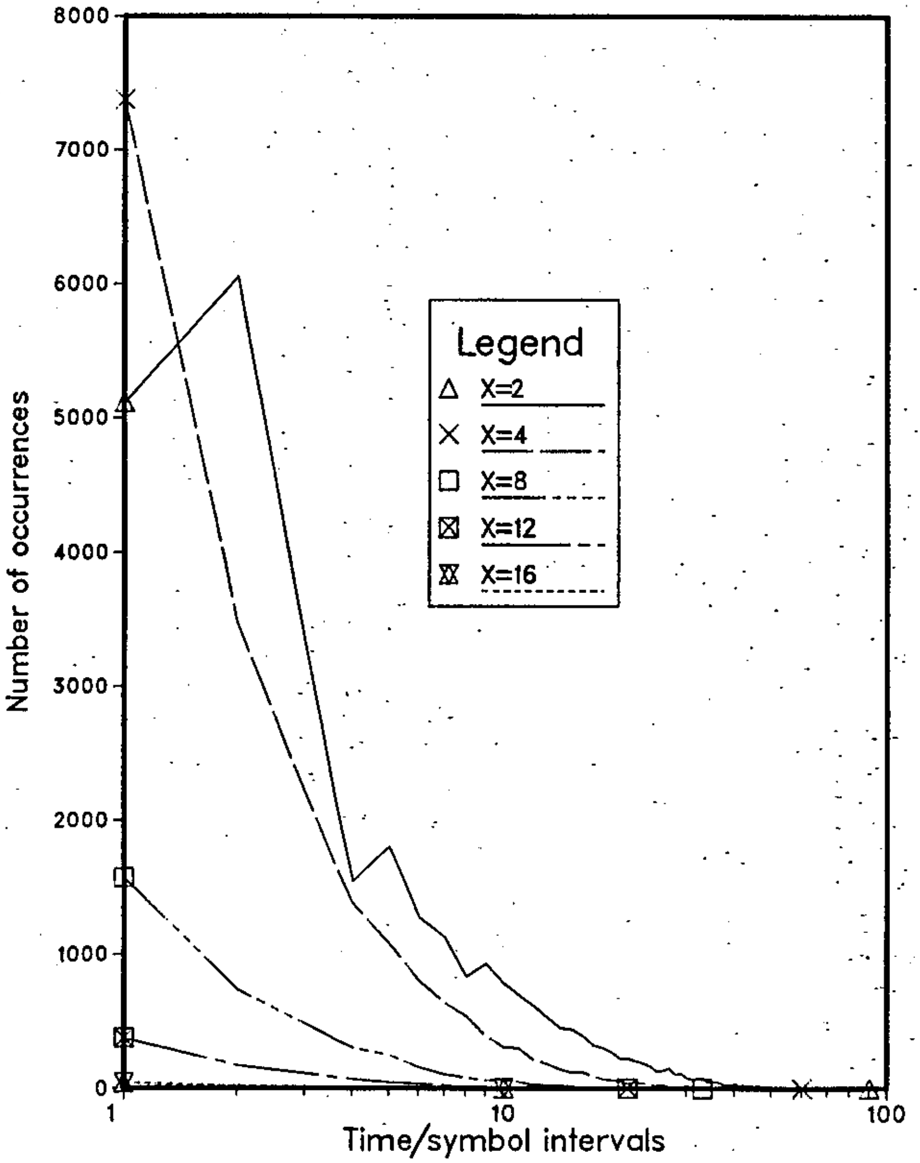
COMMON ATTRIBUTES
/M=8/C=1/Det=V16/N=64/
/Cth=-1,-1,-1/Cm=5.172/

Graph 6.2.14 Type–B Distribution. Dynamic Expansion Limitation Method. Code 1 Eb/No=5.3dB BER=0.00178



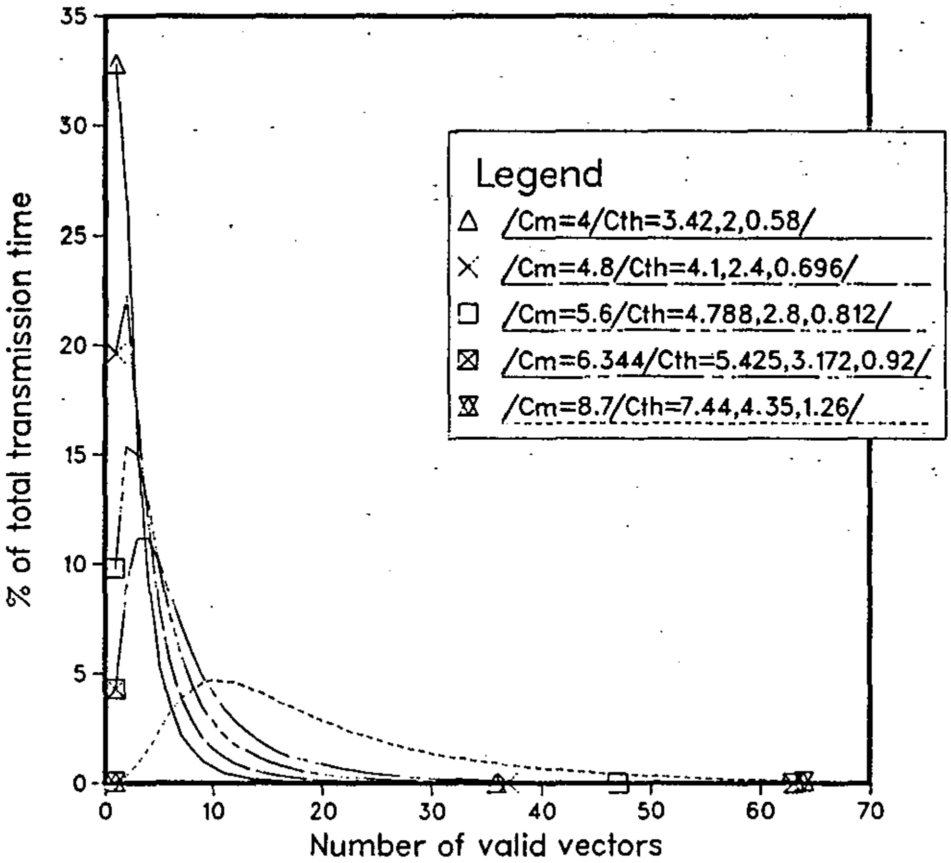
COMMON ATTRIBUTES
/M=8/C=1/Det=V16/N=64/
/Cth=1,0,−1/Cm=5.172/

Graph 6.2.15 Type–B Distribution. Dynamic Expansion Limitation Method. Code 1 Eb/No=5.3dB BER=0.0017



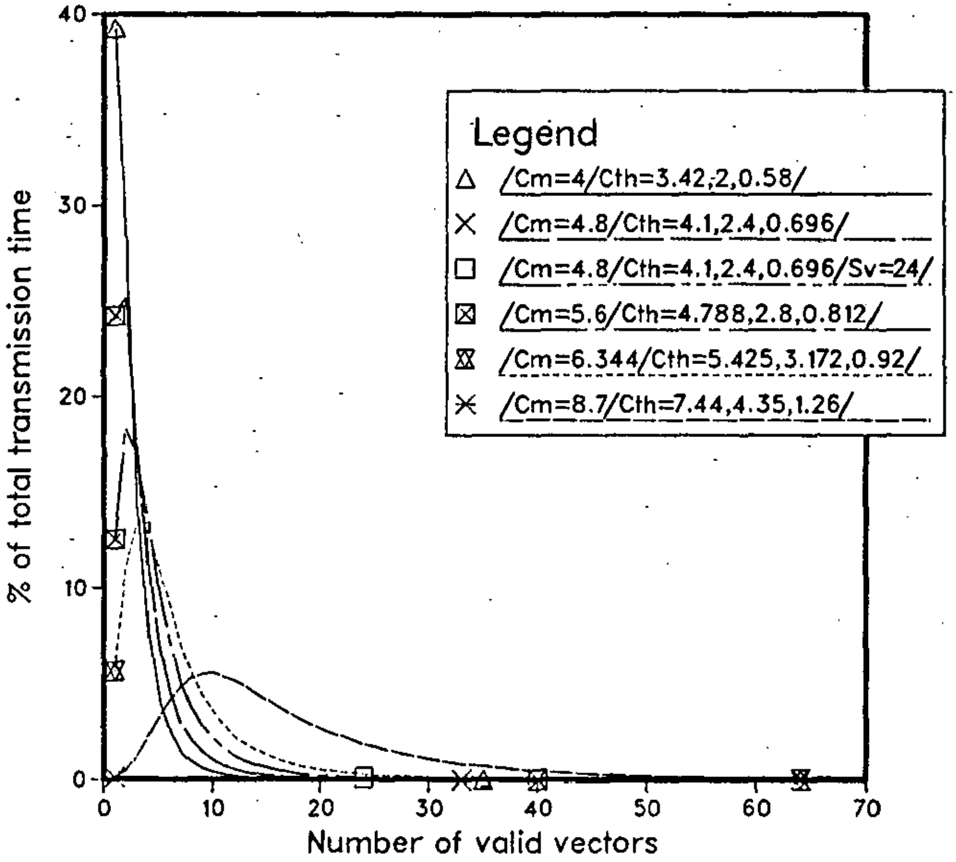
COMMON ATTRIBUTES
/M=8/C=1/Det=V16/N=64/
/Cth=3,-1,-1/Cm=5.172/

Graph 6.2.16 Type–A Distribution. Dynamic Expansion Limitation Method. Code 3 Eb/No=4.6dB



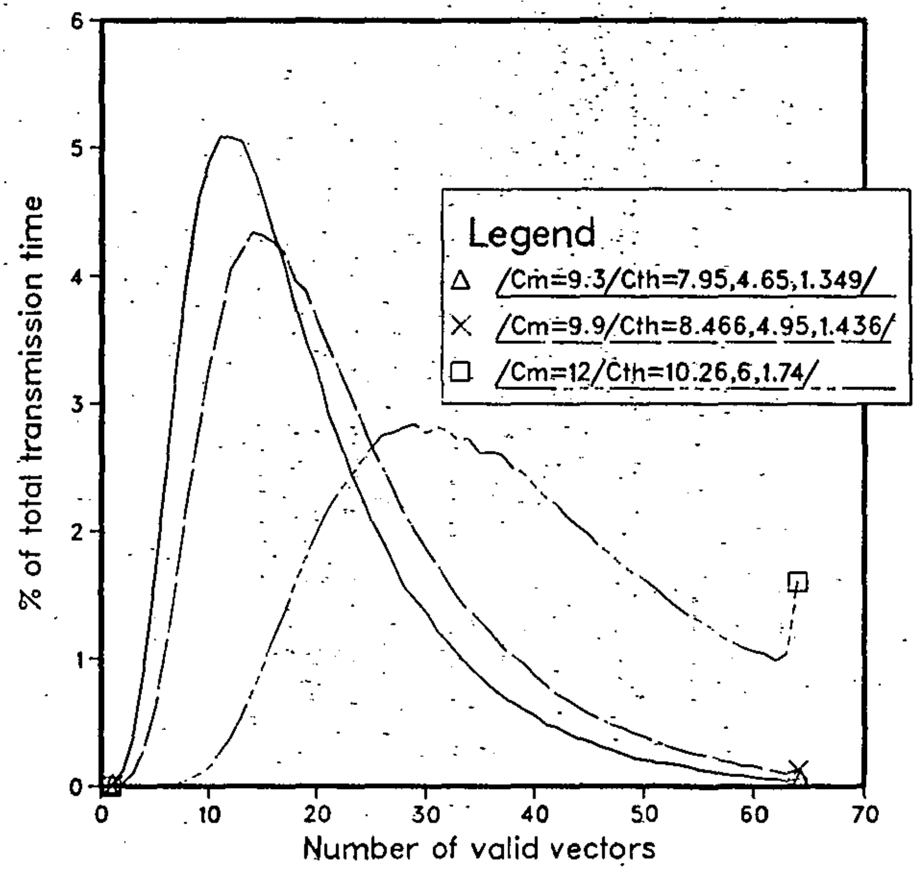
COMMON ATTRIBUTES
/M=8/C=3/Det=V64/N=64/

Graph 6.2.17 Type–A Distribution. Dynamic Expansion Limitation Method. Code 3 Eb/No=5dB



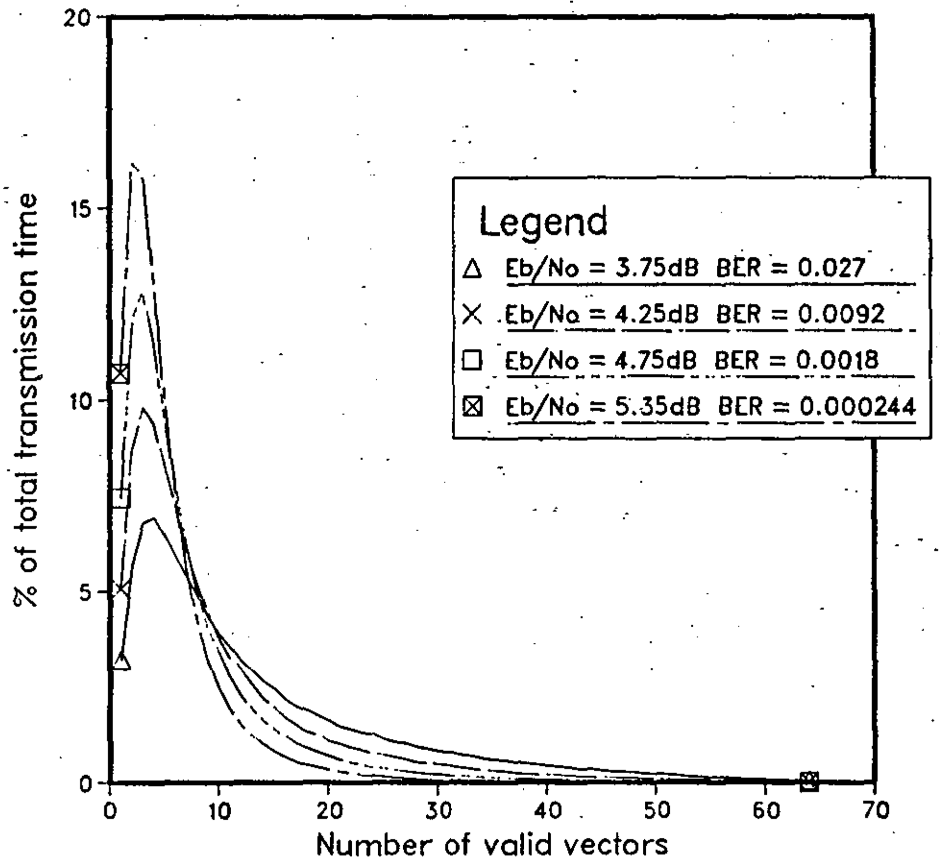
COMMON ATTRIBUTES
/M=8/C=3/Det=V64/N=64/

Graph 6.2.18 Type–A Distribution. Dynamic Expansion Limitation Method. Code 3 Eb/No=5.25dB



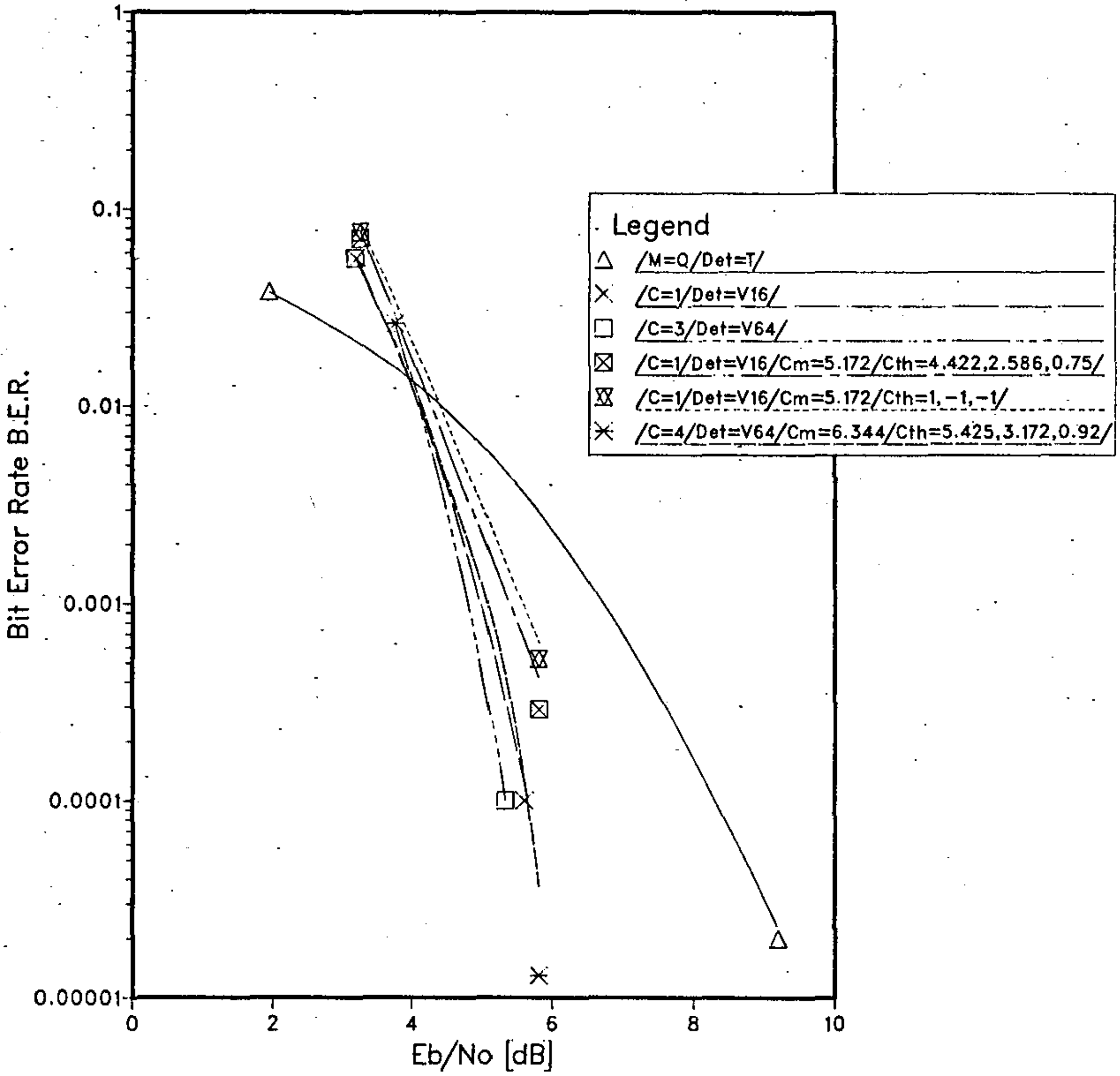
COMMON ATTRIBUTES
/M=8/C=3/Det=V64/N=64/

Graph 6.2.19 Type–A Distribution. Dynamic Expansion Limitation Method. Code 2



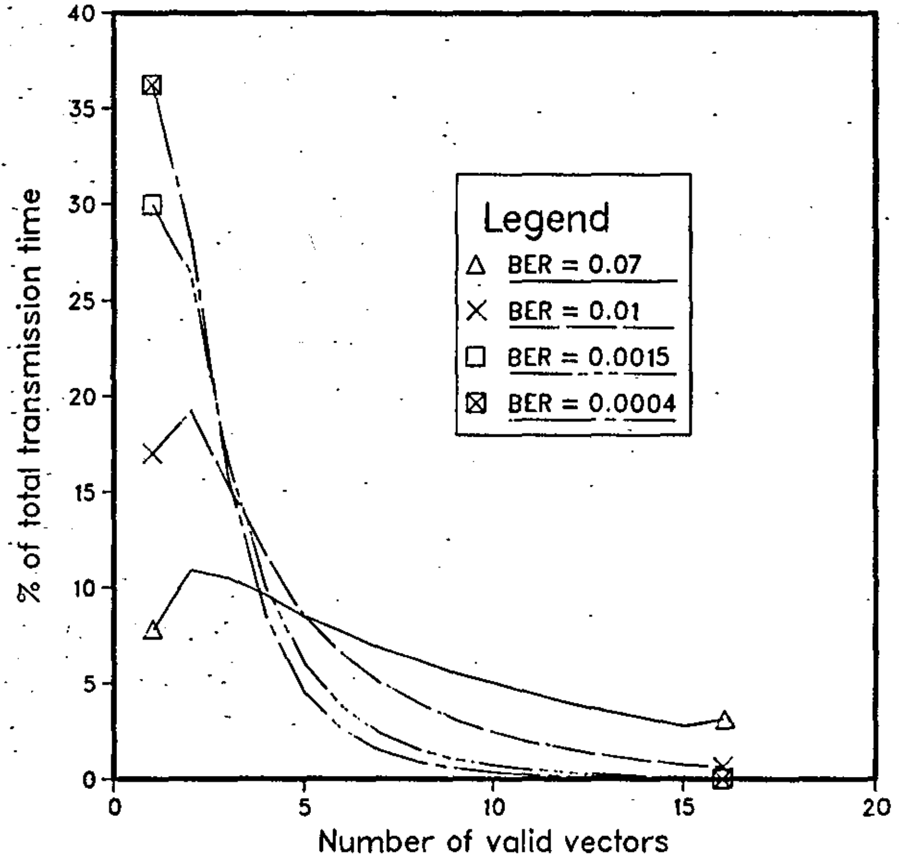
COMMON ATTRIBUTES
/M=8/C=2/Det=V64/N=64/
/Cth=3,0,0/Cm=6.344/

Graph 6.2.20 Performance of Chosen Schemes



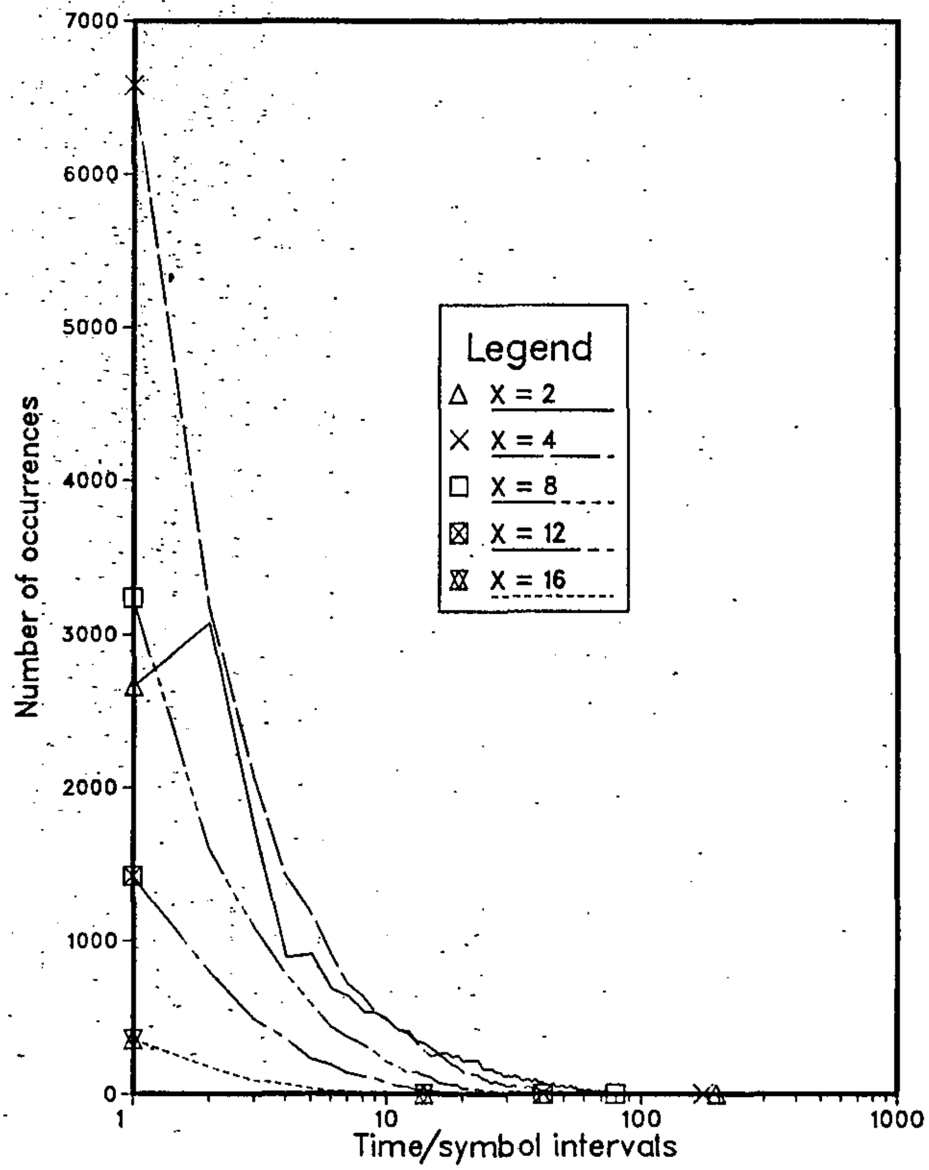
COMMON ATTRIBUTES
/M=8/N=64/

Graph 6.2.21 Type–A Distribution. Dynamic Expansion Limitation Method. Code 1



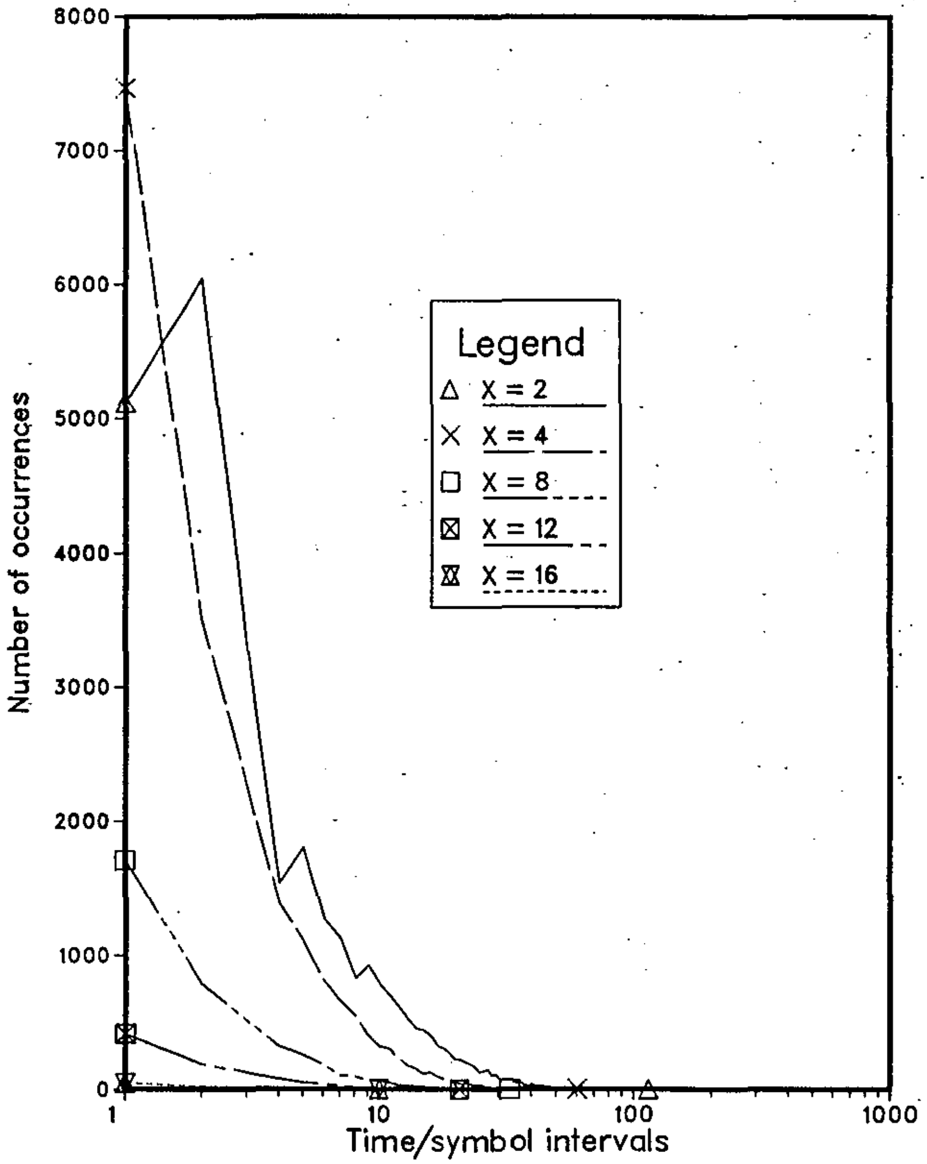
COMMON ATTRIBUTES
/M=8/C=1/Det=V16/N=64/
/Cth=4.422,2.586,0.75/Cm=5.172/

Graph 6.2.22 Type–B Distribution. Dynamic Expansion Limitation Method. Code 1 Eb/No=4.25dB BER=0.01



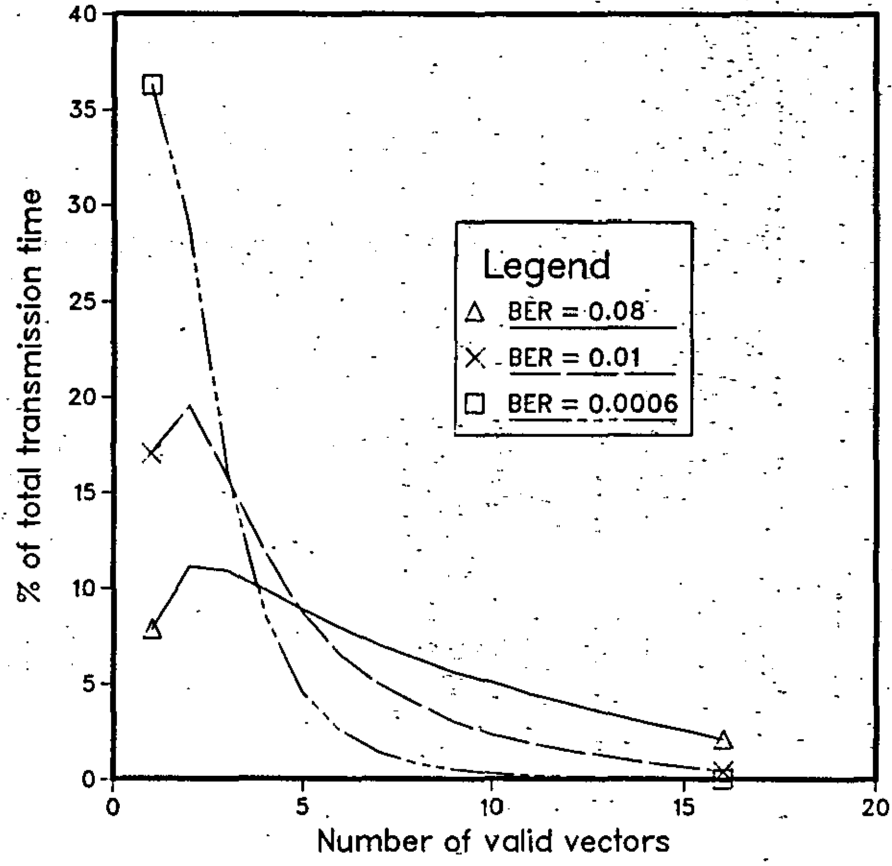
COMMON ATTRIBUTES
/M=8/C=1/Det=V16/N=64/
/Cth=4.422,2.586,0.75/Cm=5.172/

Graph 6.2.23 Type–B Distribution. Dynamic Expansion Limitation Method. Code 1 Eb/No=5.3dB BER=0.0015



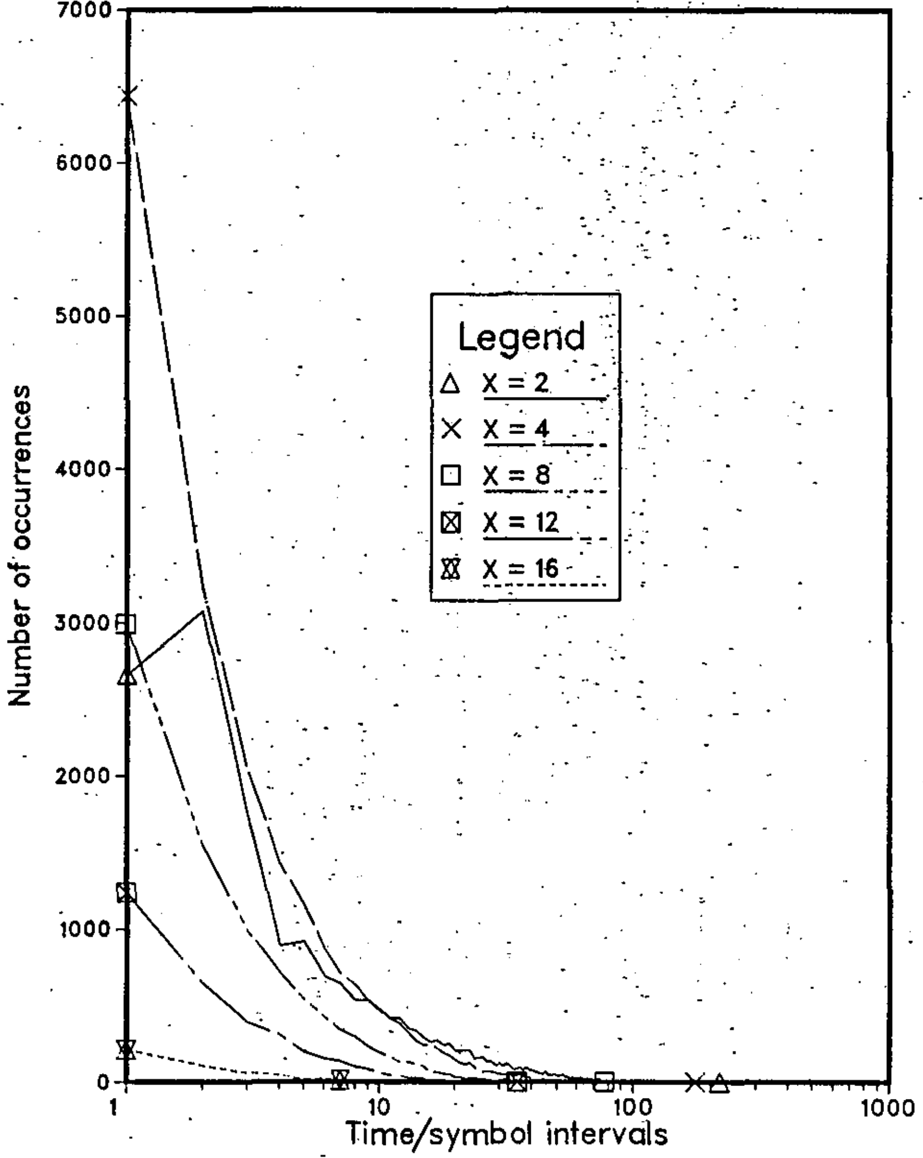
COMMON ATTRIBUTES
/M=8/C=1/Det=V16/N=64/
/Cth=4.422,2.586,0.75/Cm=5.172/

Graph 6.2.24 Type–A Distribution. Dynamic Expansion Limitation Method. Code 1



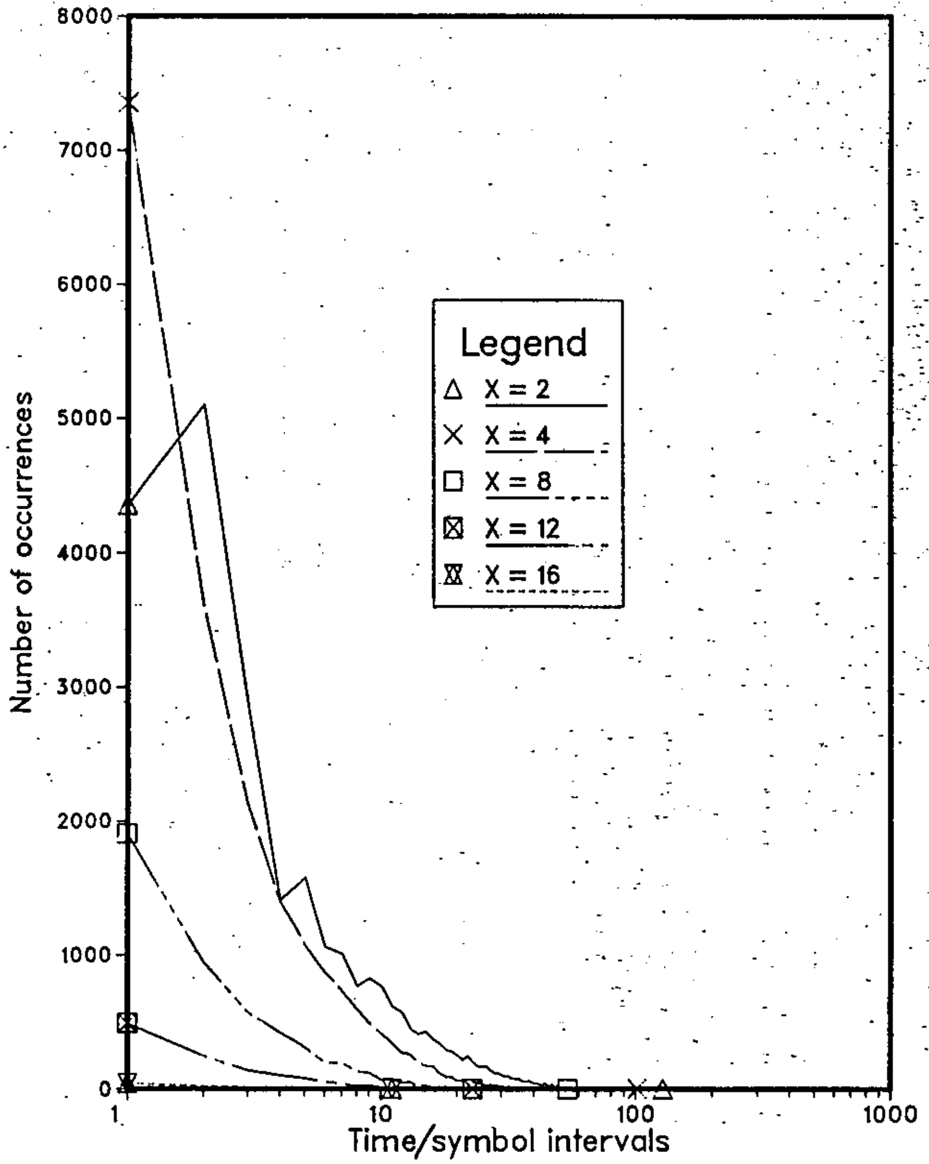
COMMON ATTRIBUTES
/M=8/C=1/Det=V16/N=64/
/Cth=1,–1,–1/Cm=5.172/

Graph 6.2.25 Type–B Distribution. Dynamic Expansion Limitation Method. Code 1 Eb/No=4.25dB BER=0.013



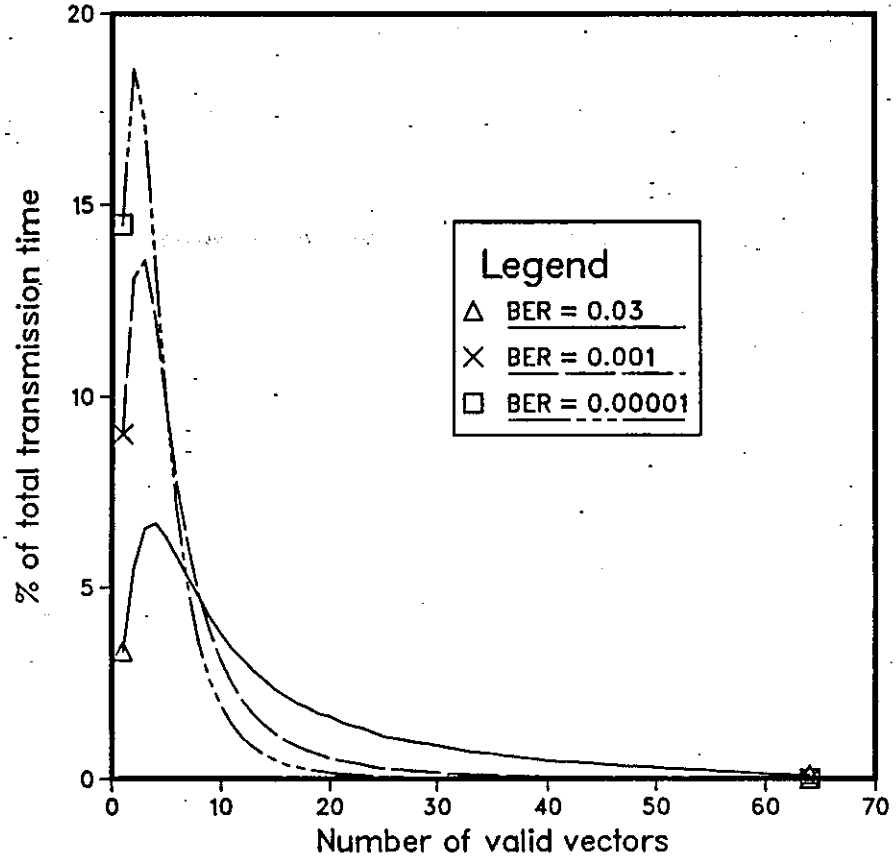
COMMON ATTRIBUTES
/M=8/C=1/Det=V16/N=64/
/Cth=1,-1,-1/Cm=5.172/

Graph 6.2.26 Type–B Distribution. Dynamic Expansion Limitation Method. Code 1 Eb/No=5dB BER=0.003



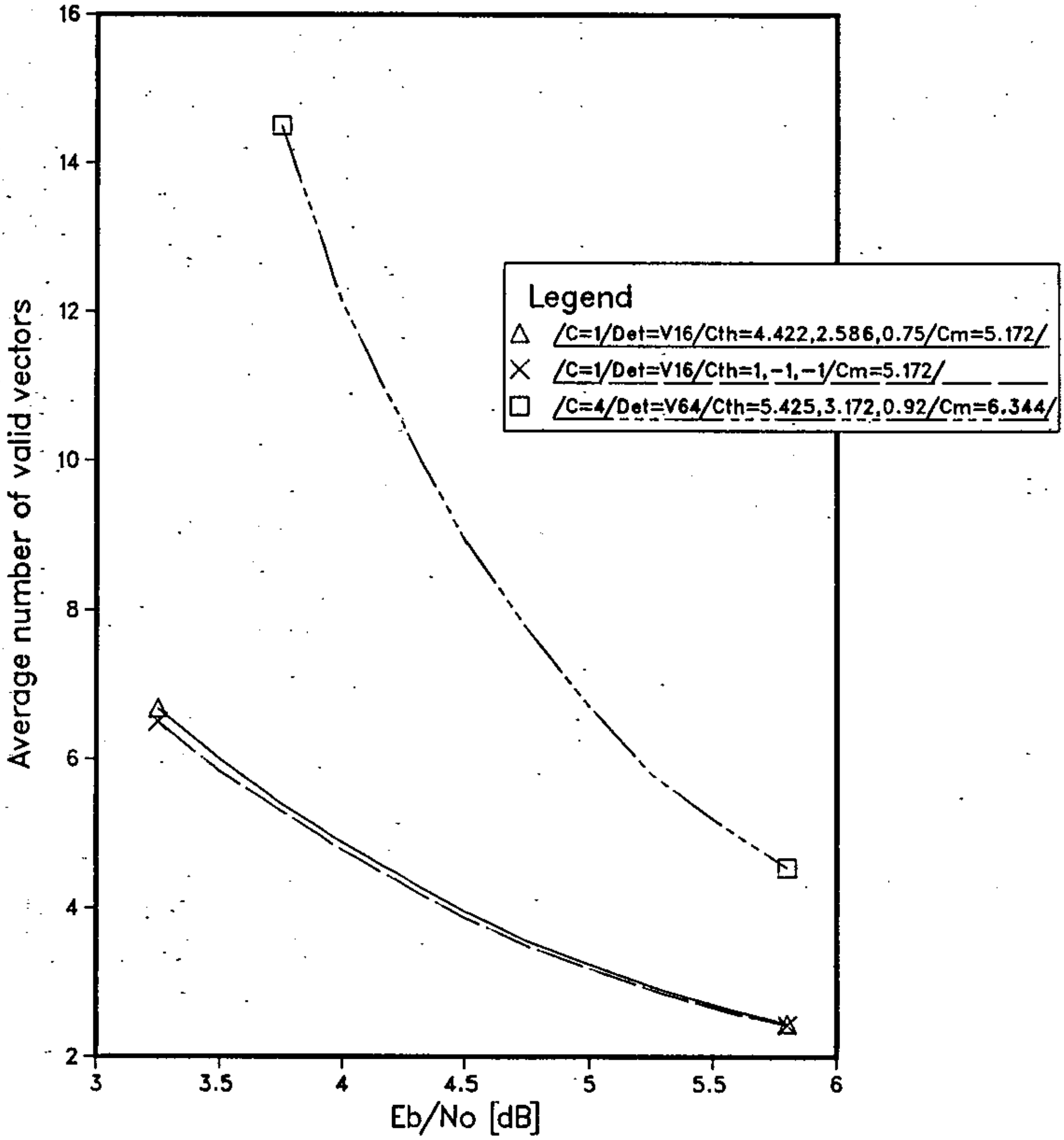
COMMON ATTRIBUTES
/M=8/C=1/Det=V16/N=64/
/Cth=1,−1,−1/Cm=5.172/

Graph 6.2.27 Type–A Distribution. Dynamic Expansion Limitation Method. Code 4



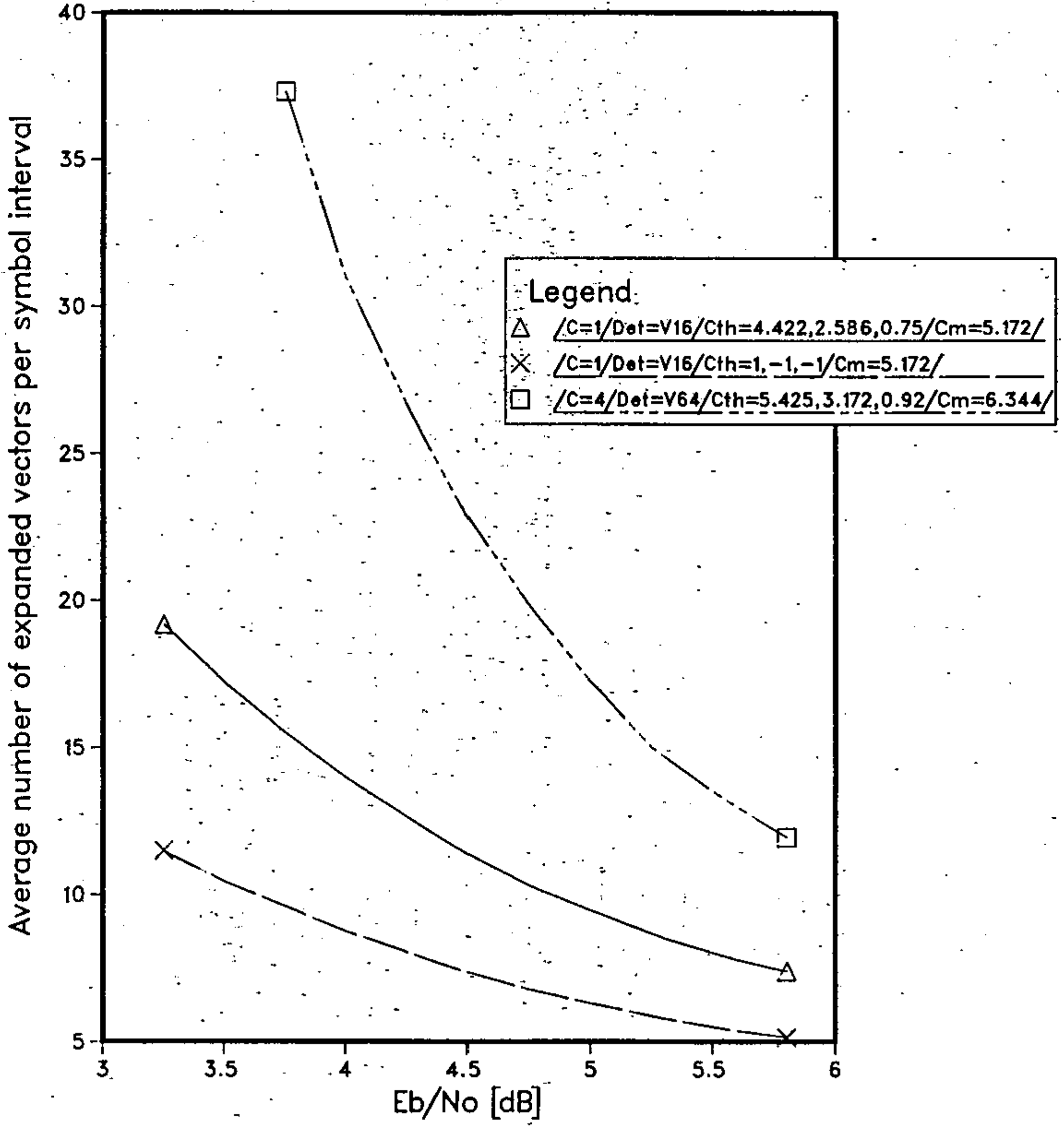
COMMON ATTRIBUTES
/M=8/C=4/Det=V64/N=64/
/Cth=5.425,3.172,0.92/Cm=6.344/

Graph 6.2.28 Variation in the average number of valid vectors with signal to noise ratio



COMMON ATTRIBUTES
/M=8/N=64/

Graph 6.2.29 Variation in the average number of valid expanded vectors per symbol interval with signal to noise ratio



COMMON ATTRIBUTES
/M=8/N=64/

Chapter 7

DISCUSSION OF RESULTS

This Chapter is split into three sections. In the first, a selection of the schemes outlined in Chapters 3 to 6 are compared, where the chosen schemes are those which are regarded as the most promising. The second section considers the originality of the study, and specifically notes those ideas which are not solely attributable to the author. The final section outlines further work which is required in order to be able to make a reasoned choice between the modulation methods and their attendant detectors.

7.1 COMPARISON OF THE MORE PROMISING SCHEMES

As noted in Chapter 3, a Viterbi Algorithm detector with four stored vectors is considered to be a viable scheme for the CORPSK(4-7,1+D) modulation method. For the coded 8PSK modulation method, Viterbi detection, near-maximum likelihood System 1 detection, and the noise-adaptive Viterbi-type detection scheme, are considered. Graphs 7.1.1 to 7.1.4 compare the schemes in terms of tolerance to noise. E_b is the average energy transmitted per data bit. $N_0/2$ is the two-sided power spectral density of the additive white Gaussian noise. (See Appendix A5 for more details of the simulation techniques. Appendix A8 describes the notation used to describe the schemes being compared.) In all the graphs, the accuracy of the results, in the region of bit error rate (BER) 1 in 10^3 to 1 in 10^4 is of the order of ±0.3dB.

From Graph 7.1.1 it is clear that Viterbi detection in the case of the coded 8PSK modulation method gains substantially in tolerance to noise, compared with the preferred scheme for CORPSK(4-7,1+D) modulation, below a BER of 1 in 10^2. At a BER of 1 in 10^4 this advantage is over 1.5dB. This advantage is achieved at the expense of a considerable increase in complexity for the detectors in the case of coded 8PSK modulation. (The Viterbi detector for the eight phase scheme using Code 1 is approximately 4 times more complex than the chosen Viterbi detector for the CORPSK(4-7,1+D) scheme, while for Code 3 the required detector is some 16 times more complex than for the CORPSK(4-7,1+D) scheme.) From Graph 3.2.7, the addition of a premodulation filter with a Nyquist III-ammended 0% Roll-Off Raised Cosine frequency modulating pulse leads to a degradation in tolerance to noise of the order of 0.5dB at a BER of 2 in 10^4, for CORPSK(4-7,1+D) modulation. Since such a filter[62] is required to restrict the otherwise rather wide bandwidth of the scheme in Graph 7.1.1, this degradation must be taken into account. The CORPSK(4-7,1+D) modulation method does provide a number of possibly important advantages. Since it provides a constant envelope signal in non-bandlimited form, the signal is not affected by nonlinear operations in high power amplifiers (HPA), both in the earth stations and on board the satellite. The effect of such nonlinear operations on non-constant envelope signals, such as bandlimited QPSK, is that the spectral sidelobes of the signal tend to spread, causing an unacceptable level of out of band radiation.[12,22] In a system comprised of many closely packed channels this inevitably leads to an unacceptable level of adjacent channel interference (ACI).[3] Therefore, for QPSK-type signals, these HPAs are often backed-off from saturation so that they operate in the linear portion of their characteristics. This may require a back-off (relative to the output level of the saturated amplifier) of up to 6dB. If the corresponding constant envelope scheme requires no such back-off, this translates into a gain of 6dB in tolerance to noise, since more power can be radiated in the case of CORPSK(4-7,1+D) modulation using the same HPA. Unfortunately, it is often found that the CORPSK(4-7,1+D) modulation scheme cannot be implemented in schemes with closely packed channels, without significant bandlimiting. The minimum channel spacing for the pure, non-bandlimited, CORPSK(4-7,1+D) signal is in the range 0.9 to 1.0 times the bit rate.[34] Simulation tests have tended to show that the CORPSK(4-7,1+D) signal's bandwidth increases substantially at the output of a HPA, if the input signal to the HPA is bandlimited.[6] This in fact, tends to be more severe than for offset QPSK-type schemes, under similar circumstances.[6] Also from Graph 7.1.1 the gain in tolerance to noise over QPSK at a BER of 1 in 10^3, (less than 1dB when the premodulation filter is taken into account), does not really justify the increased complexity due to the Viterbi detector. A second advantage of this signal is that, since CORPSK(4-7,1+D) modulation is a differential scheme, it is inherently immune to sudden phase changes. Differential coding for the coded 8PSK modulation scheme does not solve this problem since the codes are not transparent to sudden phase changes, but there are techniques which are available to overcome this.[19] Carrier frequency and phase synchronisation, and timing synchronisation, are additional problems which have not been fully solved for CORPSK(4-7,1+D)[34,62], or similar correlative phase modulation, (CPM), schemes.[49] Indeed, for coded 8PSK this may also be a major problem, simply because the signal to noise ratio at which the synchronisation circuitry must operate, is significantly lower than that for in-service 8PSK systems.[49] From Tables 3.2.2, 3.2.5 and 3.2.9, it can be concluded that the error burst characteristics for the eight phase schemes are not very greatly increased compared with those for the CORPSK(4-7,1+D) modulation method.

The definition of an error burst is given in Appendix A5. In the latter case the average number of bit errors per burst is about 3 at a BER of 1 in 10^4. In the former case this figure is about 10 for both schemes shown in Graph 7.1.1, at the same BER. In general, and given the above discussion, the results tend to point to coded 8PSK as the preferred modulation method when the detector uses the Viterbi Algorithm, as long as such detectors are feasible at the required data rates.

Graph 7.1.2 compares near-maximum likelihood System 1 detection, (Section 4.1), with noise-adaptive Viterbi-type detection, (Section 6.2), for schemes using Code 1. System 1 has been chosen as the representative near-maximum likelihood scheme for this comparison, since in Chapter 4 it provides the best results for the detectors tested. Graph 7.1.3 provides a similar comparison for the longer, constraint length k=4, codes, (Codes 3 and 4 in particular). From Graph 7.1.2, the noise-adaptive schemes have a better tolerance to noise than both System 1 schemes, at least down to a BER of 5 in 10^4. The shapes of the curves for the noise-adaptive schemes suggest that this may no longer be the case at lower BERs, but certainly in the range of BER 1 in 10^3 to 1 in 10^4, the noise-adaptive schemes have a performance which is similar to the 8-vector, (k_1=8), System 1 scheme. The reason for the difference in the slopes of the curves is most easily explained with reference to Graphs 6.2.28 and 6.2.29. These show that the equipment complexity required for a given variant of the noise-adaptive scheme, (that is, given the values of C_m, cth(1),cth(2) and cth(3)), reduces as the signal to noise ratio increases. If a method of adjusting these parameters could be found such that the complexity remains constant as the signal to noise ratio varies, one would expect the slopes of the curves to be very similar to that for Viterbi detection. In order to compare the schemes in terms of complexity, a BER of 6 in 10^{4} has been chosen as the point of comparison in Table 7.1.1. The complexity measures are difficult to compare directly, but it can be seen that the number of expanded vectors that require processing in the case of the System 1 schemes, are considerably greater than the number requiring processing in the two noise-adaptive cases. This ignores the fact that the number of costs which have to be compared, (that is ranked), in the selection of one vector in the case of System 1 detection, is considerably greater than the average number of costs to be ranked in the selection of one vector in the case of the noise-adaptive detector. Of course, the noise-adaptive detector requires input and output buffer stores and the attendant control circuitry, and the design of a system to fully exploit these potential reductions in complexity is by no means simple, (Section 6.2). Despite this, for Code 1, these noise-adaptive schemes potentially provide a very significant reduction in system complexity over System 1 detection, at no degradation in tolerance to noise, in the range of BER considered, (1 in 10^{3} to 1 in 10^{4}). In general the average number of bit errors per burst is lower for the noise-adaptive schemes, (see Tables 4.1.1, 6.2.10, and 6.2.11).

The same conclusion can be drawn from Graph 7.1.3 for the longer constraint length codes. The complexity comparison is given in Table 7.1.2. (Code 3 was chosen for the schemes using System 1 detection despite using Code 4 in the noise-adaptive scheme, because schemes for System 1 detection using Code 4 were shown to yield very poor results in Section 4.1. Therefore it was felt that this comparison is fairer.)

The results indicate a significant potential reduction in complexity, together with a considerable gain in tolerance to noise over the System 1 detectors. Notice that even the simplest System 1 detector, (k_1=4), is more complex than the noise-adaptive detector. The noise-adaptive scheme for Code 4 tends to have a larger number of bit errors per burst than System 1 detection for Code 3, (see Tables 4.1.3 and 6.2.12). Note though that System 1 detection with Code 4 produces very large error bursts, (Table 4.1.5).

Graph 7.1.4 compares noise-adaptive Viterbi-type detection with four-vector Viterbi detection for the ideal CORPSK(4-7,1+D) modulation method. (0.5dB should be added to the degradation in tolerance to noise for the latter scheme at a BER of 2 in 10^4 compared with noise-adaptive detection, due to the premodulation filtering.) Clearly, the noise-adaptive detectors for coded 8PSK are a significant improvement over Viterbi detection for the CPM scheme. Despite the fact that the complexity of the noise-adaptive schemes as defined in Section 6.2 falls off as the signal to noise ratio rises, the relative slopes of the curves show that the noise-adaptive schemes may give even greater gains in tolerance to noise at higher signal to noise ratios. This advantage in tolerance to noise is linked to a considerable potential reduction in complexity. The results are summarised in Table 7.1.3, where the complexity is relative to the preferred, 4-vector, Viterbi detector for CORPSK(4-7,1+D) modulation, and the measure used is the average number of expanded vectors per symbol interval at a BER of 6 in 10^4. It can be seen that in all cases, the potential gain in the trade-off between equipment complexity and tolerance to noise, is considerable. For example, at a BER of 6 in 10^4, a scheme using Code 1 could gain 0.4dB in tolerance to noise, with only about 1/3 of the complexity of the 4-vector Viterbi detector for CORPSK(4-7,1+D) modulation. The scheme using Code 4 could gain nearly 1dB in tolerance to noise at the same BER, for approximately the same complexity as the Viterbi detector for CORPSK(4-7,1+D) modulation. These gains increase significantly at lower values of the BER, (see Graphs 6.2.28, 6.2.29 and 7.1.4). These results do beg the question, would the noise-adaptive technique be applicable to CORPSK(4-7,1+D) modulation, and if so what reductions in equipment complexity are feasible? Note that these gains cannot be as significant as in the case of coded 8PSK modulation, simply because four-vector Viterbi detection is already reasonably simple. A much more interesting application would lie in the area of general CPM schemes,[49] where the Viterbi detector requires a prohibitive number of stored vectors, (as long as the attendant synchronisation problems can be solved[49]). This would also include multi-h modulation, where the modulation index is varied in some cyclic and systematic way over consecutive symbol intervals, (again, given a solution to the synchronisation problems[49]).

The noise-adaptive Viterbi-type detector could in fact be a candidate for any coherent detection scheme which at present requires Viterbi or near-maximum likelihood detection, (for example for telephone or HF radio channels).

SCHEME	E_b/N_0-Value at which BER $= 6 \times 10^{-4}$ (dB)	Measure of Relative Complexity (Per Symbol Interval)
/Det=1N4/N=32/	7.1	16 expanded vectors/symbol interval 4 cost rankings, each involving 16 costs
/Det=1N8/N=32/	5.9	32 expanded vectors/symbol interval 8 cost rankings, each involving 32 costs
/Det=V16/N=64/C_m=5.172/ cth=4.422,2.586,0.75/	5.6	On average, approximately 7.75 expanded vectors/symbol interval 2.5 cost rankings
/Det=V16/N=64/C_m=5.172/ cth=1,-1,-1/	5.8	On average, approximately 5.75 expanded vectors/symbol interval 2.5 cost rankings

TABLE 7.1.1: Complexity Comparisons for Near-Maximum Likelihood System 1 and Noise-Adaptive Viterbi-Type Detection of Coded 8PSK, for Code 1

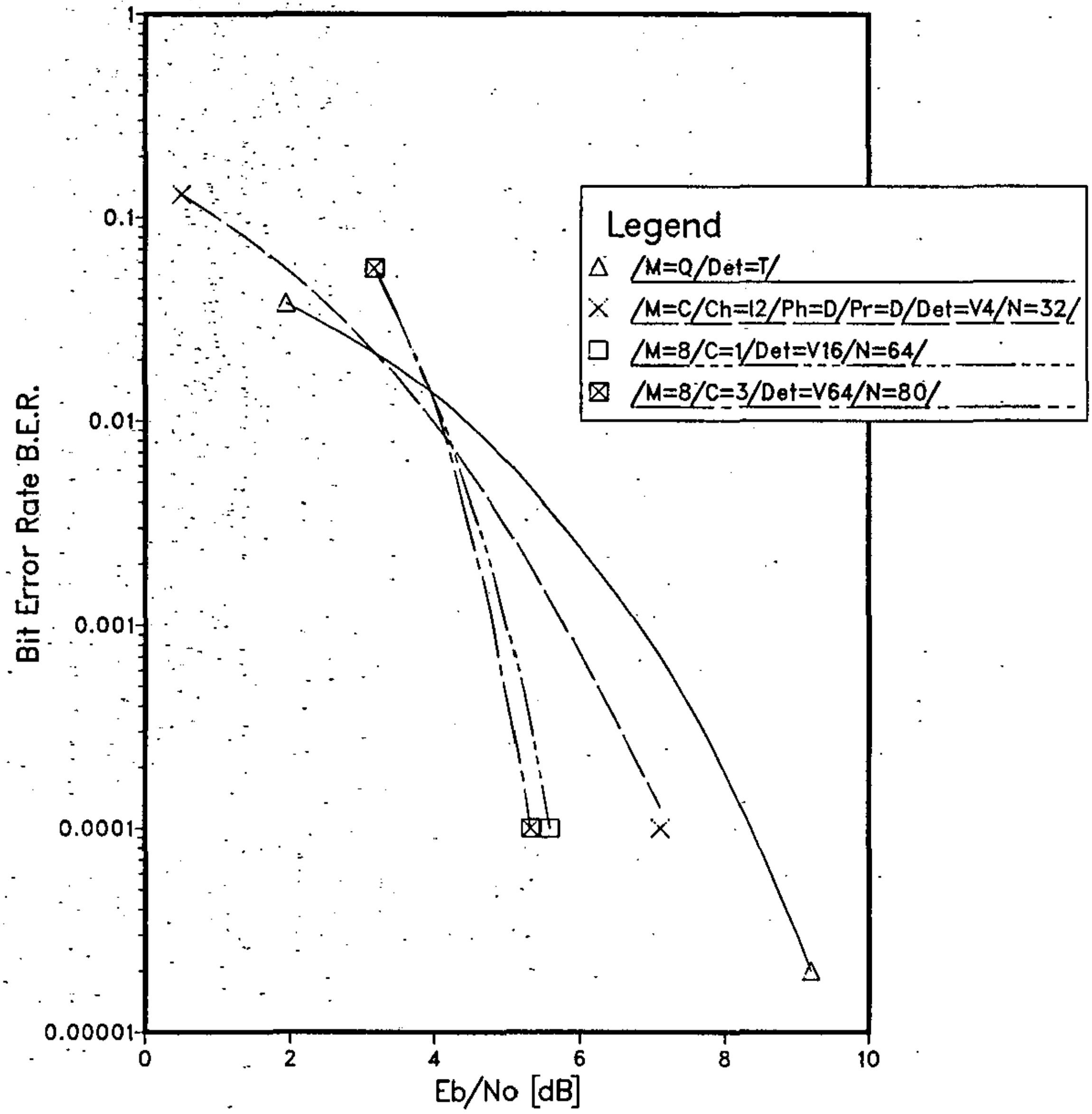
SCHEME	E_b/N_0-Value at which BER $= 1 \times 10^{-4}$ (dB)	Measure of Relative Complexity (Per Symbol Interval)
/Det=1N4/N=32/	7.0	16 expanded vectors/symbol interval 4 cost rankings, each involving 16 costs
/Det=1N8/N=32/	6.2	32 expanded vectors/symbol interval 8 cost rankings, each involving 32 costs
/Det=1N16/N=64/	6.05	64 expanded vectors/symbol interval 16 cost rankings, each involving 64 costs
/Det=V16/N=64/C_m=6.344/ cth=5.425,3.172,0.92/	5.65	On average, approximately 12.5 expanded vectors/symbol interval 4.5 cost rankings

TABLE 7.1.2: Complexity Comparisons for Near-Maximum Likelihood System 1 and Noise-Adaptive Viterbi-Type Detection of Coded 8PSK, for Code 3

SCHEME	E_b/N_o-Value at which BER $= 6 \times 10^{-4}$ (dB)	Relative Complexity in Terms of The Average Number of Expanded Vectors/ Symbol Interval
/M=C/Ch=I2/Ph=D/Pr=D/Det=V4/N=32/	6.2	1
/M=8/C=1/Det=V16/N=64/C_m=5.172/ cth=4.422,2.586,0.75/	5.6	0.48
/M=8/C=1/Det=V16/N=64/C_m=5.172/ cth=1,-1,-1/	5.8	0.36
/M=8/C=4/Det=V64/N=64/C_m=6.344/ cth=5.425,3.172,0.92/	5.3	0.9

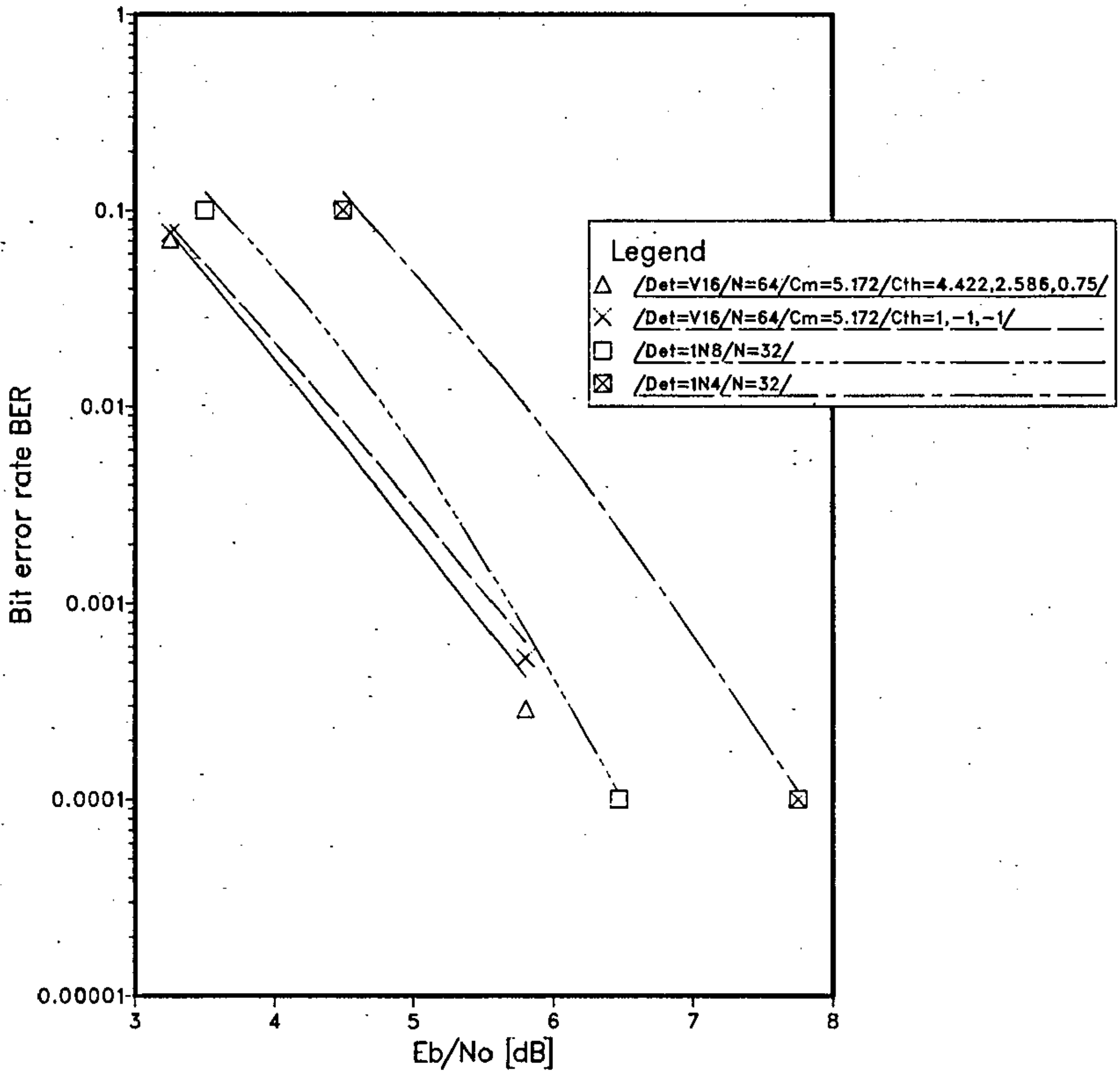
TABLE 7.1.3: Complexity Comparisons for Viterbi Detection of CORPSK(4-7,1+D) and Noise-Adaptive Viterbi-Type Detection for Coded 8PSK

Graph 7.1.1 Comparison of CORPSK[4–7,1+D] Differential Phase Perfect Channel Scheme with Viterbi Detection for Coded 8PSK



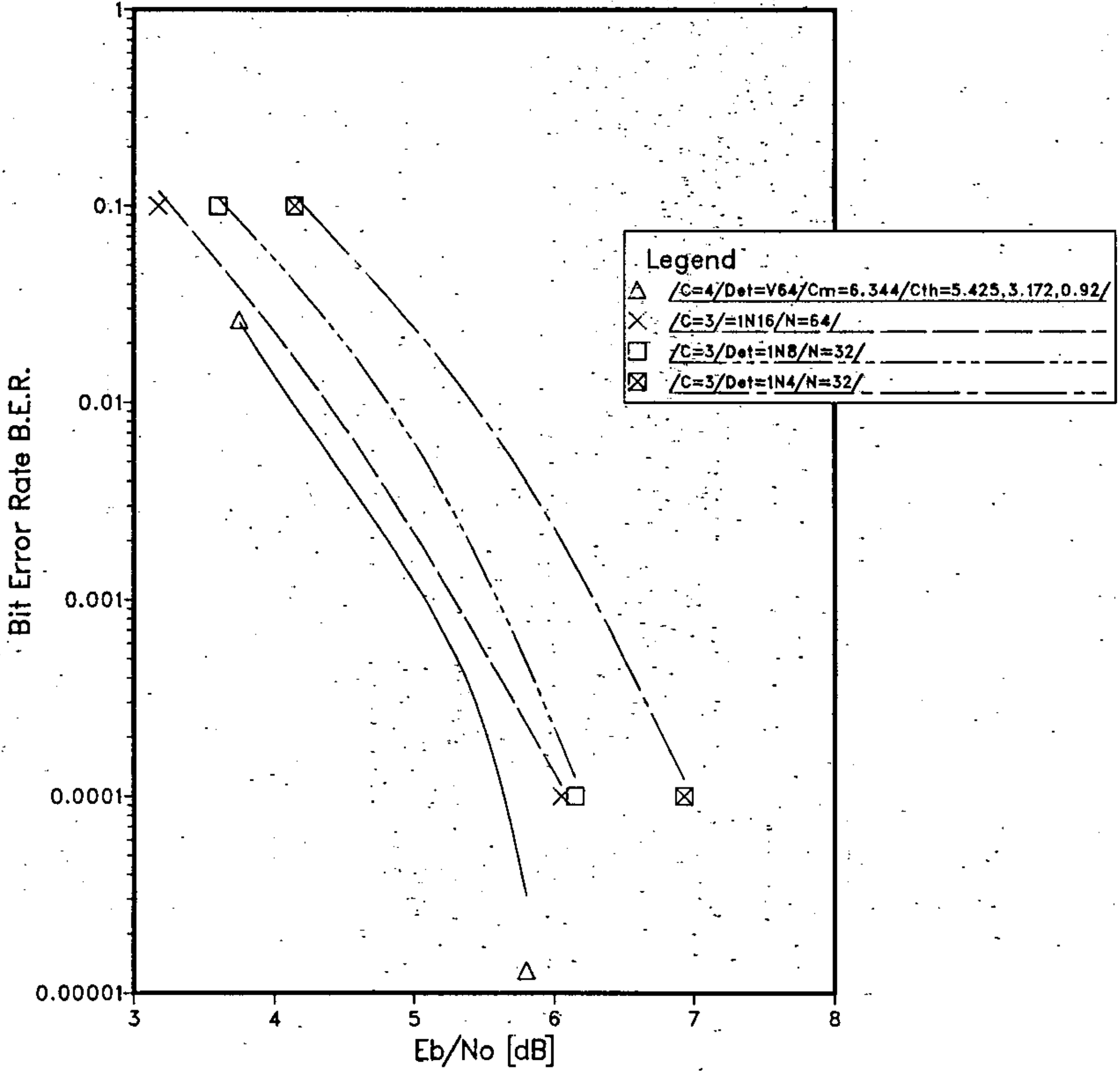
COMMON ATTRIBUTES
None

Graph 7.1.2 Code 1. Comparison of Near Maximum Likelihood System 1 Detection with Noise-Adaptive Detection



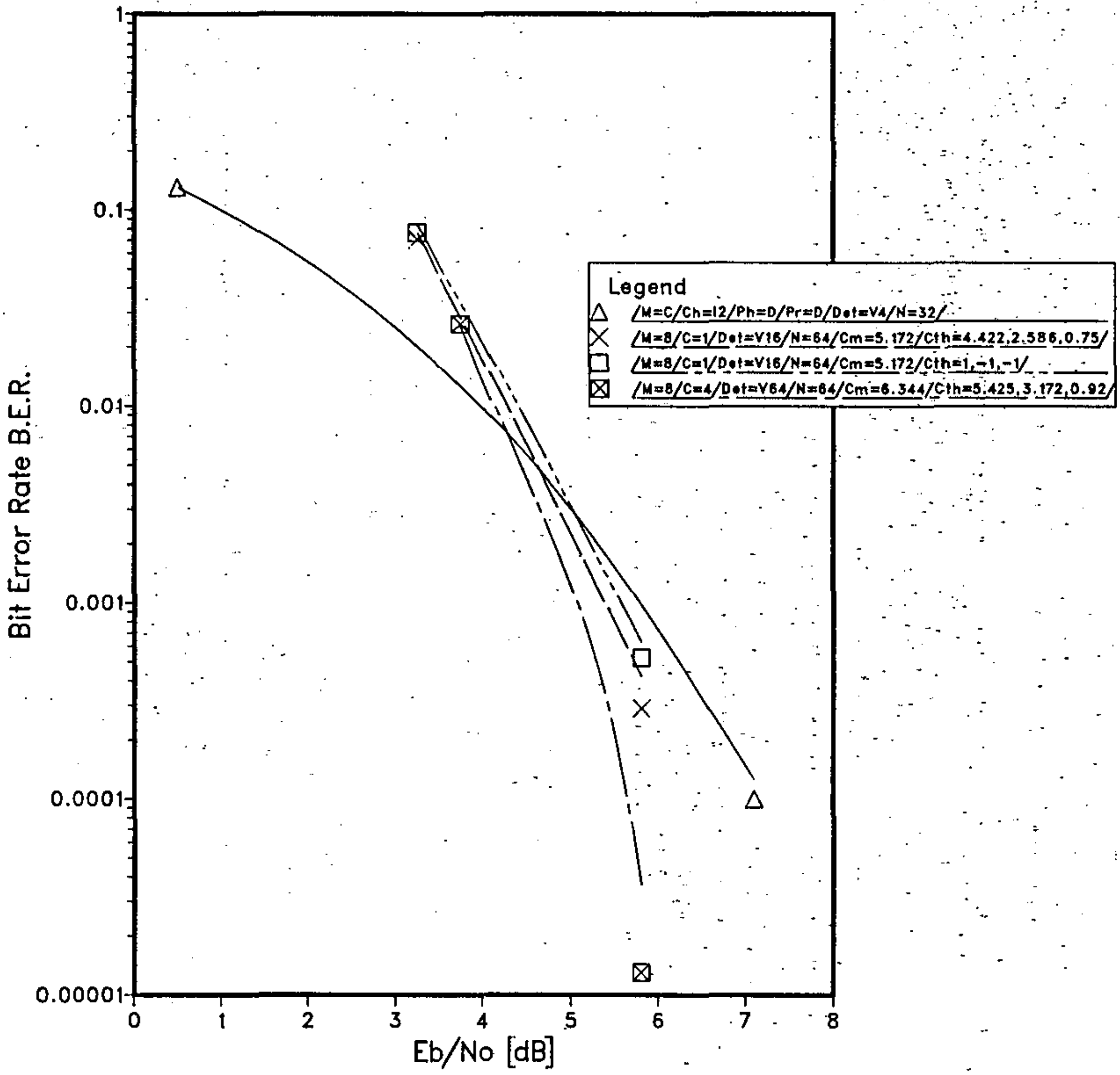
COMMON ATTRIBUTES
/M=8/C=1/

Graph 7.1.3 Comparison of Near Maximum Likelihood System 1 Detection [Code 3] with Noise–Adaptive Detection [Code 4]



COMMON ATTRIBUTES
/M=8/

Graph 7.1.4 Comparison of CORPSK[4–7,1+D] Differential Phase
Perfect Channel Scheme with Noise–Adaptive Detection
for Coded 8PSK



COMMON ATTRIBUTES
None

7.2 NOTE ON ORIGINALITY

As far as the author is aware, all of the work described within this thesis is attributable to the author, except either where credit is specifically given in the text, (for example in the form of references), or in certain cases, (noted below), where an idea originated from the author's supervisor. In particular, all the computer simulation tests described within this thesis were undertaken by the author, and all the computer programs were written by the author. The following lists the most important ideas which originated either from the author, or from the author's supervisor, during the course of the research.

(1) The use of phase as a measure of distance originated from the author's supervisor.

(2) The reduced complexity, pseudobinary System 1 detector, originated from the author, (Section 4.3).

(3) The look-forward detection scheme originated from the author's supervisor, and was further developed by the author with the addition of pseudobinary versions, (Section 4.4).

(4) The vector retention-forcing algorithm originated from Reference 75. The idea of extending the retention period to greater than one symbol interval is attributable to the author's supervisor (Section 4.5).

(5) The analysis of near-maximum likelihood detection techniques based on sequence numbers,[76] as applied to CORPSK(4-7,1+D) and 8PSK modulation, is the author's work.

(6) The application of soft decision techniques to table look-up syndrome decoding for coded 8PSK, and the algorithms by which the look-up tables were formulated, originate with the author.

(7) The use of a maximum cost for the chosen vectors in the noise-adaptive Viterbi-type detector is the author's idea, although similar ideas have been discussed before, in References 76 and 88. The static expansion limitation method is the author's idea. The idea of using cost thresholds in the dynamic expansion limitation method, originates from the author's supervisor. The design of the implementation for a noise-adaptive detector, for a scheme using Code 1, is the author's, (Section 6.2).

7.3 RECOMMENDATIONS FOR FURTHER STUDY

The aim of this section is to outline further areas of work which are required to prove the worth of the noise-adaptive Viterbi-type detector under non-ideal conditions, and to investigate the possible application of the detector to other modulation methods presently being considered for satellite systems. The following is a list of the possible avenues for further study.

(1) The effects of non-ideal channel characteristics should be considered, leading to measures of the performance of the noise-adaptive algorithm relative to Viterbi detection over representative satellite channels. The investigation would include the effects of adjacent- and co-channel interference[3], and the effects of nonlinear amplification in the form of high power amplifiers, (HPA), with various back-offs from saturation. The effect of baseband pulse-shaping of the complex modulating waveform before application to the 8PSK modulator is required. A study of the effect of bandlimiting the 8PSK signal before application to the HPA is required. This work would also include implementation-oriented investigations, such as the effects of quantisation and the approximations which have to be made in digital filter implementations. In addition, the effects of carrier and timing synchronisation errors require investigation. This work would also include the specification of complete satellite channels, either incorporating or not incorporating regeneration of the data on board the satellite.

(2) Further statistical results are required to enable firm decisions to be made on the buffer store requirements for the noise-adaptive

detector, for a number of variants of the detector, and various signal to noise ratios. This study would also increase knowledge of the effects of varying the detector parameters. New codes could be tested, and in particular, codes with specific column distance functions or distance profiles, (see Section 6.1), in order to ascertain the optimum codes.

(3) There is the possibility of varying the detector parameters within the detector, (C_m,cth(1),cth(2), and cth(3)), in order to either increase the slope of the curves in Graph 7.2.2, or to optimise the detector in non-Gaussian noise environments. This investigation could also include variants of the rules governing the number of valid expanded vectors, (Section 6.2).

(4) A more detailed study of possible practical implementations is required for various applications.[93] This would include the feasibility of high bit-rate schemes using more parallel processing, and lower bit-rate schemes where digital signal processors (DSPs)[91,92] are an attractive solution in terms of cost and flexibility, (including the option of reprogramming). The possibility of implementing the required functions using a very small number of Very Large Scale Integration, (VLSI), integrated circuits requires study. Detector structures for determining the valid expanded vectors, for vector and cost storage, and for the cost-ranking functions, require investigation.

(5) The application of noise-adaptive Viterbi-type detectors to different modulation schemes could provide attractive alternatives to coded 8PSK. These include correlative phase modulation, (for example CORPSK(4-7,1+D)), and multi-h schemes.

(6) An investigation into the feasibility of carrier and timing synchronisation is required, since the synchronisation circuitry is required to operate at much lower signal to noise ratios than in current systems (such as QPSK and 8PSK).

CHAPTER 8

CONCLUSIONS

The investigation has noted that Viterbi Algorithm detection is generally too complex for coded 8PSK modulation, (although a scheme using Code 1 with a soft-decision 16-vector Viterbi detector is considered technically feasible at data rates of up to 8Mbits/second[6,8,9]). On the other hand, Viterbi Algorithm detection is considered to be very attractive for CORPSK(4-7,1+D) modulation, with the specified rather wide equipment filtering, using 4 stored vectors.

Traditional near-maximum likelihood detection schemes, including pseudobinary variants,[64-71] are not really suited to convolutionally coded schemes, in that they provide only small reductions in detector equipment complexity, coupled to relatively high degradations in tolerance to noise compared with Viterbi Algorithm detection.

State redefinition techniques and soft-decision syndrome decoding for coded 8PSK are not viable techniques, since they yield considerable losses in tolerance to noise compared with optimal, (Viterbi Algorithm), detection.

Sequential decoding is not considered to be a viable technique for coded 8PSK, because the signal characteristics would probably lead to a large number of rather lengthy back-up searches.

Noise-adaptive Viterbi-type detection is a very promising technique in that it yields relatively small degradations in tolerance to noise compared with Viterbi Algorithm detection, coupled with considerable potential reductions in detector equipment complexity. The key to the complexity reduction is that it is a basically feedforward technique, with no time-consuming back-up searches, and with a well-defined maximum processing load per symbol interval, which is (potentially) only marginally greater than the corresponding processing load per symbol interval for Viterbi Algorithm detection. The feasibility of the technique is subject to the detailed development of practical implementations of the algorithm which approach, as closely as possible, the potential reductions in detector equipment complexity.

APPENDICES

A1 PRECODING

Precoding is a technique by which the effects of coding in correlative-level encoded[1,53] systems may be effectively removed whilst retaining the coding gains that such schemes offer. In this way excessive error propagation is avoided. Error propagation is defined as the effect by which a number of consecutive or near-consecutive detection errors are made, due to one isolated noise-induced error. Such effects occur because one code symbol is a function of more than one data symbol. This means that an error in the value of one code symbol will affect more than one detected data symbol. This analysis deals in particular with the case of correlative coding as implemented in the CORPSK(4-7,1+D) modulation scheme.[62]

The correlative-level encoder is defined by its generator function, which is a polynomial in the delay operator D,

$$G(D) = 1+D \qquad (A1.1)$$

A diagram of the system is given in Figure A1.1.

If the input data $\{s_i\}$ are statistically independent and equally likely to have any of their four values; 0,1,2 or 3 it can be shown that the four-level symbols $\{q_i\}$ are also statistically independent and equally likely to have any of their four different values.[51] In Figure A1.1 the sequences at the input and output of the coding blocks are expressed in terms of polynomials in the delay operator D, as per equations A1.2 to A1.4

$$S(D) = s_1 + s_2D + \ldots + s_iD^{i-1} + \ldots \qquad (A1.2)$$

$$Q(D) = q_1 + q_2D + \ldots + q_iD^{i-1} + \ldots \qquad (A1.3)$$

$$C(D) = c_1 + c_2 D + \ldots + c_i D^{i-1} + \ldots \qquad (A1.4)$$

The symbols $\{c_i\}$ have the possible values 0,1,2,...,6.

The precoding equation is

$$Q(D) = [S(D)/G(D)]\text{MODULO-4} \qquad (A1.5)$$

The MODULO-4 rule is given in Equation (A1.6).

$$\begin{aligned} q_i < 0 \;&;\quad q_i = q_i + 4 \\ 0 \leqslant q_i \leqslant 3 \;&;\quad q_i = q_i \\ q_i > 3 \;&;\quad q_i = q_i - 4 \end{aligned} \qquad (A1.6)$$

The correlative-level encoder codes the data symbols $\{q_i\}$ according to Equation A1.7.

$$C(D) = Q(D)G(D) \qquad (A1.7)$$

From equations A1.5 and A1.7 it follows that

$$C(D) = [S(D)/G(D)]\text{MODULO-4}.G(D) \qquad (A1.8)$$

Therefore,

$$[C(D)]\text{MODULO-4} = [[S(D)/G(D)]\text{MODULO-4}.G(D)]\text{MODULO-4} \qquad (A1.9)$$

$$= S(D) \qquad (A1.10)$$

Clearly S(D) is immediately recoverable from C(D) by interpreting C(D) MODULO-4 as in Equation A1.11.

$$s_j = [c_j]\text{MODULO-4} , \quad j=1,2,\ldots \qquad (A1.11)$$

Clearly the operation described by Equation A1.11 recovers s_j from c_j alone. This means that the size of the error bursts can be reduced, since an error in one symbol c_j affects only one data symbol s_j[94].

The receiver is given in Figure 2.1.1. The detector operates on the received samples $\{r_i\}$ and outputs the detected data symbols $\{q_i'\}$ (the polynomial $Q'(D)$). Equation A1.11 is not explicitly implemented. Instead the inverse of the precoder at the transmitter is used to convert the $\{q_i'\}$ into the detected and decoded data symbols $\{s_i'\}$.

$$S'(D) = [Q'(D)G(D)]\text{MODULO-4} \qquad \text{(A1.12)}$$

($Q'(D)G(D)$ is the noisy code sequence which is possibly incorrect in some of its element values).

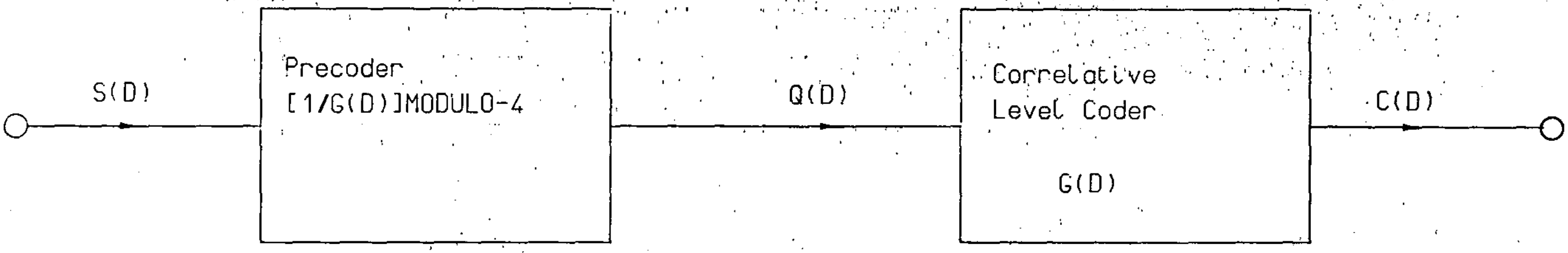


Figure A1.1 Generalised Precoding for Correlative-Level Coded Schemes

A2 A MATHEMATICAL TREATMENT FOR DIFFERENTIAL-PHASE CORPSK(4-7,1+D)

The transmitter section of Figure 2.1.1 is expanded in Figure A2.1. The precoder has been described in Appendix A1, so that in this treatment, the analysis will begin with the precoded data symbols $\{q_i\}$. The precoder is an option, and the following is unchanged for the non-precoded case, where q_i is replaced by s_i.

The correlative-level coder has at its input a level shifter which produces a polar signal at its output. The operation of the level shifter is outlined in Table A2.1. Clearly the output of the level shifter, which for convenience will continue to be represented as q_i, has one of the values $-1\frac{1}{2}$, $-\frac{1}{2}$, $+\frac{1}{2}$ or $+1\frac{1}{2}$. The symbols at the output of the level shifter are statistically independent and equally likely to have any of their four different values. Clearly the signal itself has not changed, only its representation has been ammended.

The correlative level coder is described as a power series in the delay operator D,

$$G(D) = 1+D \tag{A2.1}$$

The code symbol c_i is then given by

$$c_i = q_i + q_{i-1} \tag{A2.2}$$

c_i has the possible values -3, -2, -1 0, +1, +2, and +3. (This definition of c_i is different from that of Section 2.3. The $\{c_i\}$ in Section 2.3 are level-shifted to produce the polar versions, defined here. As for the $\{q_i\}$, the symbols themselves have not changed, only their representation has been ammended.)

The premodulation filter is defined by its frequency modulating pulse, $\alpha(t)$.[49,76] $\alpha(t)$ is normalised to have area 1/2. $\alpha(t)$ is proportional to the instantaneous rate of change of the phase of the signal at the filter's output, for a unit-valued impulse at its input.

$$\alpha(t) = k.\frac{d\phi(t)}{dt} \qquad \text{(A2.3)}$$

(k is a constant .)

$\phi(t)$ is the variation of signal phase with time, (the phase trajectory), caused by a unit impulse $\delta(t-iT)$. It is assumed that $\alpha(t)$ is of finite duration.

The phase response function for the filter is obtained from $\alpha(t)$ using Equation A2.4. (The phase response function gives $\phi(t)$ for a unit-valued impulse at the input to the filter.)

$$\beta(t) = \int_{-\infty}^{t} \alpha(\tau)d\tau \quad -\infty<t<\infty \qquad \text{(A2.4)}$$

Given the phase response function, the phase of the filter's output signal, $\phi(t)$, is given by Equation A2.5.

$$\phi(t) = 2\pi h \sum_i c_i \beta(t-iT) \quad -\infty<t<\infty \qquad \text{(A2.5)}$$

h is the modulation index which is assumed to be constant in the case of CORPSK(4-7,1+D). T is the symbol interval in seconds. For CORPSK(4-7,1+D) the maximum phase change over any symbol interval is arranged to be $3\pi/2$, and the modulation index h is set to 1/2.

The sampled form of the phase function at the time instants $\{iT\}$ is given by Equation A2.6.

$$\phi_i = 2\pi h \sum_{j=1}^{\infty} c_j \beta_{i-j} \qquad \text{(A2.6)}$$

where $\beta_j = \beta(jT)$. To this point the analysis of the premodulation filtering could apply to any correlative Phase Modulation, (CPM), scheme. The subset of CPM schemes denoted by the acronym CORPSK requires that the premodulation filter should meet Nyquist's Third Criterion, (Nyquist III), when followed by a frequency modulator. This ensures that the phase at the symbol sampling point, ϕ_i, is one of the fixed phases $jh\pi$; j=0,1,2, or 3. This facilitates carrier regeneration at the receiver.[34,62] Nyquist's Third Criterion states[62]

$$\int_{\frac{(2j-1)T}{2}}^{\frac{(2j+1)T}{2}} \alpha(t)dt = \begin{cases} 1 \text{ for } j=0 \\ 0 \text{ for } j\neq 0 \end{cases} \qquad \text{(A2.7)}$$

where j is an integer.

The Nyquist III property is such that the area under the frequency modulating pulse $\alpha(t)$, for each symbol interval for which it exists, is a fraction 1/a of the total area under the pulse, where a is an integer.[34] This ensures that the phase of the signal reaches exact sub-multiples of 2π radians at the end of each symbol interval.[34]

In reference (62) it is shown that the premodulation filter must have the following transfer function in order to satisfy Nyquist's Third Criterion.

$$\lambda_{III}(f) = \lambda_I(f) \; \frac{\pi fT}{\sin\pi fT} \qquad \text{(A2.8)}$$

$\lambda_I(f)$ is the transfer function of the filter whose frequency modulating pulse, $\alpha_1(t)$, obeys the First Nyquist Criterion, given in Equation A2.9.

$$\alpha_1(jT) = \begin{cases} 1 \text{ for } j=0 \\ 0 \text{ for } j\neq 0 \end{cases} \qquad \text{(A2.9)}$$

j is an integer. Clearly the Nyquist III-ammendment of $\lambda_I(f)$ emphasises the filter's transfer function around f=±1/2T Hz, (see Figure 2.4.4).

It is convenient to combine the correlative-level coding and premodulation filtering as shown below, in the composite premodulation filter with phase response $\beta'(t)$.

Let $$\beta'(t) = \beta(t) + \beta(t-T) \ , \quad -\infty<t<\infty \qquad \text{(A2.10)}$$

Then $$\phi(t) = 2\pi h \sum_i q_i \beta'(t-iT) \ , \quad -\infty<t<\infty \qquad \text{(A2.11)}$$

and $$\phi_i = 2\pi h \sum_{j=1}^{\infty} q_j \beta'_{i-j} \qquad \text{(A2.12)}$$

The continuous-phase waveform at the output of the modulator is given by Equation A2.13[49]:

$$x'(t) = \sqrt{\frac{4E_b}{T}} \cos(2\pi f_o t + \phi(t) + \phi_o) \qquad \text{(A2.13)}$$

E_b is the energy per bit transmitted, f_o is the carrier frequency, and ϕ_o is an arbitrary constant phase shift which can be neglected in the case of coherent detection[49]. The signal x'(t) may or may not be bandlimited as shown in Figure A2.1 to produce the transmitted signal x(t).

SYMBOL VALUE AT THE INPUT TO THE LEVEL SHIFTER	SYMBOL VALUE AT THE OUTPUT FROM THE LEVEL SHIFTER
0	-1½
1	-½
2	+½
3	+1½

TABLE A2.1: Mapping Function Performed by the Level Shifter

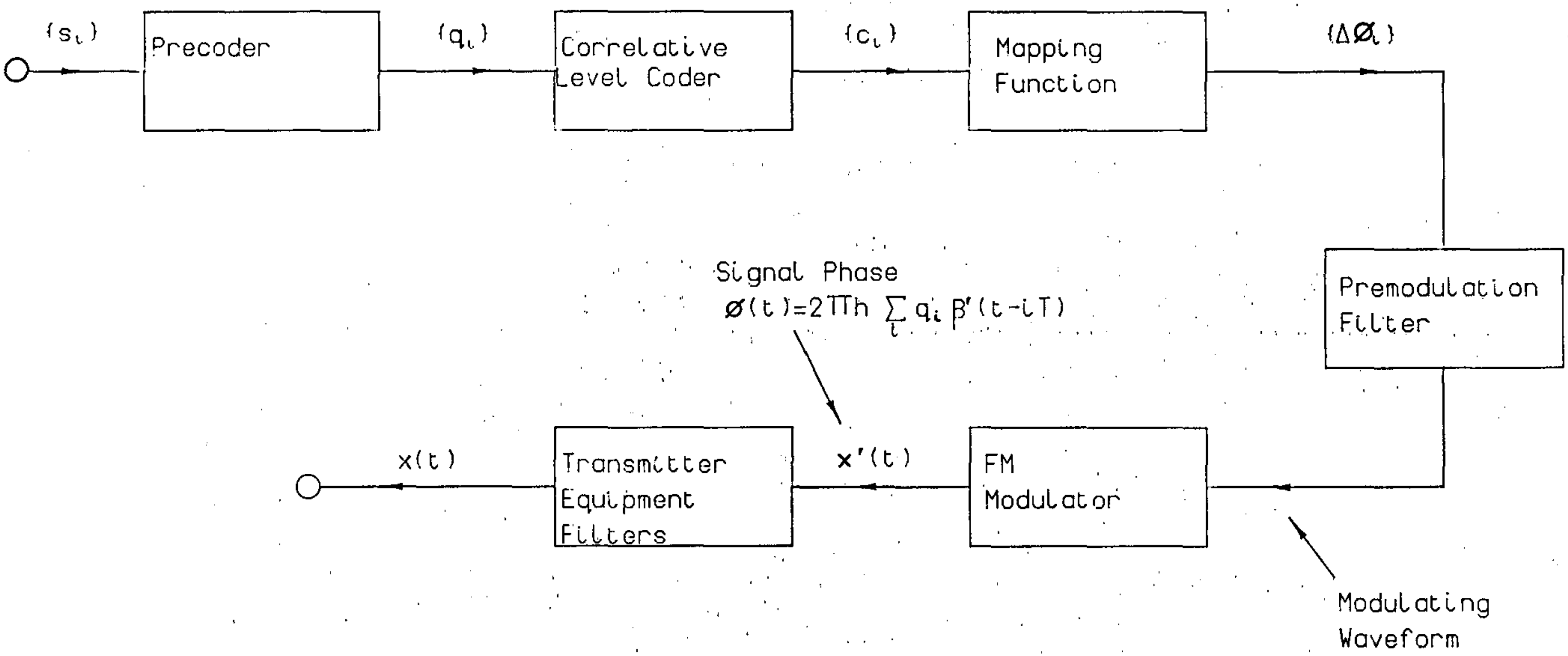


Figure A2.1 Transmitter for The CORPSK(4-7,1+D) Scheme

A3 MAXIMUM LIKELIHOOD DETECTION FOR TWO DIMENSIONAL MODULATIONS IN THE PRESENCE OF ADDITIVE WHITE GAUSSIAN NOISE

This is an analysis leading to the mathematical definition of Maximum Likelihood detection for two dimensional modulation schemes over additive white Gaussian noise, (AWGN), channels. Maximum Likelihood detection, as opposed to estimation, is considered. Detection is defined to be a process whereby a decision is made as to the value of a parameter, where that parameter has one of a given finite set of discrete possible values. In contrast, estimation is the process whereby a decision is made as to the value of a parameter, where the parameter can have any value within a given continuous range, (infinite set), of values. The analysis is restricted to schemes where the received waveform is sampled once per data symbol. Extension to multiple sampling is quite trivial.[2]

It is assumed that the received waveform r(t) in Figure A3.1 is sampled at the Nyquist rate, so that all the information in r(t), over the whole transmission period, $0 \leq t \leq nT$, is contained in the vector of received samples $R = [r_1, r_2, \ldots, r_n]$. The vector of n data symbols is given by $Q = [q_1, q_2, \ldots, q_n]$ where the $\{q_i\}$ are statistically independent and equally likely to have any of their m different possible values. The coding and mapping process produces a complex-valued signal p(t), whose sampled representation in the complex number plane is $P=[p_1, p_2, \ldots, p_n]$ where p_i is complex-valued. In general the $\{p_i\}$ are not statistically independent and p_j has m' possible values, where $m' \geq m$. The additive white Gaussian noise at the output of the demodulator, w(t), is a sample function of a stationary Gaussian random process. The noise sample at the input to the detector at time $t=jT$, $w_j=w(jT)$, is a complex sample value of a statistically independent Gaussian random variable with zero mean and variance σ^2 per component. The $\{w_j\}$ are also statistically independent of the $\{p_j\}$. The noise vector is defined as $W=[w_1,w_2,\ldots,w_n]$. The received signal is given by Equation A3.1

$$R = P + W \tag{A3.1}$$

In general the n-component vector P is a direct and unique function of the n-component row vector Q.

$$P = F(Q) \tag{A3.2}$$

P has any of its m^n possible values defined by Equation A3.2 with equal probability.

The detector has prior knowledge of the m possible values of q_j (and therefore the m^n possible values of Q), and the function F(Q) (and therefore the m^n possible values of P). For any received vector R, it is possible that any of the m^n possible values of Q, Q_ℓ, $\ell=1,2,\ldots,m^n$, was transmitted, so that the detector can never detect the value of ℓ with certainty.

Clearly the detector can make one of m^n hypotheses, H_ℓ that $Q=Q_\ell$. The detector requires a decision rule to decide which values of R lead to the acceptance of particular hypotheses H_ℓ. The vectors R,P and W, can be represented as points in an n-dimensional linear unitary vector space. The orthogonal projections of any point in the vector space onto the n orthogonal complex axes give the n complex components of the corresponding vector. The detector uses the position of R in this vector space to decide on the value of ℓ. It divides the vector space into m^n regions D_ℓ. If R lies in region D_ℓ, the detector accepts hypothesis H_ℓ, and so detects Q as Q_ℓ. The m^n regions are termed decision regions and the boundaries separating them are known as decision boundaries.

If R lies in decision region D_ℓ, this is written $R \in D_\ell$. The probability that this occurs is written $P(R \in D_\ell)$. It is assumed that R must lie in one of the decision regions. It cannot lie in more than one decision region since the regions are disjoint. In addition it cannot lie in a region separated from all the D_ℓ, since the set of all decision regions D_ℓ, $\ell=1,2,\ldots,m^n$, fills the whole of the n-dimensional unitary vector space.

The problem now is the definition of the optimal decision boundaries between the m^n decision regions, where the optimal definition minimises the probability of error in the detection of the whole message Q. In order to achieve Maximum Likelihood detection, the detector must maximise the probability of a correct decision P(C), given in Equation A3.3.[1,2]

$$P(C) = \int_{-\infty}^{\infty} P(C/R)\rho(R)dR = \int_{-\infty}^{\infty} \ldots \int_{-\infty}^{\infty} P(C/R)\rho(r_1,r_2,\ldots,r_n)dr_1,\ldots dr_n \tag{A3.3}$$

where P(C/R) is the conditional probability of a correct decision given R and $\rho(r)$ is the value of the joint probability density function of the random variables corresponding to $r_1,r_2,\ldots,r_n$ at the given values $r_1,r_2,\ldots,r_n$. From the integration limits in Equation A3.3, R ranges over all its possible values in the calculation of P(C).

$\rho(R)$ is non-negative, so $P(C)$ is maximised by maximising $P(C/R)$ for every possible value of R. For a given received vector R, $P(C/R)$ is maximised by selecting as the detected value of Q, the vector Q_j, such that[1,2]

$$P(Q_j/R) > P(Q_i/R) \quad , \quad i=1,2,\ldots,m^n, \ i\neq j \qquad \text{(A3.4)}$$

where $P(Q_i/R)$ is the conditional probability of Q_i given R. Note that this is equivalent to selecting the vector P_j such that $P_j=F(Q_j)$ from Equation A3.2. Using Bayes theorem[1,2]

$$P(Q_i/R) \quad = \quad \frac{\rho(R/Q_i)P(Q_i)}{\rho(R)} \qquad \text{(A3.5)}$$

where $P(Q_i)$ is the a priori probability of Q_i. (Since all m^m values of Q_i are equally likely, $P(Q_i)=m^{-n}$.) $\rho(R/Q_i)$ is the value of the conditional joint probability density function of the random variables $r_1,r_2,\ldots,r_n$, at the given values $r_1,r_2,\ldots,r_n$, and given the value of Q_i. Substituting Equation A3.5 in Equation A3.4, where $P(Q_i)=m^{-n}$ for $i=1,2,\ldots,m^n$, yields the decision rule of Equation A3.6.

Detect Q as Q_j where

$$\rho(R/Q_j) > \rho(R/Q_i) \quad i=1,2,\ldots,m^n, \ i\neq j \qquad \text{(A3.6)}$$

Since the noise samples $\{w_j\}$ are complex sample values of statistically independent Gaussian random variables with zero mean and variance σ^2 per real or imaginary component in the complex number plane, it follows from Equation A3.1 that r_j is a sample value of a statistically independent Gaussian random variable with mean value p_{ij} and variance σ^2 per real or imaginary component in the complex number plane. p_{ij} is the jth component of vector P_i, which is a possible value of the transmitted signal vector P. Therefore the conditional probability density function of r_j given P_i takes the form of Equation A3.7.

$$\rho(r_j/P_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-|r_j-p_{ij}|^2}{2(2\sigma^2)} \right) \tag{A3.7}$$

If Re(x) and Im(x) are, respectively, the real and imaginary components of complex number x, then $|r_j-p_{ij}|^2 = [\mathrm{Re}(r_j-p_{ij})]^2 + [\mathrm{Im}(r_j-p_{ij})]^2$. Since the $\{r_j\}$ are statistically independent, Equation A3.8 holds.

$$\begin{aligned}\rho(R/P_i) &= \rho(r_1,r_2,\ldots,r_n/P_i)\\ &= \rho(r_1/P_i)\rho(r_2/P_i)\ldots\rho(r_n/P_i)\\ &= \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-|r_j-p_{ij}|^2}{4\sigma^2}\right)\\ &= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{4\sigma^2} \sum_{j=1}^{n} |r_j-p_{ij}|^2\right)\end{aligned} \tag{A3.8}$$

But $\sum_{j=1}^{n} |r_j-p_{ij}|^2$ is simply the squared distance $|R-P_i|^2$ between the vectors R and P_i in the unitary vector space. Since, for a given value of Q_i, P_i is uniquely defined, P_i can be replaced by $F(Q_i)$ in Equation A3.8. The decision rule of Equation A3.6 can now be rewritten in terms of these distances as shown in Equation A3.9.

Detect the value of Q as Q_j such that

$$|R-P_j| < |R-P_i|, \quad i=1,2,\ldots,m^n,\ i\neq j \tag{A3.9}$$

The decision boundaries are now (n-1)-dimensional hyperplanes which bisect the lines joining the possible values of P_j, $j=1,2,\ldots,m^n$.

A decision error occurs when the chosen value P_j is not equal to P. Equivalently this occurs when the chosen value Q_j is not equal to Q. The exact calculation of the probability of error is difficult but an approximate value can be obtained by considering the minimum distance, d_{min}, between a pair of possible vectors P_a and P_b, in the unitary vector space. (For convolutionally encoded systems this minimum distance is usually termed the minimum free distance d_{free}.) The decision boundary lies at a distance $d_{min}/2$ from both points P_a and P_b in the unitary vector space. Therefore if vector P_a is actually transmitted, the probability that P_a is incorrectly detected as P_b is given by Equation A3.10.

$$P(e) = Q(\frac{d_{min}}{2\sigma}) \qquad (A3.10)$$

since the variance of the complex additive noise is σ^2 along the line connecting P_a and P_b. Also,

$$Q(y) = \int_y^\infty \frac{1}{\sqrt{2\pi}} \exp(-\tfrac{1}{2}v^2)dv$$

P(e) is accurate at high signal to noise ratios where most of the errors will be due to crossings of these minimum distance boundaries.

Clearly the probability of error P(e) refers to the whole transmitted message Q. An error in the detected value of Q is associated with errors in one or more of the n components of Q.

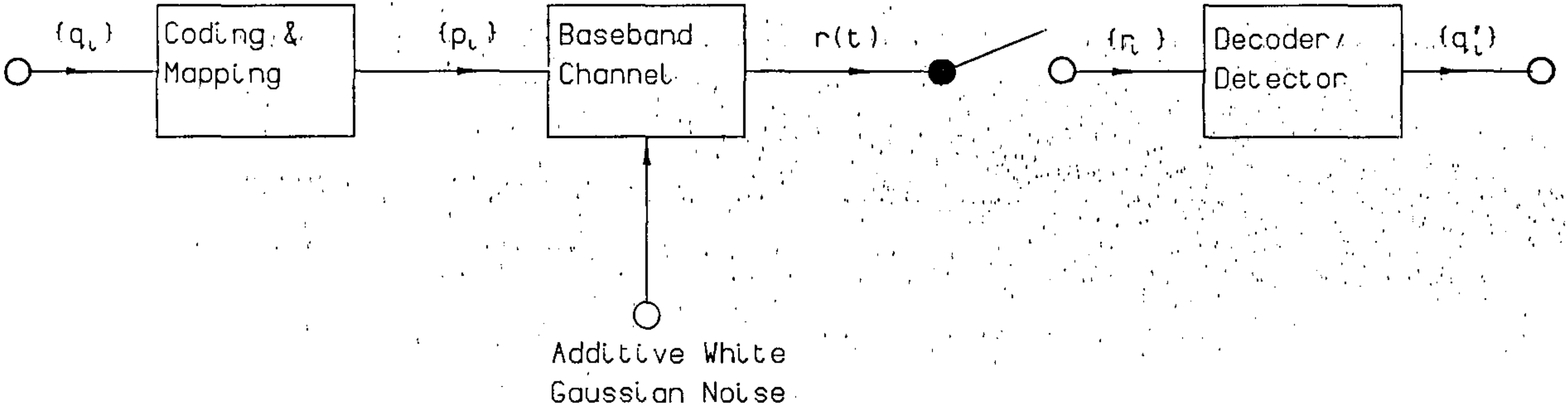


Figure A3.1 System Model

A4 COMBINED CONVOLUTIONAL CODING AND PHASE MAPPING

In general an (n_O,k_O,k) convolutional encoder over the field of binary numbers GF(2)[77] is as portrayed in Figure A4.1, where k_O is the number of binary data symbols (bits) at the encoder input at time t=iT, n_O is the number of binary code symbols at the encoder output at time t=iT, and k is the constraint length of the code in (2^{k_O})-level symbols.

The encoding equation in matrix form is given by Equation A4.1

$$C = QG \tag{A4.1}$$

Q is the semi-infinite vector of input symbols $[q_1,q_2,\ldots]$. q_i is a vector of k_O data bits $[q_i(1),q_i(2),\ldots,q_i(k_O)]$ which are the binary-valued inputs to the encoder at time t=iT. (Note that semi-infinite in this case implies that there is no explicit upper limit on i.) Similarly C is the semi-infinite vector of output symbols $[c_1,c_2,\ldots]$. c_i is a vector of n_O binary symbols $[c_i(1),c_i(2),\ldots,c_i(n_O)]$ which are the binary-valued outputs of the encoder at time t=iT. G is the semi-infinite code generator matrix given in Equation A4.2.

$$G = \begin{bmatrix} G_O, & G_1,\ldots, & G_{k-1}, & O & ; & \ldots \\ O, & G_O,\ldots, & G_{k-1}, & O & ; & \ldots \\ O, & O, & G_O, & \ldots & & \cdot \quad \cdot \\ \cdot & \cdot & \cdot & & & \cdot \quad \cdot \\ \cdot & \cdot & \cdot & & & \cdot \quad \cdot \end{bmatrix} \tag{A4.2}$$

G_ℓ, $O\leq\ell\leq k-1$, is a $k_O \times n_O$ sub-matrix. The element in the ith row and jth column of G_ℓ is $g_\ell(i,j)$, for $i=1,2,\ldots,k_0$ and $j=1,2,\ldots,n_0$, where $g_\ell(i,j)$ can have the value 0 or 1. The $\{g_\ell(i,j)\}$ are used to define the (i,j)th code sub-generator as in Equation A4.3.

$$g_{ij} = [g_0(i,j), g_1(i,j), \ldots, g_{k-1}(i,j)] \qquad \text{(A4.3)}$$

(Note that the constraint length k is such that (k-1) is the smallest integer such that $g_{k-1}(i,j)=1$ for some i and j, but $g_\ell(i,j)=0$ for all i,j when $\ell>(k-1)$.) In practical terms the output of sub-generator g_{ij} is the contribution of the ith parallel input bit sequence to the jth parallel binary-valued sequence at the output, in Figure A4.1. The determination of output sequence j is shown in Figure A4.2.

In terms of the binary-valued sequences, Equation A4.4 defines the output of the encoder at time t=iT.

$$c_i(j) = \bigoplus_{\ell=1}^{k_0} \bigoplus_{h=0}^{k-1} q_{i-h}(\ell) g_h(\ell,j) \quad , \text{ for } j=1,2,\ldots,n_0 \qquad \text{(A4.4)}$$

where $\bigoplus$ denotes MODULO-2 addition.

The encoder thus defined can be regarded as an output-independent Finite-State Machine as depicted in Figure A4.3, where Φ_i is the state of the machine at time t=iT.[72] The state Φ_i is the contents, $q_{i-k+1}, q_{i-k+2}, \ldots, q_{i-1}$, of the storage elements in the encoder at time t=iT. The output-independent Finite-State Machine is such that the output symbol c_i at time t=iT, and the machine state Φ_{i+1} at time t=(i+1)T, are completely defined by the input symbol q_i, and state Φ_i, at time t=iT.

The code itself is usually represented diagrammatically in the form of a code trellis diagram. The initial diverging portion of such a diagram for a code with 16 states where each input symbol q_i has one of four possible values, is shown in Figure A4.4. It has a root node defining the initial state of the machine, Φ_1, before transmission begins. Usually this is the all-zero state where all the memory elements in the encoder contain zeroes. The vertical axis in the diagram denotes the state value while the horizontal axis denotes time, graduated in integer multiples of T seconds where T is the symbol interval. The code trellis diagram portrays the code sequences for all possible input sequences Q, given the initial encoder state. The symbol $\Phi_i(j)$ denotes the existence of state j at time t=iT. Four branches extend from the root node, and terminate at four different nodes, which are the possible states, $\Phi_2(j)$, at time t=2T. Each such branch is for one of the four possible vectors of data symbols $q_1=[q_1(1),q_1(2)]$. Alongside each branch, the code vector, $c_1=[c_1(1),c_1(2),c_1(3)]$, is usually displayed, (not in Figure A4.4). Thus, given a particular input sequence Q and the initial state Φ_1, a path can be traced through the diagram which gives the associated code sequence C, and the state which the encoder has at each sampling instant.

Figure A4.4 shows the diverging part of the code trellis diagram where the number of states, (on the vertical axis), is increasing with time. Since the machine is finite-state, the number of states must reach an upper limit. The time taken to reach this limit is a function of the constraint length k of the code, since this determines the number of states in the machine. Equation A4.5 defines the total number of possible states, N_s.

$$N_s = (2^{k_o})^{(k-1)} = 2^{k_o(k-1)} \qquad (A4.5)$$

Clearly, since the number of states is increasing by a factor (2^{k_o}) at each stage during the expanding part of the code trellis diagram, this part is ℓT seconds in length where

$$N_s = 2^{k_o(k-1)} = (2^{k_o \ell})$$

so that

$$\ell = k-1 \qquad (A4.6)$$

Thus at time $t=(\ell+1)T$, all N_s states are shown as in Figure A4.4. For $t>\ell T$, branches occur between the N_s initial states at time $t=jT$ and the N_s final states at time $t=(j+1)T$. There are k_o branches leaving each initial state and there are k_o branches entering each final state. This is seen most clearly by considering the $\{q_j\}$ defining a particular state, (the contents of the encoder storage elements). For a given initial state at time $t=iT$, the k_o final states into which the branches diverge are given by the vectors in Equation A4.7.

$$\Phi_{i+1} = F\{[q_{i-k+2},\ldots,q_{i-1},q_i]\} \qquad (A4.7)$$

where $[q_{i-k+2},\ldots,q_{i-1}]$ is fixed and q_i has one of its (2^{k_o}) possible values. $F\{.\}$ denotes a function of the vector elements, which gives the integer value Φ_{i+1}. The function is defined in the relevant sections of Chapter 2. The initial states from which these k_o final states emanate are given by Equation A4.8 at time $t=iT$.

$$\Phi_i = F\{[q_{i-k+1},q_{i-k+2},\ldots,q_{i-1}]\} \qquad (A4.8)$$

where $[q_{i-k+2},\ldots,q_{i-1}]$ is again fixed, but in this case q_{i-k+1} has one of its (2^{k_o}) possible values. This is depicted in Figure A4.5.

Clearly, for the portion of the code trellis diagram where N_s states exist, a segment of the diagram for one isolated symbol interval, $iT\leqslant t\leqslant (i+1)T$, can be split up into a number of parts called sub-trellises, in each of which the (2^{k_o}) initial states completely define the (2^{k_o}) final states, as shown in Figure A4.5. Clearly N_{k_o} such sub-trellises exist,

$$N_{k_o} = N_s 2^{-k_o} = 2^{k_o(k-2)} \qquad (A4.9)$$

The close of transmission causes the convergence of the code trellis diagram to a single final state, (usually the all-zero state), in an analogous manner to the start of transmission, but in reverse. Clearly this convergence takes ℓT symbol intervals.

Maximum Likelihood decoding, (see Appendix A3), of convolutional code sequences is possible using the Viterbi Algorithm, (VA).[63] The VA finds the Maximum Likelihood path through the code trellis diagram for a particular received sequence of samples R in the presence of additive white Gaussian noise, (AWGN), thereby minimising the probability of choosing an incorrect sequence. This is not the same as minimising the bit error rate, (BER), but in practice the BER is very nearly minimised.[19] The VA is adept at exploiting the very regular structure of the code. For one thing it uses the sub-trellises. This allows the splitting of the hardware into N_{k_o} parallel processing units, each dealing with a particular sub-trellis. Also, and more importantly, it exploits a property concerning the convergence of k_o branches into each final state during the time for which there are N_s states. The VA only retains one of the k_o paths converging into a given final state. The algorithm selects the path into the final state associated with the greatest likelihood, (Appendix A3). The remaining (k_o-1) converging paths are discarded with the following justification. If a path converging into a final state has a lower likelihood than another path converging into the same state then, since the paths are indistinguishable in the future, the converging path with the lower likelihood will always have the lower likelihood, and therefore cannot be the Maximum Likelihood path.

Coded Trellis Modulation (CTM) is a broad class of techniques which regards the coding and modulation processes as a single entity.[12,19-26] The scheme considered here is convolutionally encoded and phase mapped 8PSK, (coded 8PSK), introduced by Ungerboeck.[20] The idea is based on the argument that if the number of points in the signal set, (the $\{p_i\}$ in Figure A4.6), is increased, while keeping the data rate constant, it is possible to gain a considerable advantage in tolerance to noise over the original unexpanded scheme.[20,24] In the case considered four-level data are convolutionally encoded to produce eight-level code symbols which are mapped onto eight-phase signal elements. In general such schemes involve rate-m/(m+1) coding of 2^m-level data, followed by mapping onto a signal set with 2^{m+1} points. See Figure A4.6. Since there is no change in the symbol transmission rate, the coded system will have approximately the same bandwidth as the uncoded system.[21] (In the case considered, coded 8PSK will have approximately the same bandwidth as uncoded QPSK.) This assumes that the correlation in the transmitted 2^{m+1}-phase signals due to the coding has no effect on the bandwidth. The theoretical gains in tolerance to noise for such systems are discussed in References 12 and 19 to 26. The requirement of soft-decision decoding[19] leads to the use of codes designed to achieve an optimum unitary rather than Hamming distance. For four-phase signalling the optimisation is equivalent for both distance measures, but for signal sets with more points this is no longer true.[19]

The remaining problem is the efficient allocation of the code symbols to points in the expanded signal set by means of a mapping function. Defined mathematically the problem is the maximisation of the minimum unitary distance between possible sequences of the $\{p_i\}$, (see Figure A4.6). This distance is usually termed the Free Euclidean Distance d_{free}. Maximisation of d_{free} ensures an asymptotic coding gain which is a function of the minimum distances for the codes and uncoded schemes since, at high signal to noise ratios, the major cause of errors will be the selection of a possible transmitted sequence of the $\{p_i\}$ at minimum distance from the correct sequence.[19] The bit error probability is lower-bounded by Equation A4.10[20]

$$P(e) \geq N.Q(d_{free}/2\sigma) \qquad \text{(A4.10)}$$

where N is the average number of bit errors due to selection of an incorrect sequence of the $\{p_i\}$ at minimum distance from the correct sequence, and Q(y) is the Gaussian Error Function,

$$Q(y) = \int_{y}^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-\tfrac{1}{2}v^2)dv \qquad \text{(A4.11)}$$

σ^2 is the variance of the noise samples along each axis in the complex number plane.

If d_0 is the minimum distance between signals in the uncoded system, (for the same average signal power), then the asymptotic coding gain, G_a, using an optimum decoding scheme, is given by Equation A4.12.

$$G_a = 20\log_{10}(d_{free}/d_0) \quad \text{(A4.12)}$$

Ungerboeck's approach is to view the coding and mapping as a single entity in the code trellis diagram. The problem is then reduced to assigning points $\{p_i\}$ to branches in the code trellis diagram. The first step in the process is the partitioning of the set of all possible values of p_i into subsets, where each partition splits the original subset into two new subsets with an equal number of points in each. Each partition produces new subsets with a greater minimum Euclidean distance between its constituent points, than in the original. Ungerboeck now assigns binary code symbols to the partitioned subsets of the possible values of p_i. In this way he defines the mapping function of the code symbols $\{c_i\}$, onto the complex numbers $\{p_i\}$ for every code. Figure A4.7 shows how this mapping function is developed, as the original set of possible values of p_i is progressively partitioned. Clearly after n_0 partitions the allocation of code symbols is complete. It is important to note here that this simply defines the mapping function for all codes. It does not optimise the mapping for any code. Ungerboeck's approach is to optimise the code, given the mapping function. In fact, his initial work did not explicitly use convolutional codes, (although the codes he developed were convolutional codes). In very simple terms the strategy of assigning points in the signal set, (and therefore code symbols), to branches in the code trellis diagram is as follows. The branches of paths in the code trellis diagram which do not converge very quickly are assigned points in the signal subsets A or B of Figure A4.7, since the points in these subsets are quite close to one another.

The subsets of possible values of p_i with larger distances, (e.g. sets C and D), are assigned to paths which converge more quickly. In particular the branches of paths that converge most quickly, (the minimum distance paths), must be assigned, as far as possible, to the subsets D. Clearly, apart from certain trivial examples, the assignment algorithm is complex and requires a rigorous computer search.[20] Ungerboeck[20] and Clark and Cain[19] describe a "by-hand" assignment technique for some very simple code trellis diagrams.

The approach taken by Hui and Fang[12] in producing the optimal codes used extensively in this study, is slightly different. They standardise on a straight binary mapping of the encoder output vector $c_i=[c_i(1),c_i(2),...,c_i(n_O)]$ as given in Equation A4.12, rather than using the set partitioning method.

$$c_i = 2^{n_O-1}c_i(1) + \ldots + 2^O c_i(n_O) \qquad \text{(A4.12)}$$

They then use a code search algorithm to optimise the code given Equation A4.12. It must be noted that this search is very laborious because the code produced by a combined coding and phase mapping scheme does not have the group property.[19,24] This means that in the calculation of d_{free}, one of the paths in the code trellis diagram cannot be fixed as the all-zero path while varying the other path. Instead the comparisons of paths to determine d_{free} must include all possible different pairs of paths.

Finally, a number of points are worth noting with regard to CTM schemes. The quite considerable gains which are possible by moving to expanded signal sets may well cause synchronisation problems.[49] For example, carrier frequency and phase-tracking synchronisation for

8PSK at typical signal to noise ratios for the coded scheme may create real problems. Ungerboeck considers this in some detail.[20] Also, the codes designed by Hui and Fang[12] are not tranparent to phase inversions, in the sense that for a transparent code, a polarity inversion at the decoder input simply causes a polarity inversion at the decoder output, after an initial transient due to the encoder's memory.[19] Ungerboeck notes that the use of systematic codes, (where the $\{c_i\}$ explicitly contain the data symbols $\{q_i\}$), with feedback phase-differential coding, (that is, precoding), can resolve phase ambiguity.[20] Unfortunately phase ambiguities for non-transparent codes cannot be resolved using this method. It has been noted though, that phase ambiguities can be detected quite easily.[19]

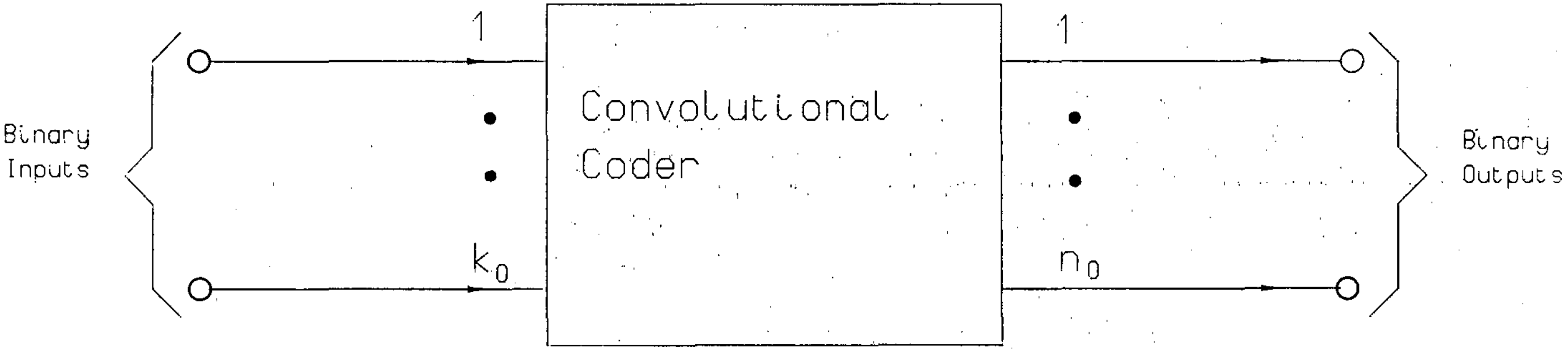


Figure A4.1 Generalised Convolutional Coder

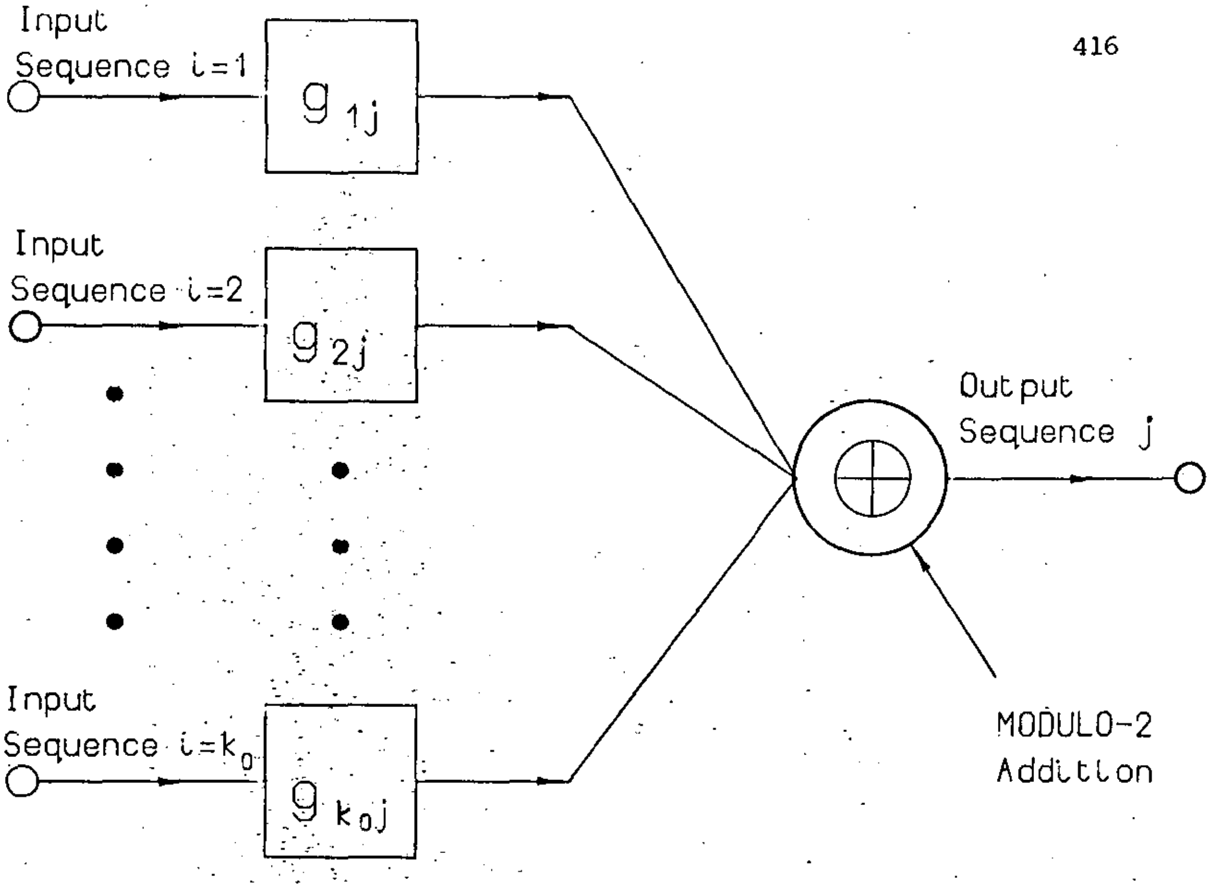


Figure A4.2 Determination of Output Sequence j

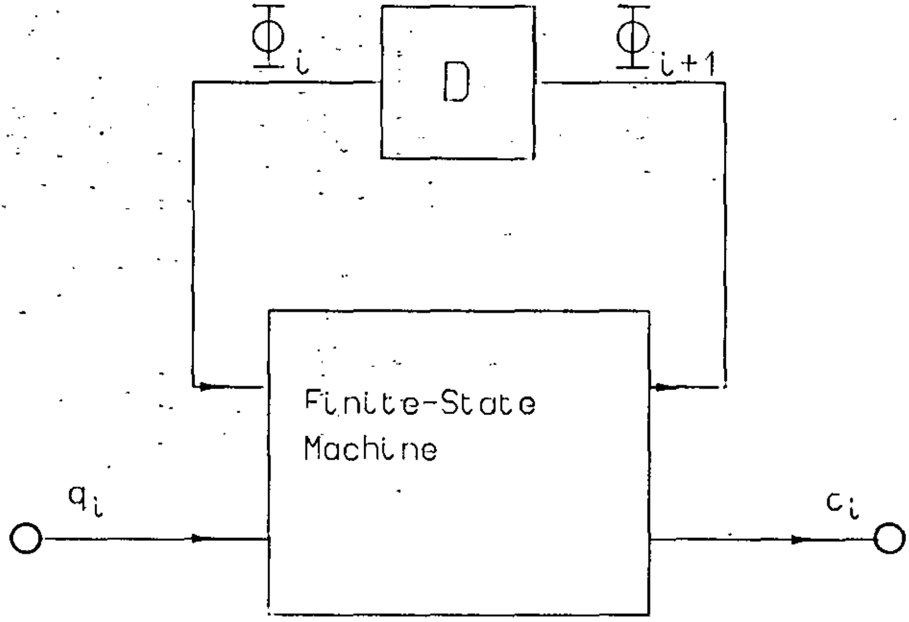


Figure A4.3 Output-Independent Finite-State Machine

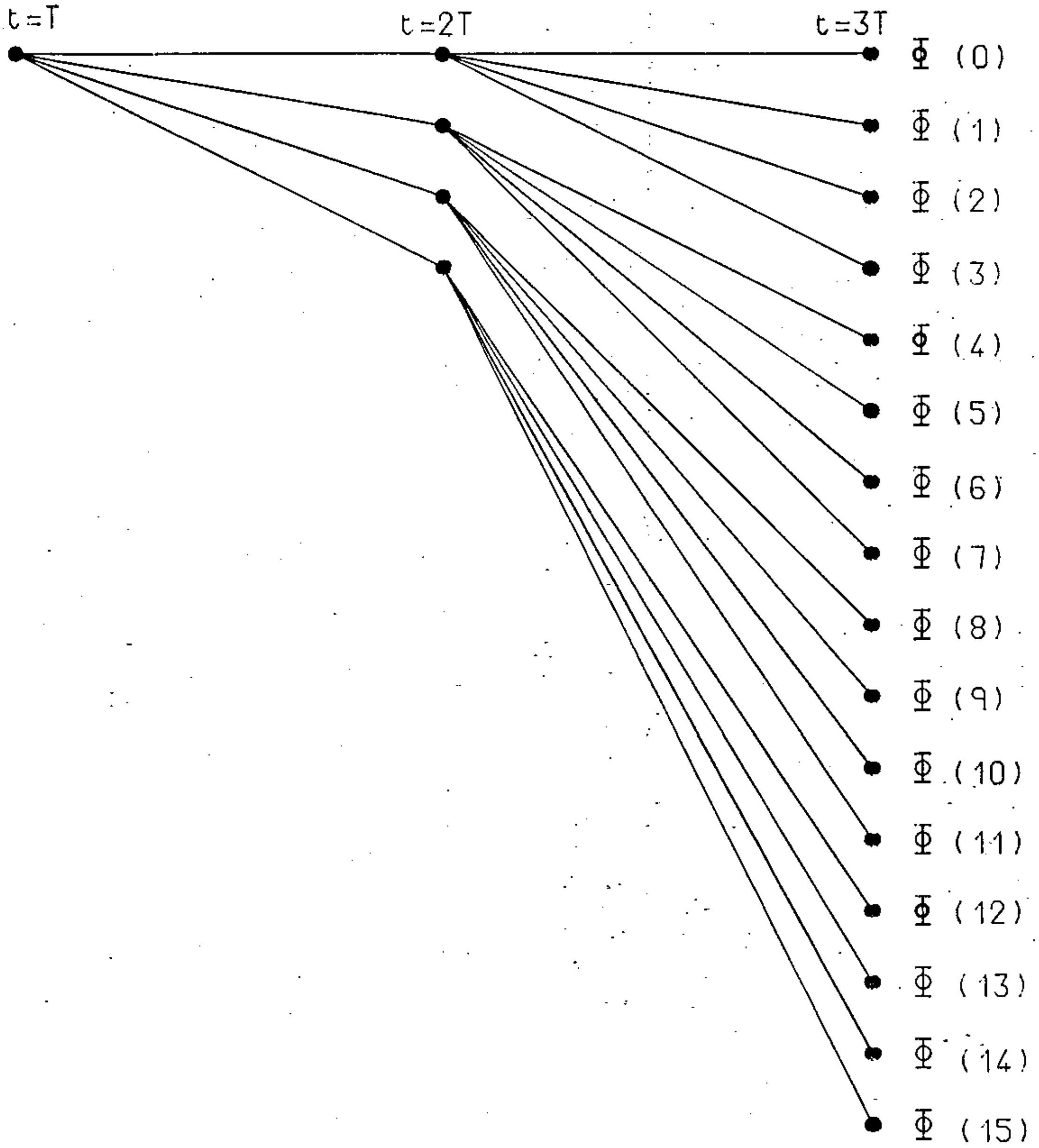


Figure A4.4 Expanding Section Of The Code Trellis Diagram
for a Convolutional Code with 16 States
and Four-Level Data

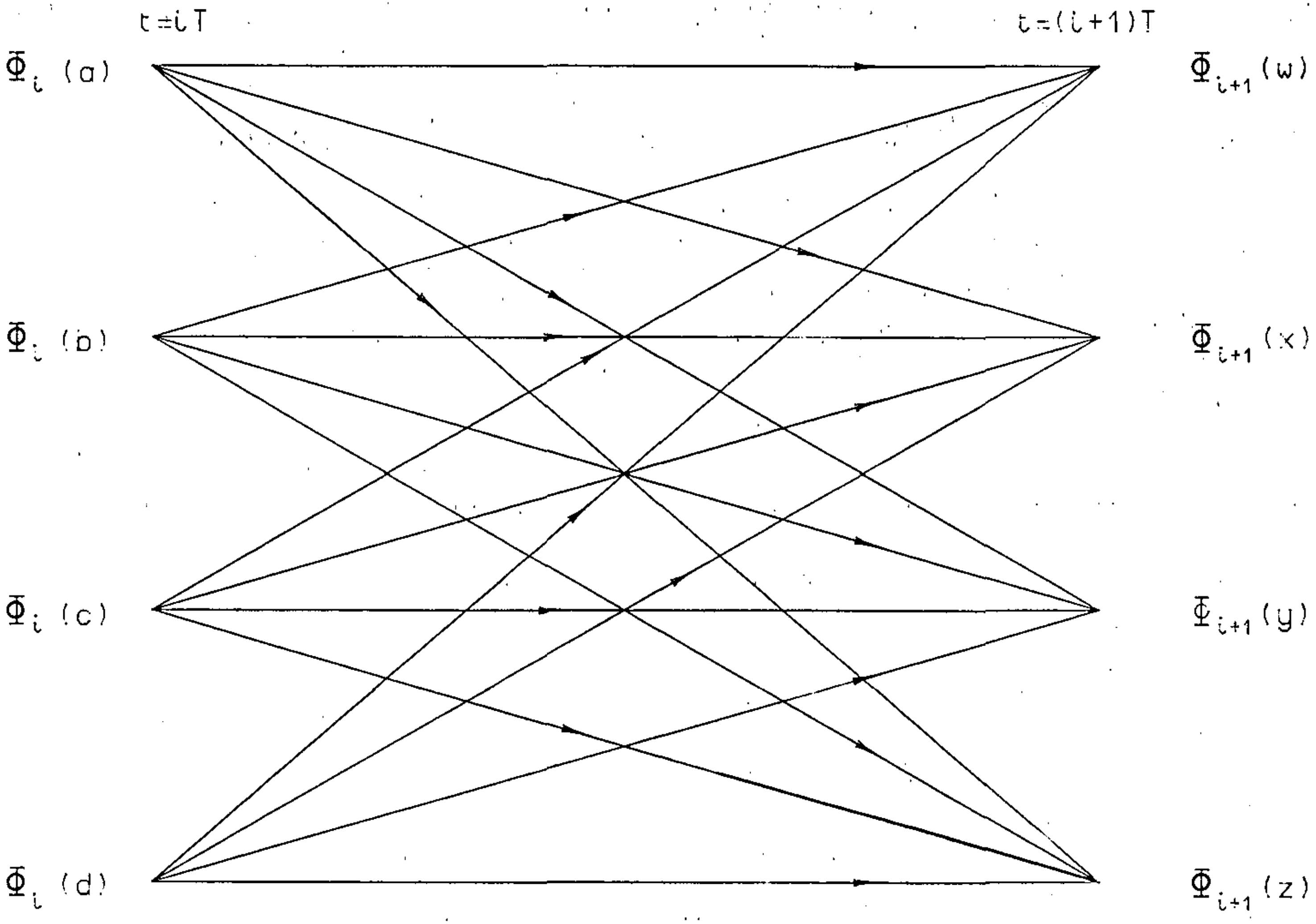


Figure A4.5 Sub-Trellis for a Code with Four-Level Data Symbols

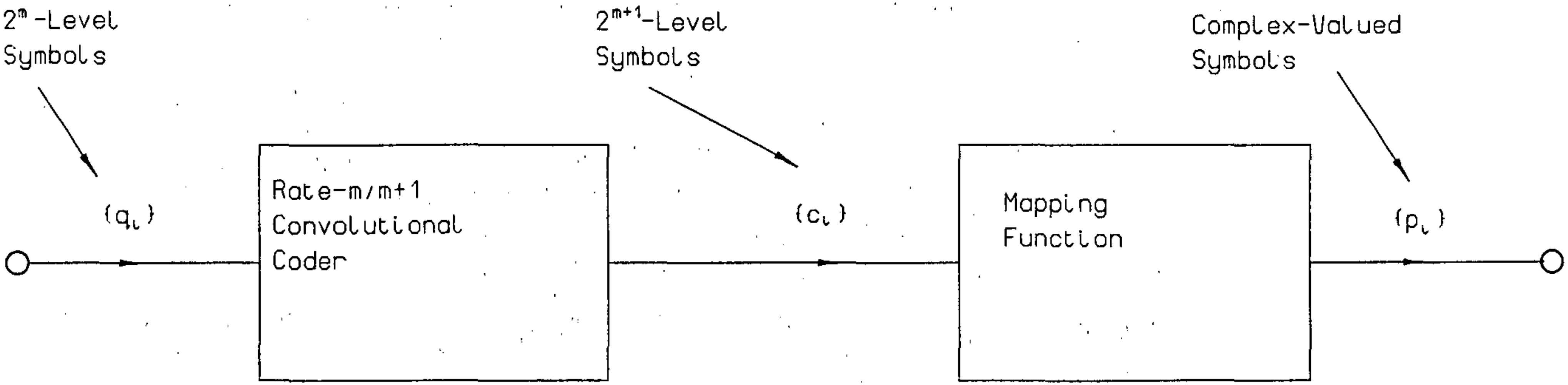


Figure A4.6 Generalised Combined Coding and Modulation Technique

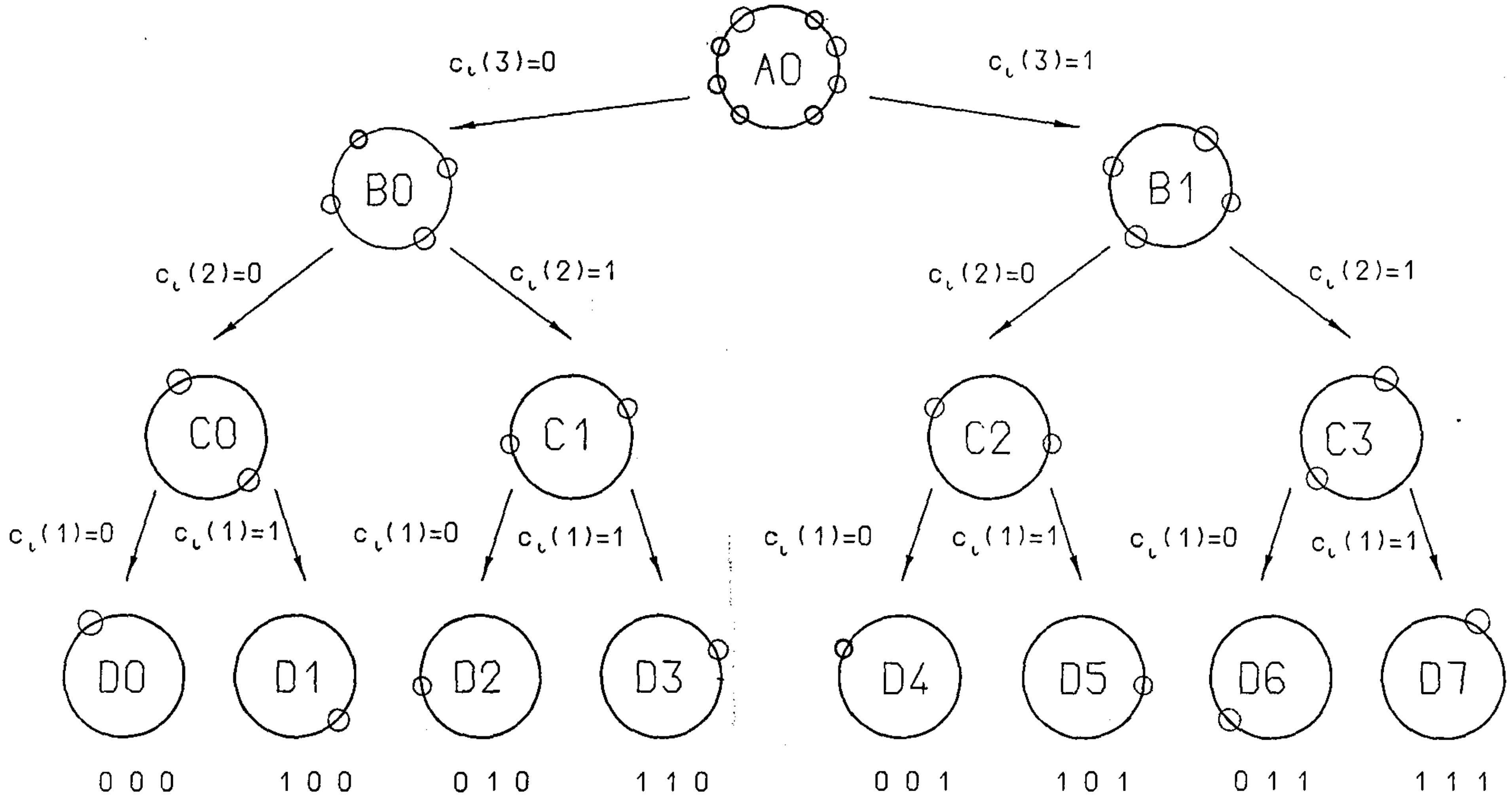


Figure A4.7 Mapping by Set Partitioning for Coded 8PSK
Assignment of Coder Output Symbols to Points in The Signal Set

A5 SIGNAL TO NOISE RATIO DEFINITIONS AND SIMULATION TECHNIQUES

The definition of signal to noise ratio is developed. Initially the general case is considered, followed by two special cases where the definitions are analytical, requiring no averaging operations within the computer programs. A short description of the techniques used within the computer simulations is given, and the method by which the accuracy of the results is gauged, is given.

In general the signal to noise ratio ψ(dB) is given by Equation A5.1.

$$\psi(\text{dB}) = 10\log_{10}\left[\frac{E(|p_i|^2)}{E(|w_i|^2)}\right] \qquad (A5.1)$$

where E(x) is the Expected Value of quantity x, p_i is the complex number derived from the appropriate mapping function of Chapter 2, and w_i is the noise component of the ith received sample, r_i. The additive white Gaussian noise in the channel has a two-sided power spectral density of $\frac{1}{2}N_0$ and a mean of zero. Although tolerance to additive white Gaussian noise may not be an accurate measure of tolerance to noise over satellite channels, the relative tolerances to additive white Gaussian noise of different data transmission schemes are a good measure of their relative tolerances to additive noise over satellite channels.[1,2] If H(f) is the frequency response of the receiver filtering, Equation A5.2 gives the variance of the noise at the detector input along either the real or imaginary axis in the complex number plane.

$$\sigma^2 = \frac{N_0}{2}\int_{-\infty}^{\infty} |H(f)|^2 df \qquad \text{(A5.2)}$$

Therefore, since the noise samples are zero-mean and statistically independent, the resultant noise variance is given by Equation A5.3.

$$\begin{aligned} E[|w_i|^2] &= E[(\text{Re}(w_i))^2 + (\text{Im}(w_i))^2] \\ &= E[\text{Re}(w_i)^2] + E[\text{Im}(w_i)^2] \\ &= 2\sigma^2 \end{aligned} \qquad \text{(A5.3)}$$

Using Parseval's Theorem[1], Equation A5.2 can be rewritten in terms of the impulse response of the receiver filtering.

$$\sigma^2 = \frac{N_0}{2}\int_{-\infty}^{\infty} |h(t)|^2 dt \qquad \text{(A5.4)}$$

A similar technique can be used to calculate $E[|p_i|^2)$ over the whole transmission period where $|p_i|^2 = (\text{Re}(p_i))^2 + (\text{Im}(p_i))^2$.

In all cases ψ(dB) is converted into E_b/N_0(dB) where E_b is the average energy per data bit transmitted, in order to compare all schemes fairly, for all receiver filter configurations. From Equation A5.4

$$N_0 = \frac{2\sigma^2}{\int_{-\infty}^{\infty} |h(t)|^2 dt} \qquad \text{(A5.5)}$$

Equation A5.6 is the equation for the calculation of E_b.

$$E_b = 1/(2\ell)\int_0^{\ell T} |p(t)|^2 dt \qquad \text{(A5.6)}$$

($p(t)$ is the continuous waveform, of which the $\{p_i\}$ are sample values.) ℓ is the number of transmitted data symbols, each symbol carrying two bits of information in all cases.

In the simulation tests which utilise the filtered models, (see Chapter 2), the above described calculations are actually performed within the computer programs. For the perfect-channel models, an analytical method can be used making these calculations unnecessary. The frequency responses for the two receiver filters used in the perfect-channel simulations are given in Equation A5.7.

$$H(f) = \begin{cases} \sqrt{T} \; ; & f \leq 1/(aT) \\ 0 \; ; & f > 1/(aT) \end{cases} \qquad \text{(A5.7)}$$

where a=2 for single sampling systems, (sampling instants t=iT), and a=1 for double sampling systems, (sampling instants iT/2). Equation A5.5 can be used to calculate N_O in both cases, as shown in Equation A5.8 for single sampling, and Equation A5.9 for double sampling.

$$N_O = \frac{2\sigma^2}{\int_{-\frac{1}{2}T}^{\frac{1}{2}T} |H(f)|^2 \, df} = 2\sigma^2 \qquad \text{(A5.8)}$$

$$N_O = \frac{2\sigma^2}{\int_{-1/T}^{1/T} |H(f)|^2 \, df} = \sigma^2 \qquad \text{(A5.9)}$$

In both cases $2E_b = E[|p_i|^2]$. The noise variance, σ^2, is set by the Gaussian random number generator, to be described later. Equation A5.1 can now be used to define the signal to noise ratio in terms of E_b/N_O for the two filtering arrangements. Equation A5.10 defines ψ_s(dB), the signal to noise ratio for single sampling systems, while Equation A5.11 defines ψ_d(dB), the signal to noise ratio for double sampling systems,

$$\psi_s(\text{dB}) = 10\log_{10}(2E_b/N_O) \qquad \text{(A5.10)}$$

$$\psi_d(\text{dB}) = 10\log_{10}(E_b/N_O) \qquad \text{(A5.11)}$$

The signal to noise ratios determined in the simulations are adjusted in all cases to give curves of bit error rate, (BER), against E_b/N_O.

The computer simulations use a Numerical Algorithms Group (NAG) random number generator subroutine to generate both the random data, $\{s_i\}$, using a uniform distribution, and the additive noise samples, $\{w_i\}$, using a Gaussian random number generator with zero mean. All programs were written in FORTRAN 77. The noise variance is varied to produce different signal to noise ratios. The range of signal to noise ratio is adjusted to produce bit error rates in the range 10^{-1} to 10^{-4}. The number of transmitted symbols is adjusted to produce greater than one hundred isolated error bursts wherever possible. An error burst is defined as follows. Following an incorrectly detected symbol, if twenty or more subsequent bits are detected correctly, the next incorrectly detected bit is considered to be the start of a new error burst. Otherwise the bit in error is counted as part of the previous burst. The figure of twenty is large enough to ensure that the first error in a burst is independent of all errors in a previous burst. In some cases, at the lower end of the range of BER, computing-time restrictions led to the production of less than one hundred bursts, affecting the accuracy of the results.

The method used to gauge the accuracy of the results is now given. Assume that the number of statistically independent errors occurring during a test N_b, is equal to the number of error bursts which occur. (This may lead to a pessimistic estimate, since independent errors may occur within error bursts as defined above.) Also, let the average error burst probability be p_{av}, and the number of transmitted data symbols be ℓ. Then

$$N_b = p_{av}.\ell \qquad \text{(A5.12)}$$

It has been shown[95,96] that if the errors are statistically independent, for $N_b > 30$, and $p_{av} << 1$, and if an accuracy of no more than 20% is required for the confidence limit, then it can be assumed that the number of error bursts has a Gaussian probability density function with mean $\mu = N_b$ and variance $\eta^2 = N_b$. For a given value of $p_{av} > 0$, the 95% confidence limit for the value of p_{av} is given in Equation A5.13.[95,96]

95% confidence limit is

$$\pm(2\eta p_{av})/\mu = \pm 2p_{av}/\sqrt{N_b} \qquad \text{(A5.13)}$$

The limit is expressed as a deviation from the given value of p_{av}. An approximate accuracy in the tolerance to noise is given for low bit error rates in the results discussion sections, based on the above analysis.

A6 SYNDROME DECODING THEORY

The Invariant-factor Decomposition Theorem[77] is introduced, leading to the determination of the inverse coder and syndrome-former for the code used in Chapter 5, (Code 1 in Table 2.5.1). A description of general syndrome decoding is then given.

Some definitions of Appendix A4 must be extended to develop them in a form suitable for this analysis. The code sub-generator definition developed in Appendix A4 is restated here in Equation A6.1 for a general (3,2,k) convolutional code. For further details see Appendix A4.

$$g_{ij} = [g_0(i,j), g_1(i,j), \dots, g_{k-1}(i,j)] \qquad \text{(A6.1)}$$

$$\text{for } i=1,2$$

$$\text{and } j=1,2,3$$

For the purposes of this work g_{ij} is given in terms of a polynomial in the delay operator D as shown in Equation A6.2.

$$g_{ij}(D) = g_0(i,j) + g_1(i,j)D + \dots + g_{k-1}(i,j)D^{k-1} \qquad \text{(A6.2)}$$

The $\{g_\ell(i,j)\}$ are binary-valued.

A code generator matrix can be defined as in Equation A6.3.

$$G(D) = \begin{bmatrix} g_{11}(D) & g_{12}(D) & g_{13}(D) \\ g_{21}(D) & g_{22}(D) & g_{23}(D) \end{bmatrix} \qquad \text{(A6.3)}$$

The elements of G(D) are in the ring of rational polynomials F[D].[77,78] (G(D) is said to be a matrix over F[D],)

For Code 1 this is

$$G(D) = \begin{bmatrix} D, & 1+D+D^2, & 0 \\ 1+D^2, & D^2 \quad , & D \end{bmatrix} \qquad \text{(A6.4)}$$

The Invariant-factor decomposition of G(D) is given in Equation A6.5.[77]

$$G(D) = A(D)\Gamma(D)B(D) \tag{A6.5}$$

where A(D) is a 2×2 matrix over F[D] with an inverse $A^{-1}(D)$ over F[D], and B(D) is a 3×3 matrix over F[D] with an inverse $B^{-1}(D)$ over F[D]. $\Gamma(D)$ is a 2×3 matrix over F[D] of the form $[\Gamma_1(D), O]$ where $\Gamma_1(D)$ is a diagonal matrix consisting of the invariant factors $Y_i(D)$ of G(D), which are elements of F[D].[77]

$$\Gamma_1(D) = \mathrm{diag}[Y_1(D), Y_2(D)] \tag{A6.6}$$

A coder with a feedback-free inverse, thus avoiding catastrophic error propagation[77], has $Y_i(D) = 1$, i=1,2. A method of column and row manipulations of G(D) is used to determine A(D), $\Gamma(D)$ and B(D).[77] Equivalently, the inverse coder, $G^{-1}(D)$, can be defined as[77]

$$G^{-1}(D) = B^{-1}(D)\Gamma^{-1}(D)A^{-1}(D) \tag{A6.7}$$

where $\Gamma^{-1}(D)$ is a diagonal 3x2 matrix of the form $[\Gamma_1^{-1}(D), O]^T$ where

$$\Gamma_1^{-1}(D) = \mathrm{diag}[(1/Y_1(D), 1/Y_2(D)] \tag{A6.8}$$

The matrices A(D), $\Gamma(D)$, and B(D), can be inverted and inserted in Equation A6.7 to give $G^{-1}(D)$. This has been done for Code 1, yielding Equation A6.9.

$$G^{-1}(D) = \begin{bmatrix} D^2 & 1+D+D^2 \\ 1+D, & D \\ D^2+D^3, & 1+D^3 \end{bmatrix} \tag{A6.9}$$

The syndrome-former, $H^T(D)$, by definition generates a code which is the null-space of the code generated by G(D). Mathematically

$$G(D).H^T(D) = O \qquad (A6.10)$$

$H^T(D)$ is not a unique matrix.[76,77] Equation A6.10 has been used to generate a syndrome-former for Code 1. The result is given in Equation A6.11.

$$H^T(D) = [D+D^2+D^3, D^2, 1+D+D^4] \qquad (A6.11)$$

A generalised syndrome decoder can be split into two parts, a codeword estimator and the inverse coder $G^{-1}(D)$, as depicted in Figure A6.1. The signals shown in Figure A6.1 are in polynomial form. The chosen notation, also used in Chapter 5, is in contrast to that in the remainder of the thesis, which is described in Appendix A4. The change has been made to facilitate the use of the definitions of $G(D)$, $G^{-1}(D)$, and $H^T(D)$ given in Equations A6.3, A6.9, and A6.11, respectively, whereby the analysis for syndrome decoding is much simplified. $R(D)$ is a polynomial in D and is the noisy received signal. $R(D) = r_1 + r_2D + \ldots + r_iD^{i-1}$ at time $t=iT$, where r_j is complex-valued. The threshold test operates separately on each individual element r_j of $R(D)$. The possible value of r_j in the absence of noise, p_j', which is nearest to r_j in the complex number plane, is found. p_j' is mapped onto the vector of binary code symbols $[c_j''(1), c_j''(2), c_j''(3)]$. (This mapping is the inverse of the mapping which at the transmitter converts the vector of code symbols $[c_j(1), c_j(2), c_j(3)$ onto the complex number p_j (see Figure 2.5.4). The sequence of these code symbols is given by the vector of polynomials in the delay operator D, $C''(D)=[C_1''(D), C_2''(D), C_3''(D)]$ where $C_\ell''(D) = c_1''(\ell)+c_2''(\ell)D+\ldots+c_i''(\ell)D^{i-1}$. The sequence of code symbols given by the vector $C''(D)$ may not be one that can be generated by the coder.

This is because noise may change the received samples $\{r_i\}$ such that some of the values of the binary code symbols $\{c''_j(\ell)\}$ are not the same as those at the transmitter. The codeword estimator converts C"(D) into a three-component vector of polynomials in D, C'(D). C'(D) could have been generated by the coder, and it should ideally be the same as the sequence of code symbols generated at the transmitter, given by the three-component vector of polynomials C(D). As for C"(D), $C'(D)=[C'_1(D),C'_2(D),C'_3(D)]$ where $C'_\ell(D)=c'_1(\ell)+c'_2(\ell)D+\ldots+c'_i(\ell)D^{i-1}$, and $C(D)=[C_1(D),C_2(D),C_3(D)]$ where $C_\ell(D)=c_1(\ell)+c_2(\ell)D+\ldots+c_i(\ell)D^{i-1}$. The $\{c'_j(\ell)\}$ and the $\{c_j(\ell)\}$ are binary valued. The output of the inverse coder is the two-component vector $Q'(D)=[Q'_1(D),Q'_2(D)]$ where $Q'_\ell(D)=q'_1(\ell)+q'_2(\ell)D+\ldots+q'_i(\ell)D^{i-1}$. The $\{q'_j(\ell)\}$ are binary valued. The output of the inverse coder at time t=iT is the two-component vector $[q'_j(1),q'_j(2)]$. This is uniquely related to the four-level detected data symbol q'_j by the Gray code mapping of Table 2.1.1.

The major complexity in any syndrome decoder lies in the codeword estimator, (although, in a practical implementation, the codeword estimator may not be a distinguishable or separate function). The vector C"(D) is related to the correct vector C(D) by the three-component vector E(D), called the error vector. E(D) is a vector of polynomials in the delay operator D. $E(D)=[E_1(D),E_2(D),E_3(D)]$ where $E_\ell(D)=e_1(\ell)+e_2(\ell)D+\ldots+e_i(\ell)D^{i-1}$. The symbols $\{e_j(\ell)\}$ are binary-valued.

$$C''(D) = C(D) \oplus E(D) \qquad \text{(A6.12)}$$

$\oplus$ denotes MODULO-2 addition.

The sequence of binary syndrome symbols in polynomial form, denoted $\beta(D)$, is given by Equation A6.13. ($\beta(D)=\beta_1+\beta_2D+\ldots+\beta_iD^{i-1}$ where β_j is binary-valued.)

$$\beta(D) = C''(D)H^T(D) \qquad (A6.13)$$

The encoding equation at the transmitter is given by Equation A6.14 Q(D) is the two-component vector $[Q_1(D), Q_2(D)]$ where $Q_\ell(D) = q_1(\ell) + q_2(\ell)D + \ldots + q_i(\ell)D^{i-1}$. The data symbol q_j is given by the Gray code mapping of the two-component vector of binary-valued symbols $[q_j(1), q_j(2)]$, (see Table 2.1.1).

$$C(D) = Q(D)G(D) \qquad (A6.14)$$

Substituting Equation A6.14 in Equation A6.13, and incorporating Equation A6.12,

$$\begin{aligned}\beta(D) &= [C(D) \oplus E(D)]H^T(D)\\ &= E(D)H^T(D) \oplus Q(D)G(D)H^T(D) \qquad (A6.15)\end{aligned}$$

But from Equation A6.10, the last term on the right-hand side of Equation A6.15 vanishes leaving Equation A6.16.

$$\beta(D) = E(D)H^T(D) \qquad (A6.16)$$

Therefore the syndrome sequence is independent of the actual transmitted code symbols. The problem to be solved by the codeword estimator now becomes the choice of the vector E(D) from a set of possible error vectors {E(D)} satisfying Equation A6.16. Equation A6.17 can then be used to produce the vector of polynomials C'(D), which is the output of the codeword estimator.

$$C'(D) = C''(D) \oplus E(D) \qquad (A6.17)$$

The vector of detected data, Q'(D) is then given by,

$$Q'(D) = C'(D)G^{-1}(D) \qquad (A6.18)$$

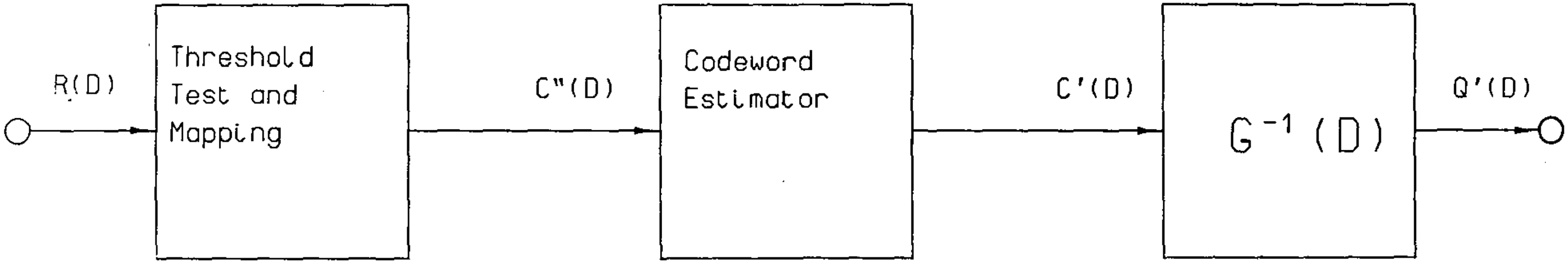


Figure A6.1 Generalised Syndrome Decoding

A7 DISTANCE MEASURES

For all the detectors considered in this investigation, the detector input samples, $\{r_i\}$, are in quantised, or soft-decision, form. The quantisation is assumed to be infinitely fine unless otherwise stated. This soft-decision information is utilised in terms of measures of the distance between the received noisy sequence of complex samples, and a number of possible received sequences (in the absence of noise). These possible received sequences in the absence of noise are determined within the detector as follows. At time $t=jT$ the detector generates possible values of the j-component vector of complex numbers $P_j=[p_1,p_2,\ldots,p_j]$ at the transmitter (see Section 2.1). Such a vector generated in the detector is termed $P'_j=[p'_1,p'_2,\ldots,p'_j]$. The detector uses its knowledge of the channel impulse response Y, (see Section 2.1), to generate the j-component vector of possible received complex samples $R'_j=[r'_1,r'_2,\ldots,r'_j]$, (in the absence of noise). Each component r'_i, of R'_j is determined from Equation A7.1.

$$r'_i = \sum_{h=0}^{g} p'_{i-h}y_h \ , \quad \text{for } i=1,2,\ldots,j \qquad \text{(A7.1)}$$

The optimum distance measure, (see Appendix A3), is the squared unitary distance, d_E^2, between R'_j and the j-component vector of received noisy samples, $R_j=[r_1,r_2,\ldots,r_j]$.

$$d_E^2 = |R_j-R'_j|^2$$

$$= \sum_{i=1}^{j}[\mathrm{Re}(r_i-r'_i)]^2 + \sum_{i=1}^{j}[\mathrm{Im}(r_i-r'_i)]^2 \qquad \text{(A7.2)}$$

Re(x) and Im(x) are, respectively, the real and imaginary parts of the complex value x.

Since the unitary distance measure inherently involves squaring operations, complexity problems are caused at the receiver, because such operations require excessive computation. Even if a look-up table implementation of distance measurement is used, accuracy problems exist because of the increase in dynamic range of the possible squared distances $\{d_E^2\}$ compared with the distances $\{d_E\}$ This leads to a requirement to represent distances with longer (binary) words than may be considered appropriate. That is, finer quantisation may be required. In order to alleviate these problems, various less complex, but suboptimal distance measures are considered in some of the previously described models. The basis for most of these proposed measures is that no squaring operations are involved. The ideas are based on Reference (68). In addition a completely new distance measure is introduced which is possibly of especial relevance to constant envelope-type schemes, where the definition of such schemes is extended to include schemes which are not truely constant envelope, but where every point p_i lies on a circle in the complex number plane, (see Figure 2.5.4 for example).

The first measure to be considered is termed the Magnitude/Sum distance measure.[68] Equation A7.3 defines this distance measure

$$\begin{aligned} d_M &= ||R_j - R_j'|| \\ &= \sum_{i=1}^{j} |\text{Re}(r_i - r_i')| + \sum_{i=1}^{j} |\text{Im}(r_i - r_i')| \qquad \text{(A7.3)} \end{aligned}$$

where $|y|$ is the unsigned value of y, and $||.||$ denotes the Magnitude/Sum distance measure. Clearly no multiplications are required, considerably reducing the complexity of the distance calculations.

The remaining distance measure is of especial relevance for constant envelope-type schemes. For the purposes of this description, it is useful to redefine the received sample r_i at time t=iT in polar coordinates, as in Equation A7.4.

$$r_i = |r_i| \underline{/\phi(r_i)} \tag{A7.4}$$

$\phi(r_i)$ is the phase angle of r_i with respect to the positive real axis in the complex number plane, (that is, with respect to the phase of the carrier), and $|r_i|$ is the magnitude of r_i as defined in Equation A7.5.

$$|r_i|^2 = [\mathrm{Re}(r_i)]^2 + [\mathrm{Im}(r_i)]^2 \tag{A7.5}$$

In the presence of noise, r_i will lie off the signal envelope, and in addition $\phi(r_i)$ will change by an amount $\phi\Delta_i$. The proposed distance measure ignores the value of $|r_i|^2$ with the argument that, for constant envelope schemes, the most important error that the additive noise induces is the phase change, $\Delta\phi_i$. This distance measure takes $|r_i|^2$ to be equal to $|p_i|^2$ in a limiting operation as shown in Figure A7.1. In order to formulate the distance measure mathematically, the detector's set of possible received signal vectors $\{R_j'\}$ at time t=jT, must also be defined in polar form, as in Equation A7.6.

$$r_i' = |r_i'| \underline{/\phi(r_i')} \tag{A7.6}$$

The phase distance measure is given by Equation A7.7.

$$d_p = \sum_{i=1}^{j} |\phi(r_i) - \phi(r_i')| \tag{A7.7}$$

In this case $|\phi(r_i)-\phi(r_i')|$ is the magnitude of the smaller of the two possible differences between the phase angles of r_i and r_i' in the complex number plane. Clearly this distance measure potentially provides a large saving in complexity. The one proviso is that $\phi(r_i)$ must be available. Clearly an explicit calculation is out of the question since it involves trigonometric functions. (An obvious implementation uses a look-up table addressed by the quantised real and imaginary parts of the complex value r_i.) An ammendment to the above distance measure was attempted in one case, (see Section 3.2), in which the distance measure proposed in Equation A7.8 was used.

$$d_p^2 = |\phi(r_i)-\phi(r_i')|^2 \tag{A7.8}$$

Clearly this includes squaring operations which, for the reasons outlined at the beginning of this section, are undesirable.

In all the detectors to be considered, the distance measures given by Equations A7.2, A7.3, A7.7, and A7.8, are described as costs. The term cost implies that there is a penalty, (in terms of increased error rate in the detector's output symbols), in choosing a value of P_j' where the distance between the corresponding vector R_j' and R_j is large, compared with choosing a value of P_j' where the distance between the corresponding vector R_j' and R_j is small. The larger the cost is for a particular value of P_j', the less likely it is that P_j' is equal to P_j at the transmitter. Also, in all cases, the costs are normalised by subtracting the minimum cost at time t=iT from all costs, in order to prevent overflow. This operation in no way affects the performance of the detectors.

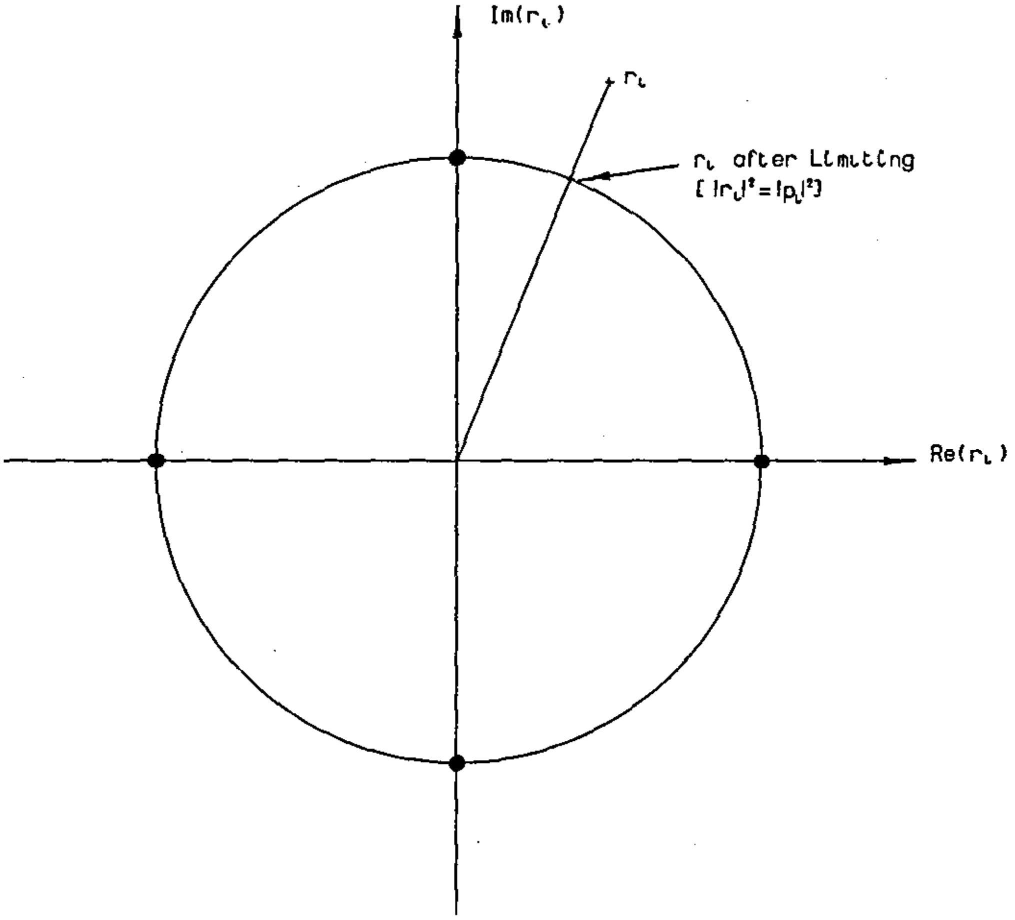


Figure A7.1 The Limiting Effect of The Phase Distance Measure

A8 UNIFIED SYSTEM DESCRIPTION

In order to deal with the many variants of the basic schemes considered in this investigation, a system has been developed which describes these variants in a simple and concise way. The schemes are described in terms of descriptors delimited by slashes. For example, /M=Q/Ch=Il/Det=T/. Within the delimiters, the character on the left hand side of the equals sign is the system attribute to be defined. The character on the right hand side is the actual "value" of this attribute. For example /M=Q/ defines the modulation method (M) as being QPSK (Q). Every attempt has been made to associate system attributes with sensible acronyms so that these can be understood without constant recourse to the table of definitions, (Table A8.1).

The system descriptors appear in the graphs throughout Chapters 2 to 7. The legend which appears in each graph describes only the the system attributes which vary between the curves in the graph. System attributes common to all curves are given in a message entitled "COMMON ATTRIBUTES", unless an attribute in question is a default value for all curves on the graph, in which case it does not appear in the message or the legend. The default values are listed in Table A8.1. In this way, the systems are described in a very concise and understandable way. Table A8.1 lists the definitions, (right hand side of the equals sign) for each system attribute, (left hand side of the equals sign). Note that some system attributes are only valid for certain modulation methods (M). Also it will be noticed that some legend descriptors contradict the common attributes. This is so that, for example, QPSK can be contrasted against a number of 8PSK systems which differ in respect of detection delay, (N), only. In such cases the legend descriptors take precedence.

SYSTEM PARAMETER (LHS of Equation)	ACRONYM	PARAMETER DEFINITION (RHS of Equation)	ACRONYM	NOTES
Modulation Method	M	QPSK 8PSK CORPSK(4-7,1+D)	Q 8 C	
Channel	Ch	Perfect Channel, Bandwidth ±1/2T Hz " " , " ±1/T Hz Raised Cosine Loughborough Filters, narrow bandwidth " " , wide "	I1 I2 RC Mn Mw	(Default)
Data Transmission Rate	Tr	8 M bits/second 4 " 2 " 1 "	8 4 2 1	(Default)
Differential Phase/ Direct Phase Mapping	Ph	Differential phase mapping Direct Map Scheme A " " " B	D Ma Mb	/M=C/ Only
Precoding	Pr	No Precoding Precoding	O D	(Default)
Coding	C	Code 1 " 2 " 3 " 4	1 2 3 4	/M=8/ Only (see Table 2.5.1)

TABLE A8.1: Unified System Description

SYSTEM PARAMETER (LHS of Equation)	ACRONYM	PARAMETER DEFINITION (RHS of Equation)	ACRONYM	NOTES
Detector	Det	Threshold-Level	T	/M=Q/ only
		Viterbi	Va	a = Number of stored vectors
		"	Va,b	" " " b = Number of states in the receiver look-up table model
		Near-Maximum Likelihood, System 1	1Na	a = Number of stored vectors
		" " " , System 3	3Na	" " " "
		Inverse Coder	G	
		Pseudo Nonlinear Equaliser	NLE	
		Soft-Decision Syndrome; definite decoding	Sd	
		" " " ; syndrome resetting	Sf	
Pseudobinary	PB	Standard Technique using Costs	Pb	
		Reduced Complexity	Pbr	/M=8/ Only
		Two-Symbol Expansion	2E	
Distance Measure	Dis	Unitary Distance	E	(Default)
		Phase-Distance	P	
		" " -Squared	P2	
		Magnitude-Sum Distance	MS	
Detection Delay	N	Symbol Intervals	N	$1 \leqslant N \leqslant 80$

TABLE A8.1 (cont.)

SYSTEM PARAMETER (LHS of Equation)	ACRONYM	PARAMETER DEFINITION (RHS of Equation)	ACRONYM	NOTES
Premodulation Filter	Pf	No specific filter 100% Roll-Off Raised Cosine, Length T seconds Nyquist III-ammended 0% Roll-off Raised Cosine	O 1RC N3	/M=C/ Only
Quantisation	Q	Number of bits per axis in the complex number plane	An integer value	Q=inf (infinity) is Default $2\leq Q\leq \text{inf}$
Look-Forward Scheme	LF	Number of symbol intervals	An integer value	/M=8/ Only, $0\leq LF\leq 4$
Vector Retention Scheme	Ret	" " " "	An integer value	/M=8/ Only, $0\leq Ret\leq 11$
Number of Symbols in the Syndrome Sequence	Ls	" " " "	An integer value	/M=8/ Only, $1\leq L_s\leq 10$
Maximum Number of Single Boundary Crosses per Error Vector	Em	Number of Non-Zero Components in the Error Vector	An integer value	/M=8/ Only, $1\leq E_m\leq 5$
State Redefinition	Rec	Redefinition Scheme Pb1 " " Pb3 " " 1a " " 1b " " 3a " " 3b	Pb1 Pb3 1a 1b 3a 3b	Code 1, Pseudobinary Code 3, " Code 1, Non-pseudo- Code 1, " binary Code 3, " Code 3, " /M=8/ only

TABLE A8.1 (cont.)

SYSTEM PARAMETER (LHS of Equation)	ACRONYM	PARAMETER DEFINITION (RHS of Equation)	ACRONYM	NOTES
Noise-Adaptive Viterbi-Type Scheme, Static Expansion Limitation Method	Rexp	1 or 2 Expanded Vectors per vector 2 or 3 " " " " 3 or 4 " " " " 4 " " " "	1 2 3 4	/M=8/ Only
Noise-Adaptive Viterbi-Type Scheme, Maximum Cost	Cm	Maximum Cost	A real positive value	/M=8/ Only $3 \leq C_m \leq 120$
Noise-Adaptive Viterbi-Type Scheme, Dynamic Expansion Limitation Cost Thresholds	cth	Three Cost Thresholds per Equation 6.2.3 (/cth=cth(3),cth(2),cth(1)/)	Real values	e.g. /cth=4.8,3.0,0.0/ /M=8/ Only
Noise-Adaptive Viterbi-Type Scheme, Maximum Number of Vectors	Sv	Maximum number of stored vectors	An integer value	Default is 4^{k-1} where k is the code constant length. $0 \leq Sv \leq 64$

TABLE A8.1 (cont.)

```
C
C
C
C
C
C          QPSK CHANNEL MODEL WITH THRESHOLD DETECTION
C
C
C
      dimension is(-100:0),ffr(0:16)
      COMPLEX st0(-300:100),st3(-300:100),aa,aw,st4(-100:100),
     1ft(0:100),fr(0:100),st2(-300:100),st1(-300:100),
     2w(-300:100),rm(0:3),wf(-300:100)
      DOUBLE PRECISION P,G05DDF,G05DAF
      integer tr3,tr4,q1
      character*3 file1,file2
      open(0,defer=.true.,prompt=.true.)
      write(0,)"Channel Filters"
      read(0,)file1
      write(0,)"Run Parameters"
      read(0,)file2
      open(0,defer=.false.)
      open(1,file=file1,form='formatted',mode='in')
      open(2,file=file2,form='formatted',mode='in')
      read(1,*)tr3,tr4
      read(2,*)IQ,M,L,L1,n,P,pp,ibr,q1
c
c
c Calculate parameters required to read in filters.
c
c
      j3=tr3*q1
      j4=tr4*q1
c
c Set trr for retiming if ibr>1
c
      trr=1+(tr3+tr4)/2
c
c Read in files
c
      do 20 i=0,j3,1
      read(1,*)b1,b2
      ft(i)=cmplx(b1,b2)
   20 continue
      do 30 i=0,j4,1
      read(1,*)b1,b2
      fr(i)=cmplx(b1,b2)
   30 continue
      do 32 i=0,15,1
      read(1,*)ffr(i)
   32 continue
c
c Set Array/Vector limits in time
c
      j1st0=-tr3*q1
      j2st0=q1
      j1st2=-tr4*q1
      j2st2=j2st0
      j1st3=0
      j2st3=j2st0
      lms=0
      if(lms.ne.1)then
      jis=-n+1-(tr3+tr4)/2
      if(ibr.gt.1)jis=jis+1
      else
      jis=-n+1
      endif
c
c Initialise Complex Mapper
c
      rm(0)-(1.0,0.0)
      rm(1)=(0.0,1.0)
      rm(2)-(-1.0,0.0)
      rm(3)=(0.0,-1.0)
c
c SNR LOOP
c
c
      call g05cbf(IQ)
      WRITE(0,600)
      do 3000 lm=1,M,1
      P=P-pp
      ie=0
      ib1=0
      ic=0
      ee=0.0
      ew=0.0
c
c Initialisation of various vectors
c
c is:
c
      do 105 i=jis,0,1
      is(i)=0
  105 continue
c
c Initialise st0,w
c
      do 120 i=j1st0,j2st0,1
      st0(i)=(0.0,0.0)
  120 continue
c
c Set noise vector and st2 to  zero
c
      do 125 i=j1st2,j2st2,1
      w(i)=(0.0,0.0)
      st2(i)=(0.0,0.0)
      st1(i)=(0.0,0.0)
  125 continue
c
c
c Array Initialisation by preamble of data in is,
c
c
      do 190 i=jis,0,1
c
c Left-shift st0,st2
c
      do 140 j=j1st0,(j2st0-q1),1
      jj=j+q1
      st0(j)=st0(jj)
  140 continue
      do 150 j=j1st2,(j2st2-q1),1
      jj=j+q1
      st2(j)=st2(jj)
      st1(j)=st1(jj)
  150 continue
c
c Complex Mapping
c
      st0(j2st0)=rm(is(i))
c
c Initialise st2
c
      do 180 ii=(j2st2-q1+1),j2st2,1
      aa=(0.0,0.0)
      do 175 j=0,j3,1
      aa=aa+ft(j)*st0(ii-j)
  175 continue
      st2(ii)=aa
      st1(ii)=aa
  180 continue
  190 continue
c
c TRANSMISSION LOOP
c
c
      do 1100 lll=1,L1,1
      do 1000 ll=1,L,1
c
c Shift is
c
      do 220 j=jis,-1,1
      jj=j+1
      is(j)=is(jj)
  220 continue
c
c Data Generation
c
      w1=g05daf(-2.0d+00,2.0d+00)
      if(w1.lt.-1.0)then
      is(0)=0
      elseif(w1.ge.-1.0.and.w1.lt.0.0)then
      is(0)=1
      elseif(w1.ge.0.0.and.w1.lt.1.0)then
      is(0)=2
      else
      is(0)=3
      endif
c
c Bit  Rate choice:-
c          (a) ibr=1 ; 8Mb/s
c          (b) ibr=2 ; 4Mb/s
c          (c) ibr=4 ; 2Mb/s
c          (d) ibr=8 ; 1Mb/s
c
      do 340 ib=1,ibr,1
c
c Shift arrays st0,st2,w, one
c symbol interval left
```

```
      do 240 j=j1st0,(j2st0-q1),1
      jj=j+q1
      st0(j)=st0(jj)
  240 continue
      do 250 j=j1st2,(j2st2-q1),1
      jj=j+q1
      st2(j)=st2(jj)
      st1(j)=st1(jj)
      w(j)=w(jj)
  250 continue
c
c Complex Mapping
c
      st0(j2st0)=rm(is(0))
c
c Tx Filtering
c
      do 300 i=(j2st2-q1+1),j2st2,1
      aa=(0.0,0.0)
      do 290 j=0,j3,1
      aa=aa+ft(j)*st0(i-j)
  290 continue
      st2(i)=aa
  300 continue
c
c Rx Filter st2 alone
c
      do 303 i=(j2st3-q1),j2st3,1
      aa=(0.0,0.0)
      do 301 j=0,j4,1
      aa=aa+fr(j)*st2(i-j)
  301 continue
      st4(i)=aa
  303 continue
c
c Noise addition
c
      do 310 i=(j2st2-q1+1),j2st2,1
      w1=g05ddf(0.0d+00,P)
      w2=g05ddf(0.0d+00,P)
      w(i)=cmplx(w1,w2)
      st1(i)=st2(i)+w(i)
  310 continue
c
c Rx Filtering
c
      do 330 i=(j2st3-q1+1),j2st3,1
      aa=(0.0,0.0)
      aw=(0.0,0.0)
      do 320 j=0,j4,1
      aa=aa+fr(j)*st1(i-j)
      aw=aw+fr(j)*w(i-j)
  320 continue
      st3(i)=aa
      wf(i)=aw
  330 continue
c
c Calculate contribution of Rx symbol to total
c signal energy and corresponding contribution
```

```
c to total noise energy
c
      do 341 j=(j2st3-q1+1),j2st3,1
      ee=ee+(real(st2(j))**2+aimag(st2(j))**2)/float(2*L1*L)
      ew=ew+(real(wf(j))**2+aimag(wf(j))**2)/float(q1*L*L1)
  341 continue
c
c Rx signal conditioning for detection
c
      if(ibr.eq.1)then
      rreal=real(st3(j2st3))
      rimag=aimag(st3(j2st3))
      elseif(ibr.eq.2)then
      if(ib.eq.1)then
      rre=real(st3(j2st3))
      rim=aimag(st3(j2st3))
      else
      rreal=rre+real(st3(j2st3))
      rimag=rim+aimag(st3(j2st3))
      endif
      else
      if(ib.ne.trr)then
      rre=rre+real(st3(j2st3))
      rim=rim+aimag(st3(j2st3))
      else
      rreal=rre
      rimag=rim
      rre=real(st3(j2st3))
      rim=aimag(st3(j2st3))
      endif
      endif
  340 continue
c
c Threshold Detection
c
      if(abs(rreal).gt.abs(rimag).and.rreal.gt.0.0)then
      ISS=0
      inn1=0
      inn2=0
      elseif(abs(rreal).gt.abs(rimag).and.rreal.lt.0.0)then
      ISS=2
      inn1=1
      inn2=1
      elseif(abs(rreal).lt.abs(rimag).and.rimag.gt.0.0)then
      ISS=1
      inn1=0
      inn2=1
      else
      ISS=3
      inn1=1
      inn2=0
      endif
c
c
c ERROR COUNT
c
c
      if(is(jis).ne.ISS)then
       if(is(jis).eq.0)then
       in1=0
       in2=0
       elseif(is(jis).eq.1)then
       in1=0
       in2=1
       elseif(is(jis).eq.2)then
       in1=1
       in2=1
       else
       in1=1
       in2=0
       endif
      if(inn1.ne.in1)ie=ie+1
      if(inn2.ne.in2)ie=ie+1
      if(ie.ne.1)goto 500
      ib1=1
      goto 510
  500  if(ic.gt.20)then
       ib1=ib1+1
       else
       endif
  510 continue
      ic=0
      else
      ic=ic+2
      endif
 1000 continue
 1100 continue
C
C THE ERROR RATE,ER, AND THE AVERAGE NUMBER OF ERRORS PER BURST,
C AEPB,ARE NOW CALCULATED. THE SNR IS ALSO CALCULATED AND THE
C RESULTS ARE OUTPUTED.
C
      ER=FLOAT(ie)/(FLOAT(L)*2*FLOAT(L1))
      IF(ib1.EQ.0)GOTO 680
      AEPB=FLOAT(ie)/FLOAT(ib1)
      GOTO 690
  680 AEPB=0
  690 CONTINUE
      ef=0.0
      do 691 i=1,15,1
      ef=ef+ffr(i)**2
  691 continue
      ef=(1.0/16.0)*(ef+ef+ffr(0)**2)
c
c IFFT relationship 'Fiddle Factor'
c
      ek1=(64.0/17.351)**2
c
c RCOS Channel noise variance compensation for
c super-Nyquist sampling and for data rate reduction
c
c     ek1=16.0*ibr
      ee=ee/float(q1)
      EEE=ek1*ef*ee/ew
      SNR=10.0*ALOG10(EEE)
  600 FORMAT(1H ,10X,4H SNR,10X,10HERROR RATE,
     110X,16HERRORS PER BURST)
      WRITE(0,700)SNR,ER,AEPB
  700 FORMAT(1H ,7X,F9.5,6X,E12.5,13X,F9.5)
 3000 continue
```

```
C
C A NUMBER OF IMPORTANT PARAMETERS ARE PRINTED OUT
C
      if(ibr.eq.1)then
      ibrr=8
      elseif(ibr.eq.2)then
      ibrr=4
      elseif(ibr.eq.4)then
      ibrr=2
      elseif(ibr.eq.8)then
      ibrr=1
      else
      ibrr=0
      endif
      write(0,800)IQ,M,L,L1,N,P,pp,ibrr
  800 format(10x,'IQ = ',i2,3x,'M = ',i2,3x,'L = ',i6,3x,'L1 = ',i2,
     13x,'N = ',i2,3x,'P = ',f6.4,3x,'pp = ',f6.4,3x,
     2'Bit Rate = ',i2,'Mb/s'//)
      write(0,810)ee,ew,ef,q1,tr3,tr4,(j2st3),
     1(1-jis)
  810 format(5x,'Energy per bit = ',f10.6,5x,
     1'Expected Noise Power = ',f10.6//
     a5x,'Filter Energy = ',f10.6//
     25X,'No. OF SAMPLES PER SYMBOL INTERVAL = ',I2//
     45x,'SYMBOL LENGTH OF SYMMETRICAL Tx CHANNEL FILTER = ',I2//
     55X,'SYMBOL LENGTH OF SYMMETRICAL Rx CHANNEL FILTER = ',I2//
     65X,'MAIN SAMPLING INSTANT = ',I3,2X,'SAMPLING INTERVALS'//
     85X,'No. OF COMPONENTS IN Tx VECTOR = ',I2////)
      write(0,820)(ft(i),i=0,j3-1),
     1(fr(i),i=0,j4-1)
  820 format('Tx Channel Filter:'/
     22(4(5x,f10.6,3h + ,1hj,f10.6)/)//
     3'Rx Channel Filter:'/2(4(5x,f10.6,3h + ,1hj,f10.6)/)////)
      write(0,830)(is(i),i=jis,0)
  830 format('Tx Source Data:'/2i2//)
      STOP
      END
```

B2 Differential Phase CORPSK(4-7,1+D) Program

```
JOB Z8150N7,:EUXXX.CP76(P0000,TD1280)
FTN5(DB=0/PMD)
LIBRARY(PROCLIB,*)
NAG(FTN5)
LGO.
££££S
C              PROGRAM CORPSK4-7_D2
C
C
C THIS PROGRAM SIMULATES THE TRANSMISSION OF CORRELATIVELY ENCODED
C 4 PHASE (PSK) SYMBOLS OVER AN AWGN CHANNEL WHICH INTRODUCES NO
C DISTORTION (MEMORYLESS CHANNEL). THE VITERBI ALGORITHM IS USED AT
C THE RECEIVER TO PERFORM THE DECODING/DETECTION PROCESS. THIS IS A
C DIFFERENTIAL IMPLEMENTATION OF THE SYSTEM. FOR MORE DETAILS
C SEE THE PROGRAM DOCUMENTATION ENTITLED 'SIMULATION OF
C CORPSK(4-7,1+D) OVER A DISTORTIONLESS CHANNEL'.
C
C DECLARE ALL VARIABLES
C
      PROGRAM COR47D(INPUT,OUTPUT,TAPE1=INPUT,TAPE2=OUTPUT)
      DIMENSION IS(85),IX(32,85),CX(32),CXX(32,4),IXX(32,85),
     1IN(2),INN(2),IZ(32),IZZ(32,4),IVV(2),AR(2),AI(2),RR(2),RI(2),
     2CN(2,8)
      REAL CC,ER,AEPB,W,WI,WR,MAP(4),MAP2(8,2)
      INTEGER IQ,M,L,K,N,IE,IB1,IC,IV
C
C INITIALISE ALL VARIABLES
C
      IQ=30
      M=1
      L=50000
      K=4
      N=2
      P=0.6867
      AII=2*ATAN(1.0)
      MAP(1)=0.0
      MAP(2)=AII
      MAP(3)=AII+AII
      MAP(4)=MAP(3)+AII
      AII=SQRT(2.0)
      MAP2(1,1)=2.0
      MAP2(1,2)=0.0
      MAP2(2,1)=AII
      MAP2(2,2)=AII
      MAP2(3,1)=0.0
      MAP2(3,2)=2.0
      MAP2(4,1)=-AII
      MAP2(4,2)=AII
      MAP2(5,1)=-2.0
      MAP2(5,2)=0.0
      MAP2(6,1)=-AII
      MAP2(6,2)=-AII
      MAP2(7,1)=0.0
      MAP2(7,2)=-2.0
      MAP2(8,1)=AII
      MAP2(8,2)=-AII
      WRITE(2,600)
C
C CALL RANDOM NUMBER GENERATOR BEFORE ALL PROGRAM LOOPS AND
C GENERATE THE NEXT SYMBOL.
```

```
C
      CALL G05CBF(IQ)
      DO 800 LM=1,M,1
      IQM=1
      IZ(1)=7
      IZ(2)=1
      IZ(3)=3
      IZ(4)=5
      P=P-0.00
      IE=0
      IB1=0
      IC=0
      DO 10 I=1,N,1
      IS(I)=1
   10 CONTINUE
      DO 30 I=1,K,1
      NN=N-1
      DO 20 J=1,NN,1
      IX(I,J)=1
   20 CONTINUE
      CX(I)=+1.0E+06
      IX(I,N)=I-1
   30 CONTINUE
      CX(2)=0.0
      DO 671 LLL=1,20,1
      DO 670 LL=1,L,1
      NN=N-1
      DO 40 I=1,NN,1
      JJ=I+1
      IS(I)=IS(JJ)
   40 CONTINUE
      DO 60 I=1,K,1
      NN=N-1
      DO 50 J=1,NN,1
      JJ=J+1
      IX(I,J)=IX(I,JJ)
   50 CONTINUE
   60 CONTINUE
      W=G05DAF(-2.0,2.0)
      IF(W)70,70,100
   70 IF(W+1.0)80,80,90
   80 IS(N)=0
      GOTO 130
   90 IS(N)=1
      GOTO 130
  100 IF(W-1.0)110,110,120
  110 IS(N)=2
      GOTO 130
  120 IS(N)=3
  130 CONTINUE
C
C THE  DATA SYMBOLS ARE CODED : (1+D)  TO PRODUCE
C THE CODE SEQUENCE. THIS IS LEVEL SHIFTED,ADDED TO THE PREVIOUS
C PHASE STORED IN IQM,AND THE RESULT IS MAPPED ONTO ONE OF
C FOUR QPSK SYMBOLS. NOTE THAT THE CODE SEQUENCE CONSISTS OF SEVEN
C LEVELS WHICH  REPRESENT PHASE CHANGES. THE SIGN OF THE CHANGE
C DETERMINES THE DIRECTION AND THEREFORE THE MID-POINT.
C THE QUADRATURE COMPONENTS,AR(I)&AI(I),ARE NOW TRANSMITTED AND ARE
C SUBJECTED TO THE AWGN COMPONENTS,WR &WI,WHICH ARE GENERATED
C USING A RANDOM NUMBER GENERATOR WITH A GAUSSIAN PDF,WITH IT'S
```

```
C STANDARD DEVIATION GIVEN BY P.
C
      IV=IS(N)+IS(NN)-3
      IF(IV)116,133,116
  116 IVV(1)=IQM+IV
      IVV(2)=IQM+IV+IV
      DO 132 I=1,2,1
      IF(IVV(I))117,117,118
  117 IVV(I)=IVV(I)+8
  118 CONTINUE
      IF(IVV(I)-8)131,131,119
  119 IVV(I)=IVV(I)-8
  131 CONTINUE
  132 CONTINUE
      GOTO 134
  133 IVV(1)=IQM
      IVV(2)=IQM
  134 CONTINUE
      IQM=IVV(2)
      DO 135 I=1,2,1
      AR(I)=MAP2(IVV(I),1)
      AI(I)=MAP2(IVV(I),2)
      WR=G05DDF(0.0,P)
      RR(I)=AR(I)+WR
      WI=G05DDF(0.0,P)
      RI(I)=AI(I)+WI
  135 CONTINUE
C
C CALCULATE THE 12 DISTINCT COST HALF-INCREMENTS
C
      DO 142 J=1,7,2
      CN(2,J)=(RR(2)-MAP2(J,1))*(RR(2)-MAP2(J,1))
     1+(RI(2)-MAP2(J,2))*(RI(2)-MAP2(J,2))
  142 CONTINUE
      DO 144 J=1,8,1
      CN(1,J)=(RR(1)-MAP2(J,1))*(RR(1)-MAP2(J,1))
     1+(RI(1)-MAP2(J,2))*(RI(1)-MAP2(J,2))
  144 CONTINUE
C
C MAXIMUM LIKELIHOOD DECODING/DETECTION IS NOW PERFORMED.
C FOR EACH OF THE EXPANSIONS,0,1,2,3,THE IX ARE  CODED &
C MAPPED AND ADDED TO THE PREVIOUS PHASE IZ(I). THE ASSOCIATED
C COSTS ARE FOUND BY ADDING THE APPROPRIATE CN(1, ),
C AND CN(2, ) TO CX(I). VITERBI DECODING/DETECTION IS NOW PERFORMED
C BY PICKING THE BEST VECTOR FOR EACH EXPANSION. THE BEST OF
C THE RESULTING VECTORS IS THE TRUE ML VECTOR AND IT'S
C LEFT-MOST ELEMENT IS THE DETECTED SYMBOL VALUE.
C
      DO 150 I=1,K,1
      DO 140 J=1,4,1
      JJ=J-1
      NN=N-1
      IV=JJ+IX(I,NN)-3
      IF(IV)102,112,102
  102 IVV(1)=IZ(I)+IV
      IVV(2)=IZ(I)+IV+IV
      DO 108 IJ=1,2,1
      IF(IVV(IJ))103,103,104
  103 IVV(IJ)=IVV(IJ)+8
  104 CONTINUE
```

```
      IF(IVV(IJ)-8)106,106,105
  105 IVV(IJ)=IVV(IJ)-8
  106 CONTINUE
  108 CONTINUE
      GOTO 114
  112 IVV(1)=IZ(I)
      IVV(2)=IZ(I)
  114 CONTINUE
      IZZ(I,J)=IVV(2)
      CXX(I,J)=CN(1,IVV(1))+CN(2,IVV(2))+CX(I)
  140 CONTINUE
  150 CONTINUE
      DO 210 J=1,4,1
      CC=10.0E+06
      DO 180 I=1,K,1
      IF(CXX(I,J)-CC)160,170,170
  160 CC=CXX(I,J)
      III=I
  170 CONTINUE
  180 CONTINUE
      IZ(J)=IZZ(III,J)
      NN=N-1
      DO 200 IL=1,NN,1
      IXX(J,IL)=IX(III,IL)
  200 CONTINUE
      IXX(J,N)=J-1
      CX(J)=CC
  210 CONTINUE
      CC=10.0E+06
      DO 240 I=1,K,1
      IF(CX(I)-CC)220,230,230
  220 CC=CX(I)
      III=I
  230 CONTINUE
  240 CONTINUE
      ISS=IXX(III,1)
C
C TRANSFER THE IXX BACK INTO THE IX VECTORS.
C
      CC=CX(III)
      DO 310 I=1,K,1
      DO 300 J=1,N,1
      IX(I,J)=IXX(I,J)
  300 CONTINUE
      CX(I)=CX(I)-CC
  310 CONTINUE
C
C THE NEXT SECTION TESTS FOR ERRORS IN THE DETECTED DIGITS.
C IF A SYMBOL IS FOUND TO BE IN ERROR, BOTH IS(1) & ISS ARE
C CONVERTED TO THEIR BINARY EQUIVALENTS USING THE GRAY CODE.
C THE INDIVIDUAL BITS ARE THEN COMPARED TO COUNT THE ERRORS.
C THE BIT ERROR COUNT,IE,IS INCREMENTED WHENEVER A BIT ERROR
C OCCURS. IF THE NUMBER OF CORRECTLY DETECTED BINARY SYMBOLS
C SINCE THE LAST ERROR IS GREATER OR EQUAL TO 20,THE BURST
C ERROR COUNTER,IB1,IS INCREMENTED ON THE OCCURRENCE OF AN
C ERROR. OTHERWISE,(IF AN ERROR HAS OCCURRED),THE COUNT OF
C CORRECTLY DETECTED SYMBOLS,IC,IS SET TO ZERO. IN ADDITION
C WHEN THE FIRST ERROR OCCURS,IB1 IS SET TO ZERO.
C
      IC=IC+2
      IF(IS(1)-ISS)320,490,320
  320 IF(IS(1)-2)350.330,340
  330 IN(1)=1
      IN(2)-1
      GOTO 380
  340 IN(1)=1
      IN(2)=0
      GOTO 380
  350 IF(IS(1)-1)360,370,370
  360 IN(1)=0
      IN(2)=0
      GOTO 380
  370 IN(1)=0
      IN(2)=1
  380 CONTINUE
      IF(ISS-2)410,390,400
  390 INN(1)=1
      INN(2)=1
      GOTO 440
  400 INN(1)=1
      INN(2)=0
      GOTO 440
  410 IF(ISS-1)420,430,430
  420 INN(1)=0
      INN(2)=0
      GOTO 440
  430 INN(1)=0
      INN(2)=1
  440 CONTINUE
      IF(INN(1).NE.IN(1))IE=IE+1
      IF(INN(2).NE.IN(2))IE=IE+1
      IF(IE.NE.1)GOTO 450
      IB1=1
      GOTO 470
  450 IF(IC-20)480,480,460
  460 IB1=IB1+1
  470 CONTINUE
  480 IC=0
  490 CONTINUE
  670 CONTINUE
  671 CONTINUE
C
C THE ERROR RATE,ER,AND THE AVERAGE NUMBER OF ERRORS PER BURST,
C AEPB,ARE NOW CALCULATED. THE SNR IS ALSO CALCULATED AND
C THE RESULTS ARE SENT TO THE OUTPUT.
C
      ER=(FLOAT(IE))/(FLOAT(L+L))/20.0
      IF(IB1.EQ.0)GOTO 680
      AEPB=(FLOAT(IE))/(FLOAT(IB1))
      GOTO 690
  680 AEPB=0
  690 CONTINUE
      SNR=10.0*ALOG10(2.0/(P*P))
  600 FORMAT(1H ,10X,4H SNR,10X,10HERROR RATE,
     110X,16HERRORS PER BURST)
      WRITE(2,700)SNR,ER,AEPB
  700 FORMAT(1H ,7X,F9.5,7X,E12.5,13X,F9.5)
  800 CONTINUE
C
C A NUMBER OF IMPORTANT PARAMETERS ARE PRINTED OUT.
```

```
C
      WRITE(2,900)((MAP2(I,J),I=1,8),J=1,2),P,IQ,L,K,N
  900 FORMAT(1H ,2(1H ,10X,8F9.5/)/1H ,10X,'P = ',F6.4,5X,
     1'IQ = '.I3,5X,'L = ',I6,5X,'K = ',I2,5X,'N =',I2////)
      WRITE(2,950)((IX(I,J),J=1,N),I=1,K),(CX(I),I=1,K)
  950 FORMAT(1H ,4(1H ,10X,33I1/).4(1H ,10X,F11.5/))
      STOP
      END
££££S
****
```

B3 DIRECT MAP SCHEME B CORPSK(4-7,1+D) PROGRAM

```
JOB Z8150B3,:EUXXX,CP76(P2000,TD256)
FTN5(DB=0/PMD)
LIBRARY(PROCLIB,*)
NAG(FTN5)
LGO.
££££S
C              PROGRAM CORPSK4-7_ND4
C
C
C THIS PROGRAM SIMULATES THE TRANSMISSION OF CORRELATIVELY ENCODED
C 4 PHASE (PSK) SYMBOLS OVER AN AWGN CHANNEL WHICH INTRODUCES NO
C DISTORTION (MEMORYLESS CHANNEL). THE VITERBI ALGORITHM IS USED AT
C THE RECEIVER TO PERFORM THE DECODING/DETECTION PROCESS. THIS IS A
C NON-DIFFERENTIAL IMPLEMENTATION OF THE SYSTEM. FOR MORE DETAILS
C SEE THE PROGRAM DOCUMENTATION ENTITLED 'SIMULATION OF
C CORPSK(4-7,1+D) OVER A DISTORTIONLESS CHANNEL'.
C
C DECLARE ALL VARIABLES
C
      PROGRAM C47ND4(INPUT,OUTPUT,TAPE1=INPUT,TAPE2=OUTPUT)
      DIMENSION IS(85),IX(32,85),CX(32),CXX(32,4),IXX(32,85),
     1IN(2),INN(2),IZ(32),IZZ(32,4),CN(2,8),
     2IMAP(7,7),AR(2),AI(2),RR(2),RI(2),IU(2)
      REAL CC,ER,AEPB,W,WI,WR,MAP(8,2)
      INTEGER IQ,M,L,K,N,IE,IB1,IC,IV
C
C INITIALISE ALL VARIABLES
C
      IQ=83
      M=1
      L=50000
      K=4
      N=33
      P=0.576
C
C DEFINE MAPPING
C
      AII=SQRT(2.0)
      MAP(1,1)=0.0
      MAP(1,2)=2.0
      MAP(2,1)=-2.0
      MAP(2,2)=0.0
      MAP(3,1)=0.0
      MAP(3,2)=-2.0
      MAP(4,1)=2.0
      MAP(4,2)=0.0
      MAP(5,1)=AII
      MAP(5,2)=AII
      MAP(6,1)=-AII
      MAP(6,2)=AII
      MAP(7,1)=-AII
      MAP(7,2)=-AII
      MAP(8,1)=AII
      MAP(8,2)=-AII
      IMAP(1,1)=1
      IMAP(1,2)=6
      IMAP(1,3)=2
      IMAP(1,4)=7
      IMAP(2,1)=6
      IMAP(2,2)=2
      IMAP(2,3)=7
      IMAP(2,4)=3
      IMAP(2,5)=8
      IMAP(3,1)=2
      IMAP(3,2)=7
      IMAP(3,3)=3
      IMAP(3,4)=8
      IMAP(3,5)=4
      IMAP(3,6)=5
      IMAP(4,1)=7
      IMAP(4,2)=3
      IMAP(4,3)=8
      IMAP(4,4)=4
      IMAP(4,5)=5
      IMAP(4,6)=1
      IMAP(4,7)=6
      IMAP(5,2)=8
      IMAP(5,3)=4
      IMAP(5,4)=5
      IMAP(5,5)=1
      IMAP(5,6)=6
      IMAP(5,7)=2
      IMAP(6,3)=5
      IMAP(6,4)=1
      IMAP(6,5)=6
      IMAP(6,6)=2
      IMAP(6,7)=7
      IMAP(7,4)=6
      IMAP(7,5)=2
      IMAP(7,6)=7
      IMAP(7,7)=3
      WRITE(2,600)
C
C CALL RANDOM NUMBER GENERATOR BEFORE ALL PROGRAM LOOPS AND
C GENERATE THE NEXT SYMBOL.
C
      CALL G05CBF(IQ)
      DO 800 LM=1,M,1
      P=P-0.00
      IE=0
      IB1=0
      IC=0
      IQM=3
      IZ(1)=2
      IZ(2)=3
      IZ(3)=4
      IZ(4)=5
      IU(1)=0
      IUX=0
      DO 10 I=1,N,1
      IS(I)=1
   10 CONTINUE
      DO 30 I=1,K,1
      NN=N-1
      DO 20 J=1,NN,1
      IX(I,J)=1
   20 CONTINUE
      IX(I,N)=I-1
      CX(I)=+1.0E+06
   30 CONTINUE
      CX(2)=0.0
      DO 671 LLL=1,10,1
      DO 670 LL=1,L,1
      NN=N-1
      DO 40 I=1,NN,1
      JJ=I+1
      IS(I)=IS(JJ)
   40 CONTINUE
      DO 60 I=1,K,1
      NN=N-1
      DO 50 J=1,NN,1
      JJ=J+1
      IX(I,J)=IX(I,JJ)
   50 CONTINUE
   60 CONTINUE
      W=G05DAF(-2.0,2.0)
      IF(W)70,70,100
   70 IF(W+1.0)80,80,90
   80 IS(N)=0
      GOTO 130
   90 IS(N)=1
      GOTO 130
  100 IF(W-1.0)110,110,120
  110 IS(N)=2
      GOTO 130
  120 IS(N)=3
  130 CONTINUE
C
C PRECODE THE IS(I)
C
      IU(2)=IS(N)-IU(1)
      IF(IU(2).LT.0)IU(2)=IU(2)+4
C
C THE DATA SYMBOLS ARE CODED : (1+D)  TO PRODUCE
C THE CODE SEQUENCE. THIS IS LEVEL SHIFTED AND MAPPED ONTO ONE OF
C FOUR PHASES. THE INITIAL AND FINAL PHASE DESIGNATIONS ARE USED AS
C POINTERS INTO THE IMAP ARRAY TO FIND THE MID-POINT.
C THE QUADRATURE COMPONENTS,AR(I)&AI(I),ARE NOW TRANSMITTED AND ARE
C SUBJECTED TO THE AWGN COMPONENTS,WR &WI,WHICH ARE GENERATED
C USING A RANDOM NUMBER GENERATOR WITH A GAUSSIAN PDF,WITH IT'S
C STANDARD DEVIATION GIVEN BY P.
C
      NN=N-1
      IV=IU(2)+IU(1)+1
      IU(1)=IU(2)
      IVS=IV
      IF(IV-4)117,117,116
  116 IVS=IV-4
  117 CONTINUE
      IVV=IMAP(IQM,IV)
      IQM=IV
      AR(1)=MAP(IVV,1)
      AI(1)=MAP(IVV,2)
      AR(2)=MAP(IVS,1)
      AI(2)=MAP(IVS,2)
      DO 118 I=1,2,1
      WR=G05DDF(0.0,P)
      RR(I)=AR(I)+WR
      WI=G05DDF(0.0,P)
      RI(I)=AI(I)+WI
  118 CONTINUE
C
C CALCULATE THE 12 DISTINCT COST HALF-INCREMENTS
C
      DO 142 J=1,4,1
      CN(2,J)=(RR(2)-MAP(J,1))*(RR(2)-MAP(J,1))
     1+(RI(2)-MAP(J,2))*(RI(2)-MAP(J,2))
  142 CONTINUE
      DO 144 J=1,8,1
      CN(1,J)=(RR(1)-MAP(J,1))*(RR(1)-MAP(J,1))
     1+(RI(1)-MAP(J,2))*(RI(1)-MAP(J,2))
  144 CONTINUE
C
C MAXIMUM LIKELIHOOD DECODING/DETECTION IS NOW PERFORMED.
C FOR EACH OF THE EXPANSIONS,0,1,2,3,THE IX ARE CODED &
C MAPPED. THE ASSOCIATED COSTS ARE FOUND BY ADDING THE
C APPROPRIATE CN(1, ) & CN(2; ) TO CX(I). VITERBI
C DECODING/DETECTION IS NOW PERFORMED BY PICKING THE BEST
C VECTOR FOR EACH EXPANSION 0,1,2,3,4. THE BEST OF THE RESULTING
C VECTORS IS THE TRUE ML VECTOR AND IT'S LEFT-MOST ELEMENT
C IS THE DETECTED SYMBOL VALUE.
C
      DO 150 I=1,K,1
      DO 140 J=1,4,1
      NN=N-1
      IV=J+IX(I,NN)
      IVS=IV
      IF(IV-4)103,103,102
  102 IVS=IV-4
  103 CONTINUE
      IVV=IMAP(IZ(I),IV)
      IZZ(I,J)=IV
      CXX(I,J)=CN(1,IVV)+CN(2,IVS)+CX(I)
C     CXX(I,J)=CN(2,IVS)+CX(I)
  140 CONTINUE
  150 CONTINUE
      DO 210 J=1,4,1
      CC=10.0E+06
      DO 180 I=1,K,1
      IF(CXX(I,J)-CC)160,170,170
  160 CC=CXX(I,J)
      III=I
  170 CONTINUE
  180 CONTINUE
      NN=N-1
      DO 200 IL=1,NN,1
      IXX(J,IL)=IX(III,IL)
  200 CONTINUE
      IXX(J,N)=J-1
      CX(J)=CC
      IZ(J)=IZZ(III,J)
  210 CONTINUE
      CC=10.0E+06
      DO 240 I=1,K,1
      IF(CX(I)-CC)220,230,230
  220 CC=CX(I)
      III=I
  230 CONTINUE
  240 CONTINUE
C
C DECODE THE PRECODED DETECTED VALUE
C
      ISS=IXX(III,1)+IUX
      IUX=IXX(III,1)
      IF(ISS.GE.4)ISS=ISS-4
C
C TRANSFER THE IXX BACK INTO THE IX VECTORS.
C
      CC=CX(III)
      DO 310 I=1,K,1
      DO 300 J=1,N,1
      IX(I,J)=IXX(I,J)
  300 CONTINUE
      CX(I)=CX(I)-CC
  310 CONTINUE
C
C THE NEXT SECTION TESTS FOR ERRORS IN THE DETECTED DIGITS.
C IF A SYMBOL IS FOUND TO BE IN ERROR, BOTH IS(1) & ISS ARE
C CONVERTED TO THEIR BINARY EQUIVALENTS USING THE GRAY CODE.
C THE INDIVIDUAL BITS ARE THEN COMPARED TO COUNT THE ERRORS.
C THE BIT ERROR COUNT,IE,IS INCREMENTED WHENEVER A BIT ERROR
C OCCURS. IF THE NUMBER OF CORRECTLY DETECTED BINARY SYMBOLS
C SINCE THE LAST ERROR IS GREATER OR EQUAL TO 20,THE BURST
C ERROR COUNTER,IB1,IS INCREMENTED ON THE OCCURRENCE OF AN
C ERROR. OTHERWISE,(IF AN ERROR HAS OCCURRED),THE COUNT OF
C CORRECTLY DETECTED SYMBOLS,IC,IS SET TO ZERO. IN ADDITION
C WHEN THE FIRST ERROR OCCURS,IB1 IS SET TO ZERO.
C
      IC=IC+2
      IF(IS(1)-ISS)320,490,320
  320 IF(IS(1)+1)330,340,350
  330 IN(1)=1
      IN(2)=1
      GOTO 380
  340 IN(1)=1
      IN(2)=0
      GOTO 380
  350 IF(IS(1)-2)360,370,370
  360 IN(1)=0
      IN(2)=0
      GOTO 380
  370 IN(1)=0
      IN(2)=1
  380 CONTINUE
      IF(ISS+1)390,400,410
  390 INN(1)=1
      INN(2)=1
      GOTO 440
  400 INN(1)=1
      INN(2)=0
      GOTO 440
  410 IF(ISS-2)420,430,430
  420 INN(1)=0
      INN(2)=0
      GOTO 440
  430 INN(1)=0
      INN(2)=1
  440 CONTINUE
```

```
      IF(INN(1).NE.IN(1))IE=IE+1
      IF(INN(2).NE.IN(2))IE=IE+1
      IF(IE.NE.1)GOTO 450
      IB1=1
      GOTO 470
  450 IF(IC-20)480,480,460
  460 IB1=IB1+1
  470 CONTINUE
  480 IC=0
  490 CONTINUE
  670 CONTINUE
  671 CONTINUE
C
C THE ERROR RATE,ER,AND THE AVERAGE NUMBER OF ERRORS PER BURST,
C AEPB,ARE NOW CALCULATED. THE SNR IS ALSO CALCULATED AND
C THE RESULTS ARE SENT TO THE OUTPUT.
C
      ER=(FLOAT(IE))/(FLOAT(L+L))/10.0
      IF(IB1.EQ.0)GOTO 680
      AEPB=(FLOAT(IE))/(FLOAT(IB1))
      GOTO 690
  680 AEPB=0
  690 CONTINUE
      EE=2.0/(P*P)
      SNR=10.0*ALOG10(EE)
  600 FORMAT(1H ,10X,4H SNR,10X,10HERROR RATE,
     110X,16HERRORS PER BURST)
      WRITE(2,700)SNR,ER,AEPB
  700 FORMAT(1H ,7X,F9.5,7X,E12.5,13X,F9.5)
  800 CONTINUE
C
C A NUMBER OF IMPORTANT PARAMETERS ARE PRINTED OUT.
C
      WRITE(2,900)((MAP(I,J),I=1,8),J=1,2),P,IQ,L,K,N
  900 FORMAT(2(1H ,10X,8F9.5/)/1H ,10X,'P = ',F6.4,5X,
     1'IQ = ',I3,5X,'L = ',I6,5X,'K = ',I2,5X,'N =',I2////)
      WRITE(2,950)((IX(I,J),J=1,N),I=1,K),(CX(I),I=1,K)
  950 FORMAT(4(1H ,10X,33I1/),4(1H ,10X,F11.5/))
      STOP
      END
££££S
****
```

```
C
C
C
C
C       DIFFERENTIAL CORPSK4-7,1+D WITH I/Q DEMODULATION
C
C
C
C
      dimension izz(0:63,0:3),hd(-60:76),thr(0:255),is(-100:0),
     3ix(0:63,65),cx(0:63),cxx(0:63,0:3),iz(0:63),
     4icheck(0:63),ifull(0:63),sph(-100:100),ffr(0:15)
      COMPLEX st0(-300:100),st3(-300:100),aa,aw,st4(-100:100),
     1ft(0:100),fr(0:100),st2(-300:100),fmap(0:255),st1(-300:100),
     2C00(0:255,0:3),C01(0:255,0:3),w(-300:100),wf(-300:100)
      DOUBLE PRECISION P,G05DDF,G05DAF,a1,a2,pi
      integer g,gg,gg1,gg2,gg3,e(0:63),ett(0:63),et(0:63,0:3),
     1tr1,tr2,tr3,tr4,sa,q1,q,ics(0:255,0:3),sa1,sa2,sa3
      character*3 file1,file2,file3,file5
      open(0,defer=.true.,prompt=.true.)
      write(0,)"Run-dependent Parameters File"
      read(0,)file1
      write(0,)"Premod. Filter Parameters File"
      read(0,)file2
      write(0,)"Premodulation Filter File"
      read(0,)file3
      write(0,)"Rx State Arrays and Minimum-phase Channel Filters"
      read(0,)file5
      open(0,defer=.false.)
      open(1,file=file1,form='formatted',mode='in')
      open(2,file=file2,form='formatted',mode='in')
      open(3,file=file3,form='formatted',mode='in')
      open(5,file=file5,form='formatted',mode='in')
      read(1,*)IQ,M,L,L1,N,P,pp,g,nb
      read(2,*)q,q1,tr1,tr2
      read(5,*)tr3,tr4
c
c
c Calculate parameters required to read in filters.
c sa: No. states in Rx array model
c
c
      j1=-tr1*q1
      j2=(tr2+2)*q1
      j3=tr3*q1
      j4=tr4*q1
      nn=tr1+tr2
      read(2,*)ifft,ishift,isa
      if(isa.eq.0)then
      sa=4**(nn+1)
      nnn=nn
      else
      sa=4**isa
      nnn=isa-1
      endif
c
c Read in files
c
      do 10 i=j1,j2,1
      read(3,*)hd(i)
```

```
   10 continue
      do 20 i=0,j3,1
      read(5,*)b1,b2
      ft(i)=cmplx(b1,b2)
   20 continue
      do 30 i=0,j4,1
      read(5,*)b1,b2
      fr(i)=cmplx(b1,b2)
   30 continue
      do 50 i=0,(sa-1),1
      do 40 j=0,3,1
      read(5,*)b1,b2
      co0(i,j)=cmplx(b1,b2)
   40 continue
   50 continue
      do 70 i=0,(sa-1),1
      do 60 j=0,3,1
      read(5,*)b1,b2
      co1(i,j)=cmplx(b1,b2)
   60 continue
   70 continue
      do 90 i=0,(sa-1),1
      do 80 j=0,3,1
      read(5,*)ics(i,j)
   80 continue
   90 continue
      do 95 i=0,15,1
      read(5,*)ffr(i)
   95 continue
c
c
c Phase Quantiser Initialisation
c
c
c             nb: No. of quantiser bits
c             jx: No. of levels/thresholds
c          xincr: Spacing between thresholds (angle)
c         thr( ): Thresholds spaced xincr apart
c        fmap( ): Levels spaced xincr apart-complex array
c
c
c Initialise mapping of phase (sph) onto quadrature components (st0)
c Initialise thresholds
c
      jx=2**nb-1
      xincr=2.0/float(jx+1)
      pi=dacos(-1.0d+00)
      do 100 i=0,jx,1
      a1=dcos(pi*float(i)*xincr)
      a2=dsin(pi*float(i)*xincr)
      fmap(i)=cmplx(a1,a2)
      thr(i)=float(i)*xincr+(xincr/2.0)
  100 continue
      ahd=4.0*hd((tr2+1)*q1)
c
c
c SNR LOOP
c
c
      call g05cbf(IQ)
      WRITE(0,600)
      do 3000 lm=1,M,1
      P=P-pp
      ie=0
      ib1=0
      ic=0
      ee=0.0
      ew=0.0
c
c Initialisation of various vectors
c
c is:
c
      isd=0
      issd=0
c
c Ammend jis if channel is symmetrical (ilmc not equal to 1)
c
      ilmc=0
      if(ilmc.ne.1)then
      jis=-(N+tr1)+1-(tr3+tr4)/2
      else
      jis=-(N+tr1)+1
      endif
      do 105 i=jis,0,1
      is(i)=0
 105  continue
      gg=(4**g)-1
c
c Initialise sph,st0,w
c
      do 110 i=j1,j2,1
      sph(i)=hd(q1)
 110  continue
      j1st0=-(tr1+tr3)*q1
      j2st0=-(tr1-1)*q1
      j1st2=-(tr1+tr4)*q1
      j2st2=j2st0
      j1st3=-(tr1+1)*q1
      j2st3=j2st0
      do 120 i=j1st0,j2st0,1
      st0(i)=(0.0,0.0)
 120  continue
c
c Set noise vector and st2 to  zero
c
      do 125 i=j1st2,j2st2,1
      w(i)=(0.0,0.0)
      st2(i)=(0.0,0.0)
      st1(i)=(0.0,0.0)
 125  continue
c
      do 190 i=jis,0,1
c
c Left-shift sph,st0,st2
c
      do 130 j=j1,(j2-q1),1
      jj=j+q1
      sph(j)=sph(jj)
 130  continue
```

```
      do 140 j=j1st0,(j2st0-q1),1
      jj=j+q1
      st0(j)=st0(jj)
  140 continue
      do 150 j=j1st2,(j2st2-q1),1
      jj=j+q1
      st2(j)=st2(jj)
      st1(j)=st1(jj)
  150 continue
c
c Filter data (=0) through premodulation
c filter.
c
      do 160 j=j1,j2,1
      sph(j)=sph(j)-3.0*hd(j)
      if(sph(j).gt.2.0)sph(j)=sph(j)-ahd
      if(sph(j).lt.0.0)sph(j)=sph(j)+ahd
  160 continue
c
c Calculate st0
c
      do 170 ii=(j2st0-q1+1),j2st0,1
c
c Phase quantisation & mapping
c
      ij=jx-1
      do 165 j=0,ij,1
      jj=j+1
      if(sph(ii).ge.thr(j).and.sph(ii).lt.thr(jj))then
      st0(ii)=fmap(jj)
      j=ij+1
      else
      continue
      endif
  165 continue
      if(sph(ii).ge.thr(jx).or.sph(ii).lt.thr(0))st0(ii)=fmap(0)
  170 continue
c
c Initialise st2
c
      do 180 ii=(j2st2-q1+1),j2st2,1
      aa=(0.0,0.0)
      do 175 j=0,j3,1
      aa=aa+ft(j)*st0(ii-j)
  175 continue
      st2(ii)=aa
      st1(ii)=aa
  180 continue
  190 continue
c
c Initialisation of states of Rx vectors
c
c
c Determine Initial Phase
c
      if(ilmc.ne.1)then
      il1=j2st0-q1*(tr3+tr4)/2
      else
      il1=j2st0
      endif
      pre=real(st0(ill))
      pim=aimag(st0(ill))
      if(abs(pre).gt.abs(pim).and.pre.gt.0.0)then
      iz1=0
      elseif(abs(pre).gt.abs(pim).and.pre.lt.0.0)then
      iz1=2
      elseif(abs(pre).lt.abs(pim).and.pim.gt.0.0)then
      iz1=1
      else
      iz1=3
      endif
      if(sa.ge.4*(gg+1))then
      do 210 i=0,gg,1
      iz(i)=i+iz1*(4**nnn)
  210 continue
      else
      sa1=4**nnn
      sa2=2*sa1
      sa3=3*sa1
      do 200 i=0,gg,1
       if(i.lt.sa1)then
       is1=0
       elseif(i.ge.sa1.and.i.lt.sa2)then
       is1=sa1*4
       elseif(i.ge.sa2.and.i.lt.sa3)then
       is1=sa2*4
       else
       is1=sa3*4
       endif
      iz(i)=i+(4**nnn)*(iz1-is1)
  200 continue
      endif
C
C******************$$$$$$$$$$$$*******************
C
C DETECTOR INITIALISATION
C
      ivec=N-g
      ivec2=ivec-1
      gg1=(gg+1)/4
      gg2=(gg+1)/2
      gg3=(gg+1)*3/4
      do 21 i=0,gg,1
      do 11 j=1,ivec,1
      ix(i,j)=0
   11 continue
      cx(i)=1.0e+06
      e(i)=i
      ett(i)=i
   21 continue
      cx(0)=0.0
C
C*******************$$$$$$$$$$$$**************
C
c
c TRANSMISSION LOOP
c
c
      do 1100 lll=1,L1,1
      do 1000 ll=1,L,1
```

```
c
c Shift arrays is,sph,st0,st2,w, one
c symbol interval left
c
      do 220 j=jis,-1,1
      jj=j+1
      is(j)=is(jj)
  220 continue
      do 230 j=j1,(j2-q1),1
      jj=j+q1
      sph(j)=sph(jj)
  230 continue
      do 240 j=j1st0,(j2st0-q1),1
      jj=j+q1
      st0(j)=st0(jj)
  240 continue
      do 250 j=j1st2,(j2st2-q1),1
      jj=j+q1
      st2(j)=st2(jj)
      st1(j)=st1(jj)
      w(j)=w(jj)
  250 continue
C
C
C*****************$$$$$$$$$**************
C
C SHIFT THE IX LEFT
C
      ifull1=0
      do 41 i=0,gg,1
      icheck(i)=0
   41 continue
      do 61 i=0,gg,1
      if(icheck(ett(i)))44,44,51
   44 icheck(ett(i))=1
      do 46 j=1,ivec2,1
      jj=j+1
      ix(ett(i),j)=ix(ett(i),jj)
   46 continue
      if(i.lt.gg1)then
      ix(ett(i),ivec)=0
      elseif(i.ge.gg1.and.i.lt.gg2)then
      ix(ett(i),ivec)=1
      elseif(i.ge.gg2.and.i.lt.gg3)then
      ix(ett(i),ivec)=2
      else
      ix(ett(i),ivec)=3
      endif
      e(i)=ett(i)
      goto 59
   51 do 58 j=ifull1,gg,1
      if(ifull(j))54,54,57
   54 do 55 jj=1,ivec2,1
      ix(j,jj)=ix(ett(i),jj)
   55 continue
      if(i.lt.gg1)then
      ix(j,ivec)=0
      elseif(i.ge.gg1.and.i.lt.gg2)then
      ix(j,ivec)=1
      elseif(i.ge.gg2.and.i.lt.gg3)then
      ix(j,ivec)=2
      else
      ix(j,ivec)=3
      endif
      e(i)=j
      ifull1=j+1
      j=gg+1
   57 continue
   58 continue
   59 continue
   61 continue
C
C***************$$$$$$$$$$$$$$$$$****************
C
c
c Data Generation
c
      w1=g05daf(-2.0d+00,2.0d+00)
      if(w1.lt.-1.0)then
      is(0)=0
      elseif(w1.ge.-1.0.and.w1.lt.0.0)then
      is(0)=1
      elseif(w1.ge.0.0.and.w1.lt.1.0)then
      is(0)=2
      else
      is(0)=3
      endif
c
c Precoding
c
      isd=is(0)-isd
      if(isd.lt.0)isd=isd+4
c
c Premodulation Filtering
c
      ss=2*(float(isd)-1.5)
      do 260 j=j1,j2,1
      sph(j)=sph(j)+ss*hd(j)
      if(sph(j).gt.2.0)sph(j)=sph(j)-ahd
      if(sph(j).lt.0.0)sph(j)=sph(j)+ahd
  260 continue
c
c Convert sph into st0
c
      do 280 i=(j2st0-q1+1),j2st0,1
c
c Phase quantisation & mapping
c
      ij=jx-1
      do 270 j=0,ij,1
      jj=j+1
      if(sph(i).ge.thr(j).and.sph(i).lt.thr(jj))then
      st0(i)=fmap(jj)
      j=ij+1
      else
      continue
      endif
  270 continue
      if(sph(i).ge.thr(jx).or.sph(i).lt.thr(0))st0(i)=fmap(0)
  280 continue
```

```
c
c Tx Filtering
c
      do 300 i=(j2st2-q1+1),j2st2,1
      aa=(0.0,0.0)
      do 290 j=0,j3,1
      aa=aa+ft(j)*st0(i-j)
  290 continue
      st2(i)=aa
  300 continue
c
c Rx Filter st2 alone
c
      do 303 i=(j2st3-q1),j2st3,1
      aa=(0.0,0.0)
      do 301 j=0,j4,1
      aa=aa+fr(j)*st2(i-j)
  301 continue
      st4(i)=aa
  303 continue
c
c Noise addition
c
      do 310 i=(j2st2-q1+1),j2st2,1
      w1=g05ddf(0.0d+00,P)
      w2=g05ddf(0.0d+00,P)
      w(i)=cmplx(w1,w2)
      st1(i)=st2(i)+w(i)
  310 continue
c
c Rx Filtering
c
      do 330 i=(j2st3-q1+1),j2st3,1
      aa=(0.0,0.0)
      aw=(0.0,0.0)
      do 320 j=0,j4,1
      aa=aa+fr(j)*st1(i-j)
      aw=aw+fr(j)*w(i-j)
  320 continue
      st3(i)=aa
      wf(i)=aw
  330 continue
c
c Calculate contribution of Rx symbol to total
c signal energy and corresponding contribution
c to total noise energy
c
      do 341 j=(j2st3-q1+1),j2st3,1
      ee=ee+(real(st2(j))**2+aimag(st2(j))**2)/float(2*L1*L)
      ew=ew+(real(wf(j))**2+aimag(wf(j))**2)/float(q1*L*L1)
  341 continue
C
C
C*******************$$$$$$$$$$$$$$**************
C
C MAXIMUM LIKELIHOOD DECODING/DETECTION IS NOW PERFORMED.
C FOR EACH OF THE EXPANSIONS 0,1,2,3, THE IX ARE CODED &
C MAPPED. THE ASSOCIATED INCREMENTAL COSTS ARE FOUND BY
C COMPARING THE RECEIVED COMPLEX SIGNALS WITH THE
C APPROPRIATE MID & END POINTS HELD IN THE
```

```
C TABLES COO & CO1. VITERBI
C DECODING/DETECTION IS NOW PERFORMED BY PICKING THE BEST
C VECTOR FOR EACH EXPANSION. THE BEST OF THE RESULTING
C VECTORS IS THE TRUE ML VECTOR AND IT'S LEFT-MOST ELEMENT
C IS THE DETECTED VALUE
C
c Expansion & Cost calculation
c
      do 350 i=0,gg,1
      do 340 j=0,3,1
c
c Virtual Sub-vector left-shift
c
      if(i.ge.gg3)then
      ii=(4*i)-(4*gg3)+j
      et(ii,3)=e(i)
      ij=3
      elseif(i.ge.gg2.and.i.lt.gg3)then
      ii=(4*i)-(4*gg2)+j
      et(ii,2)=e(i)
      ij=2
      elseif(i.ge.gg1.and.i.lt.gg2)then
      ii=(4*i)-(4*gg1)+j
      et(ii,1)=e(i)
      ij=1
      else
      ii=4*i+j
      et(ii,0)=e(i)
      ij=0
      endif
c
c Expansion and mapping to points in the constellation
c
      izz(ii,ij)=ics(iz(i),j)
c
c Real & imag. parts of main received sample
c
      sepr=real(st3(j2st3))
      sepi=aimag(st3(j2st3))
c
c Real & imag. parts of interm. received sample
c
      sipr=real(st3(j2st3-q1/2))
      sipi=aimag(st3(j2st3-q1/2))
c
c Real & imag. parts of possible received interm sample
c
      c0r=real(co0(iz(i),j))
      c0i=aimag(co0(iz(i),j))
c
c Real & imag. parts of possible received main sample
c
      c1r=real(co1(iz(i),j))
      c1i=aimag(co1(iz(i),j))
c
c Cost Calculation
c
      cxx(ii,ij)=cx(i)+(sepr-c1r)*(sepr-c1r)
     1+(sepi-c1i)*(sepi-c1i)+(sipr-c0r)*(sipr-c0r)
     2+(sipi-c0i)*(sipi-c0i)
```

```
  340 continue
  350 continue
c
c Set ifull for all vectors to signify empty
c
      do 375 i=0,gg,1
      ifull(i)=0
  375 continue
c
c Selection
c
      do 410 i=0,gg,1
      cc=10.0e+06
      do 400 j=0,3,1
      if(cxx(i,j)-cc)380,390,390
  380 jj=j
      cc=cxx(i,j)
  390 continue
  400 continue
      ett(i)=et(i,jj)
      cx(i)=cxx(i,jj)
      ifull(ett(i))=1
      iz(i)=izz(i,jj)
  410 continue
c
c Detection
c
      cc=10.0e+06
      do 440 i=0,gg,1
      if(cx(i)-cc)420,430,430
  420 ii=i
      cc=cx(i)
  430 continue
  440 continue
      ISS=ix((ett(ii)),1)+issd
      if(ISS.gt.3)ISS=ISS-4
      issd=ix(ett(ii),1)
C
c Subtract lowest cost from all costs
C
      cc=cx(ii)
      DO 311 i=0,gg,1
      cx(i)=cx(i)-cc
  311 CONTINUE
c
c
c ERROR COUNT
c
c
      if(is(jis).ne.ISS)then
       if(is(jis).eq.0)then
       in1=0
       in2=0
       elseif(is(jis).eq.1)then
       in1=0
       in2=1
       elseif(is(jis).eq.2)then
       in1=1
       in2=1
       else
       in1=1
       in2=0
       endif
       if(ISS.eq.0)then
       inn1=0
       inn2=0
       elseif(ISS.eq.1)then
       inn1=0
       inn2=1
       elseif(ISS.eq.2)then
       inn1=1
       inn2=1
       else
       inn1=1
       inn2=0
       endif
      if(inn1.ne.in1)ie=ie+1
      if(inn2.ne.in2)ie=ie+1
      if(ie.ne.1)goto 500
      ib1=1
      goto 510
  500  if(ic.gt.20)then
       ib1=ib1+1
       else
       continue
       endif
  510 continue
      ic=0
      else
      ic=ic+2
      endif
 1000 continue
 1100 continue
C
C THE ERROR RATE,ER, AND THE AVERAGE NUMBER OF ERRORS PER BURST,
C AEPB,ARE NOW CALCULATED. THE SNR IS ALSO CALCULATED AND THE
C RESULTS ARE OUTPUTED.
C
      ER=FLOAT(ie)/(FLOAT(L)*2*FLOAT(L1))
      IF(ib1.EQ.0)GOTO 680
      AEPB=FLOAT(ie)/FLOAT(ib1)
      GOTO 690
  680 AEPB=0
  690 CONTINUE
      ef=0.0
      do 691 i=1,15,1
      ef=ef+ffr(i)**2
  691 continue
      ef=(2.0/16.0)*(ef+ef+ffr(0)**2)
c
c IFFT relationship 'Fiddle Factor'
c
      ek1=(64.0/17.351)**2
      ee=ee/float(q1)
      EEE=ek1*ef*ee/ew
      SNR=10.0*ALOG10(EEE)
  600 FORMAT(1H ,10X,4H SNR,10X,10HERROR RATE,
     110X,16HERRORS PER BURST)
      WRITE(0,700)SNR,ER,AEPB
  700 FORMAT(1H ,7X,F9.5,6X,E12.5,13X,F9.5)
 3000 continue
C
C A NUMBER OF IMPORTANT PARAMETERS ARE PRINTED OUT
C
  799 format('af='f10.5)
      write(0,800)IQ,M,L,L1,N,P,pp,g,(gg+1),sa
  800 format(10x,'IQ = ',i2,3x,'M = ',i2,3x,'L = ',i6,3x,'L1 = ',i2,
     13x,'N = ',i2,3x,'P = ',f6.4,3x,'pp = ',f6.4,3x,'g = ',i2,
     2/'No. states in Viterbi Model = ',i2/
     3'No. states assumed in Rx array model = ',i2////)
      write(0,810)ee,ew,q1,tr1,tr2,tr3,tr4,(j2st3),(j2st3-q1/2),
     1(1-jis)
  810 format(5x,'Energy per bit = ',f10.6,5x,
     1'Expected Noise Power = ',f10.6//
     25X,'No. OF SAMPLES PER SYMBOL INTERVAL = ',I2//
     35X,'SYMBOL LENGTH OF "FREQ. PULSE" FILTER = -',I2,
     a' TO +',i2,' INTERVALS'//
     45x,'SYMBOL LENGTH OF SYMMETRICAL Tx CHANNEL FILTER = ',I2//
     55X,'SYMBOL LENGTH OF SYMMETRICAL Rx CHANNEL FILTER = ',I2//
     65X,'MAIN SAMPLING INSTANT = ',I3,2X,'SAMPLING INTERVALS'//
     75X,'INTERM. SAMPLING INSTANT = ',I3,2X,'SAMPLING INTERVALS'//
     85X,'No. OF COMPONENTS IN Tx VECTOR = ',I2////)
      write(0,820)(hd(i),i=j1,j2-1),(ft(i),i=0,j3-1),
     1(fr(i),i=0,j4-1)
  820 format('Phase Response Filter:'/
     112(8(5x,f10.6)/)//
     2'Tx Channel Filter:'/8(4(5x,f10.6,3h + ,1hj,f10.6)/)//
     3'Rx Channel Filter:'/8(4(5x,f10.6,3h + ,1hj,f10.6)/)////)
      write(0,830)(is(i),i=jis,0),isd,((ix(i,j),j=1,ivec),i=0,gg),
     1issd,(cx(i),i=0,gg)
  830 format('Tx Source Data:'/40i2/
     a'Tx Precoded Value:',i2//
     1'Rx Vectors:'/16(5x,31i2/)/
     b'Rx Previously Decoded value:',i2//
     2'Rx Vector Costs:'/16(5x,f11.5/)////)
      write(0,840)(e(i),i=0,gg),(icheck(i),i=0,gg),
     1(ifull(i),i=0,gg),(iz(i),i=0,gg)
  840 format('Sub-vector designations:',16i3//
     1'Rx vector availability condition flags;',16i2//
     a'Rx vector full/empty condition flags;',16i2//
     2'Rx vector state designations:',16i3////)
      write(0,850)((co0(i,j),j=0,3),i=0,(sa-1)),
     1((co1(i,j),j=0,3),i=0,(sa-1)),((ics(i,j),j=0,3),i=0,(sa-1))
  850 format('Rx Array of possible received interm. samples; co0:'/
     164(4(5x,f10.6,3h + ,1hj,f10.6)/)//
     2'Rx Array of possible received main samples; co1:'/
     364(4(5x,f10.6,3h + ,1hj,f10.6)/)//
     4'Rx Array of possible final states; ics:'/
     516(16(2x,i3)/)////)
      write(0,860)nb,xincr,(thr(i),i=0,jx),(fmap(i),i=0,jx)
  860 format('No. of Quantiser bits:',i2/
     1'Quantiser Spacing:',f10.6/
     2'Array of Thresholds:'/32(8(5x,f10.6)/)//
     3'Array of complex levels:'/64(4(f10.6,3h + ,1hj,f10.6)/)////)
      STOP
      END
```

B5 PHASE RESPONSE GENERATION PROGRAM

```
c
c         Program premod1.fortran
c
c
c
c
c This program assertains the phase response filter for
c Raised Cosine filters of various lengths and
c roll-off factors.
c The parameters concerning sampling rate for the
c premodulation and phase response filters and the
c premodulation filter length are loaded from an
c input file at the start of the program. The algorithm
c consists of
c
c         (a) Premod. filter calculation
c         (b) [1+d] correlation
c         (c) Trapezoidal integration to yield the
c             phase response.
c
c
      integer q,q1,tr1,tr2
      double precision hh(-3000:3000),ha(-3000:3300),hd(-500:500),
     1dx,pi,aa
      real hf(-256:256)
      complex a(16390),w,u,t
      character*3 filein,fileout
      open(0,defer=.true.,prompt=.true.)
      write(0,)"Input data filename"
      read(0,)filein
      write(0,)"Output filename"
      read(0,)fileout
      open(0,defer=.false.)
      open(1,file=filein,form='formatted',mode='in')
      open(2,file=fileout,form='formatted',mode='out')
      read(1,*)q,q1,tr1,tr2
      read(1,*)ifft,ishift
c
c
c Define  premod. filter
c
      pi=dacos(-1.0d+00)
      j1=-(tr1*q)
      j2=q*(tr2)
      j3=j2+q
      j4=-(tr1*q1)
      j5=q1*(tr2+2)
c
c
      if(ifft.ne.1)goto 21
c IFFT PROCEDURE
c
c
      alpha=0.0
      if(q.eq.16)m=9
      if(q.eq.32)m=11
      if(q.eq.64)m=13
      n=2**m
c
c Freq. response definition
```

```
c
      is=nint(q*(1.0-alpha))+1
      do 2 i=1,is,1
      a(i)=(1.0,0.0)
      hf(i-1)=1.0
      hf(1-i)=1.0
    2 continue
      isx=nint(q*(1.0+alpha))+1
      aa=(1-alpha)/2.0
      dx=0.5/float(q)
      do 3 i=is+1,isx,1
      aa=aa+dx
c     a1=0.5*(1.0-sin(pi*(aa-0.5)/alpha))
      a1=(1.0,0.0)
      a(i)=cmplx(a1,0.0)
      hf(i-1)=a1
      hf(1-i)=a1
    3 continue
      isx=65
      a(1)=(1.0,0.0)
      aa=0.0
      do 201 i=2,isx,1
      aa=aa+dx
      a1=pi*aa/sin(pi*aa)
      hf(i-1)=a1
      hf(1-i)=a1
      a(i)=cmplx(a1,0.0)
  201 continue
      nv2=n/2
      do 4 i=isx+1,nv2,1
      a(i)=(0.0,0.0)
    4 continue
      j=2
      do 5 i=n,nv2,-1
      a(i)=a(j)
      j=j+1
    5 continue
c
c IFFT
c
      nm1=n-1
      j=1
      do 8 i=1,nm1,1
      if(i.ge.j)goto 6
      t=a(j)
      a(j)=a(i)
      a(i)=t
    6 k=nv2
    7 if(k.ge.j)goto 8
      j=j-k
      k=k/2
      goto 7
    8 j=j+k
      do 20 l=1,m,1
      le=2**l
      le1=le/2
      u=(1.0,0.0)
      w=cmplx(cos(pi/le1),sin(pi/le1))
      do 20 j=1,le1,1
      do 10 i=j,n,le
      ip=i+le1
      t=a(ip)*u
      a(ip)=a(i)-t
   10 a(i)=a(i)+t
   20 u=u*w
      dx=1.0d+00/(float(n)*dx)
   21 continue
c
      if(ifft.eq.1)then
c
c
c Transfer a(i) to hh(i+1) and normalise
c
      do 31 i=0,j2,1
      hh(i)=real(a(i+1))/real(a(1))
      hh(-i)=hh(i)
   31 continue
c
c Else define hh(t)
c
      else
      do 22 i=-400,j1-1,1
      hh(i)=0.0
   22 continue
      dx=1.0d+00/float(q)
      aa=-dx
      xl=float(tr2+tr1)
      do 24 i=j1,j2,1
      aa=aa+dx
      hh(i)=(0.5d+00/xl)*(1.0d+00-dcos(2.0d+00*pi*aa/xl))
   24 continue
      do 23 i=j2+1,400,1
      hh(i)=0.0
   23 continue
      endif
c
c [1+D] correlation
c
      do 30 i=j1,j3,1
      ha(i)=hh(i)+hh(i-q)
   30 continue
c
c
c Trapezoidal integration
c
      aa=0.0
      if(ifft.eq.1)then
      ib=j1+q
      jst=j4+q1
      jfi=j5-q1
      else
      ib=j1
      jst=j4
      jfi=j5
      endif
      hd(jst)=0.0
      qq=q/q1
      do 50 i=jst+1,jfi,1
      do 40 j=1,qq,1
      ib=ib+1
      aa=aa+ha(ib)*dx
   40 continue
      hd(i)=aa
   50 continue
c
c
c Test ishift & ifft in case T/2 shift is required
c
      if(ishift.eq.1.and.ifft.eq.1)then
      do 57 i=j5,j4+q1/2,-1
      ii=i-q1/2
      hd(i)=hd(ii)
   57 continue
      do 58 i=j4,j4+q1/2+1,1
      hd(i)=0.0
   58 continue
      else
      continue
      endif
c
c
c Scale hd(i) so that hd([tr2+1]T)=1/2
c
      if(ifft.eq.1)then
      do 52 i=jfi+1,j5,1
      hd(i)=hd(jfi)
   52 continue
      else
      continue
      endif
      aa=hd((tr2+1)*q1)/0.5d+00
      do 55 i=j4,j5,1
      hd(i)=hd(i)/aa
   55 continue
c
c
c Output
c          (a) hh(i),ha(i),hd(i) to fileout for
c              graph production and corpsk4-7_d3 use
c          (b) parameters & hd(i) to .absout for print-off
c
      write(2,60)(hd(i),i=j4,j5),(hh(i),i=j1,j2),(ha(i),i=j1,j3)
   60 format(f25.20)
      if(ifft.eq.1)write(0,63)is,isx,(hf(i),i=-127,128)
   63 format('is = ',i3,5x,'isx = ',i4/'hf = '/
     132(8(f10.6,3x)/)//)
      write(0,70)q,q1,tr1,tr2,(hh(i),i=j1,j2-1),(ha(i),i=j1,j3-1),
     1(hd(i),i=j4,j5-1)
   70 format('No. samples per T for Premod. filter:',i5/
     1'No. samples per T for phase response:',i5/
     2'Length of Premod. filter:  -',i3,' to +',i3,' symbols'//
     3'Premod. Filter Characteristic:'/
     480(8(5x,f10.6)/)//
     5'[1+d] adjusted filter:'/
     588(8(5x,f10.6)/)//
     7'Phase Response Filter Characteristic:'/
     812(8(5x,f10.6)/)//)
      stop
      end
```

B6 CORPSK(4-7,1+D) LOOK-UP TABLE GENERATION

```
c             Program corlup2
c
c
c
c This program determines the look-up tables to be
c stored at the receiver in corpsk4-7_d3. These are
c
c          (a) co0(i,j): Array of mid-points
c          (b) co1(i,j): Array of end-points
c          (c) ics(i,j): Array of new states
c
c The program sets up the input and output files and
c then inputs q,q1,tr1,tr2, and the phase response filter
c hd(i). The program then decides on the number of
c states defined by the look-up tables and determines
c the arrays in a loop. The mid- and end-point arrays
c are found by starting from a particular state and
c input symbol and passing the appropriate symbols
c through the phase response filter. The new state is
c determined from the old state variables and the output
c phase point. Finally the arrays are printed in .absout
c along with the other pertinent parameters and the
c tables are also outputed to a separate file for use
c in corpsk4-7_d3/d4
c This version also determines the minimum phase
c equivalents of the channel filters and incorporates
c these into the determination of the Rx Arrays if
c a minimum phase channel is desired.
c
c
      complex co0(0:511,0:3),co1(0:511,0:3),st0(-300:100),
     1st2(-300:100),st3(-300:100),fr(0:100),ft(0:100),aa
      double precision si(-1:45),hd(-500:500),a,b,sxx,
     1bb,ahd,sph(-100:100)
      integer q,q1,tr1,tr2,ics(0:511,0:3),s,ss,is(-1:10),l,
     1tr3,tr4
      character*3 filep,file1,file2,file4
      double precision tol,x02aaf,qq
      integer im(2),n,ifail
      double precision ar(100),aj(100),rr(100),rj(100)
      complex xvect(100),xval,xsum,xroot,f(2,64)
      open(0,defer=.true.,prompt=.true.)
      write(0,)"Input Parameter Filename"
      read(0,)filep
      write(0,)"Input Data Filename"
      read(0,)file1
      write(0,)"Output Filename"
      read(0,)file2
      write(0,)"Channel Filter & Parameters file"
      read(0,)file4
      open(0,defer=.false.)
      open(1,file=filep,form='formatted',mode='in')
      open(2,file=file1,form='formatted',mode='in')
      open(4,file=file4,form='formatted',mode='in')
      open(6,file=file2,form='formatted',mode='out')
      read(1,*)q,q1,tr1,tr2
      read(1,*)ifft,ishift,isa,ila
c
c
c Input hd(i)
```

```
c
      pi=dacos(-1.0d+00)
      j1=-tr1*q1
      j2=q1*(tr2+2)
      do 10 i=j1,j2,1
      read(2,*)hd(i)
   10 continue
c     do 12 i=j1,0,1
c     hd(i)=0.0
c  12 continue
c     do 14 i=17,j2,1
c     hd(i)=hd(16)
c  14 continue
c
c Read in the filters and their lengths
c
      read(4,*)tr3,tr4
      j3=q1*tr3
      j4=q1*tr4
      do 390 i=1,j3+1,1
      read(4,*)b1,b2
      f(1,i)=cmplx(b1,b2)
  390 continue
      do 395 i=1,j4+1,1
      read(4,*)b1,b2
      f(2,i)=cmplx(b1,b2)
  395 continue
c
c Linear/minimum phase choice
c
      ilmc=0
      im(1)=j3+1
      im(2)=j4+1
      if(ilmc.eq.1)then
c
c Filter Loop
c
      do 500 iend=1,2,1
c
c Transfer filter to ar,aj
c
      do 400 i=1,im(iend),1
      ar(i)=real(f(iend,i))
      aj(i)=aimag(f(iend,i))
  400 continue
c
c Root Calculation
c
      do 405 i=1,100,1
      rr(i)=0.0
      rj(i)=0.0
  405 continue
      ifail=0
      n=im(iend)
      tol=x02aaf(qq)
      call c02adf(ar,aj,n,rr,rj,tol,ifail)
c
c Check for failure
c
      if(ifail.ne.0.or.n.ne.1)then
      write(0,410)n,ifail
  410 format('Algorithm Failure'/10x,'n = ',i2,2x,'ifail = ',i2)
      else
c
c Multiply out factors as a check
c
      xvect(1)=(1.0,0.0)
      do 415 i=2,im(iend),1
      xvect(i)=(0.0,0.0)
  415 continue
      do 425 j=1,im(iend),1
      xsum=(0.0,0.0)
      xroot=cmplx(-rr(j),-rj(j))
      i=1
  420 xval=xvect(i)
      xvect(i)=xvect(i)+xsum
      i=i+1
      xsum=xval*xroot
      if(i-im(iend))420,420,425
  425 continue
      do 430 i=1,im(iend),1
      xvect(i)=xvect(i)*f(iend,1)
  430 continue
      write(0,435)(xvect(i),i=1,im(iend)-1),
     1(rr(i),i=1,im(iend)-1),(rj(i),i=1,im(iend)-1)
  435 format('Factor Multiplication-Test Results'/
     12(4(f10.6,3h + ,1hj,f10.6,2x)//)
     2'rr = ',1(8(f10.6,2x)/)//
     3'rj = ',1(8(f10.6,2x)/)//)
      endif
c
c
c Take the complex conjugate of the roots
c outside of the unit circle.
c
c
      do 440 i=1,im(iend),1
      rmag=rr(i)**2+rj(i)**2
      if(rmag.gt.1.0)then
      rr(i)=rr(i)/rmag
      rj(i)=rj(i)/rmag
      else
      endif
  440 continue
c
c Calculate the Minimum-phase Filter Response
c
      xvect(1)=(1.0,0.0)
      do 445 i=2,im(iend),1
      xvect(i)=(0.0,0.0)
  445 continue
      do 455 j=1,im(iend),1
      xsum=(0.0,0.0)
      xroot=cmplx(-rr(j),-rj(j))
      i=1
  450 xval=xvect(i)
      xvect(i)=xvect(i)+xsum
      i=i+1
      xsum=xval*xroot
      if(i-im(iend))450,450,455
  455 continue
c
c Multiply by Yo
c
      do 460 i=1,im(iend),1
      f(iend,i)=xvect(i)*f(iend,1)
  460 continue
  500 continue
      else
c
c End of linear/minimum phase choice
c
      endif
c
c Transfer filters to ft,fr
c
      do 503 i=1,im(1),1
      ft(i-1)=f(1,i)
  503 continue
      do 507 i=1,im(2),1
      fr(i-1)=f(2,i)
  507 continue
      ahd=4*(hd(j2-q1))
c
c
c Calculate No. states in Finite State Machine
c not including the Phase State
c
      nn=tr1+tr2
      if(isa.eq.0)then
      s=4**nn
      nnn=nn
      else
      s=4**(isa-1)
      nnn=isa-1
      endif
      j1st0=-(tr1+tr3)*q1
      j2st0=-(tr1-1)*q1
      j1st2=-(tr1+tr4)*q1
      j2st2=j2st0
      j1st3=-(tr1+1)*q1
      j2st3=j2st0
c
c Look-up Table calculation loop. Convert initial state into:-
c
c         (a) Previous symbols, si(1) to si(nn)
c         (b) Phase State, si(-1)
c
      do 100 l=0,s-1,1
      ss=l
      do 30 i=nnn,1,-1
      ii=4**(i-1)
      if(ss.lt.ii)then
      si(i)=-3.0d+00
      is(i)=0
      elseif(ss.ge.ii.and.ss.lt.(ii+ii))then
      si(i)=-1.0d+00
      is(i)=1
      ss=ss-ii
      elseif(ss.ge.(ii+ii).and.ss.lt.(3*ii))then
      si(i)=1.0d+00
      is(i)=2
      ss=ss-ii-ii
      else
      si(i)=3.0d+00
      is(i)=3
      ss=ss-ii-ii-ii
      endif
   30 continue
c
c
c
c Phase State Inner Loop
c
      do 90 l1=0,3,1
c
c Expansions Inner Loop
c
      do 80 ijj=0,3,1
c
c Initialise sph in accordance with l1 and
c reset all other arrays
c
      spp=hd(q1)+float(l1)*(1.0d+00/2.0d+00)
      do 35 i=j1,j2,1
      sph(i)=spp
   35 continue
      do 40 i=j1st0,j2st0,1
      st0(i)=(0.0,0.0)
   40 continue
      do 42 i=j2st0-q1+1,j2st0,1
      a1=cos(pi*sph(i))
      a2=sin(pi*sph(i))
      st0(i)=cmplx(a1,a2)
   42 continue
      do 45 i=j1st2,j2st2,1
      st2(i)=(0.0,0.0)
   45 continue
      do 47 j=j1st3,j2st3,1
      st3(j)=(0.0,0.0)
   47 continue
c
c Tx Filtering
c
      do 54 j=j2st2-q1+1,j2st2,1
      aa=(0.0,0.0)
      do 53 jj=0,j3,1
      aa=aa+ft(jj)*st0(j-jj)
   53 continue
      st2(j)=aa
   54 continue
c
c Rx Filtering
c
      do 56 j=j2st3-q1/2,j2st3,4
      aa=(0.0,0.0)
      do 55 jj=0,j4,1
      aa=aa+fr(jj)*st2(j-jj)
   55 continue
      st3(j)=aa
   56 continue
      if(l.eq.10.and.l1.eq.10.and.ijj.eq.1)then
      write(0,48)(sph(i),i=j1,j2)
   48 format('sph = ',f10.6)
      write(0,49)(st0(i),i=j1st0,j2st0)
   49 format('st0 = ',f10.6,3h + ,1hj,f10.6)
      write(0,51)(st2(i),i=j1st2,j2st2)
   51 format('st2 = ',f10.6,3h + ,1hj,f10.6)
      write(0,52)(st3(i),i=j1st3,j2st3)
   52 format('st3 = ',f10.6,3h + ,1hj,f10.6)
      else
      endif
c
c Pass si(1) to si(nnn) through the channel
c including the preamble si(n2)
c
      sxx=+3.0d+00
      do 59 ii=nnn+1,nnn+20,1
      sxx=-sxx
      si(ii)=sxx
   59 continue
      do 60 ij=nnn+20,1,-1
c
c Left-shift
c
      do 200 j=j1,j2-q1,1
      jj=j+q1
      sph(j)=sph(jj)
  200 continue
      do 210 j=j1st0,j2st0-q1,1
      jj=j+q1
      st0(j)=st0(jj)
  210 continue
      do 220 j=j1st2,j2st2-q1,1
      jj=j+q1
      st2(j)=st2(jj)
  220 continue
      do 222 j=j1st3,j2st3-q1,1
      jj=j+q1
      st3(j)=st3(jj)
  222 continue
c
c Premodulation Filtering
c
      do 230 j=j1,j2,1
      sph(j)=sph(j)+si(ij)*hd(j)
      if(sph(j).gt.2.0)sph(j)=sph(j)-ahd
      if(sph(j).lt.0.0)sph(j)=sph(j)+ahd
  230 continue
c
c Convert to st0
c
      do 240 j=j2st0-q1+1,j2st0,1
      a1=cos(pi*sph(j))
      a2=sin(pi*sph(j))
      st0(j)=cmplx(a1,a2)
  240 continue
c
c Tx Filtering
c
```

```
      do 260 j=j2st2-q1+1,j2st2,1
      aa=(0.0,0.0)
      do 250 jj=0,j3,1
      aa=aa+ft(jj))*st0(j-jj)
  250 continue
      st2(j)=aa
  260 continue
c
c Rx Filtering
c
      do 264 j=j2st3-q1/2,j2st3,4
      aa=(0.0,0.0)
      do 262 jj=0,j4,1
      aa=aa+fr(jj)*st2(j-jj)
  262 continue
      st3(j)=aa
  264 continue
      if(l.eq.10.and.l1.eq.10.and.ijj.eq.1)then
      write(0,48)(sph(i),i=j1,j2)
      write(0,49)(st0(i),i=j1st0,j2st0)
      write(0,51)(st2(i),i=j1st2,j2st2)
      write(0,52)(st3(i),i=j1st3,j2st3)
      else
      endif
   60 continue
c
c Expansion symbol contribution
c
c
c Shift Left
c
      do 310 j=j1,j2-q1,1
      jj=j+q1
      sph(j)=sph(jj)
  310 continue
      do 320 j=j1st0,j2st0-q1,1
      jj=j+q1
      st0(j)=st0(jj)
  320 continue
      do 330 j=j1st2,j2st2-q1,1
      jj=j+q1
      st2(j)=st2(jj)
  330 continue
      do 333 j=j1st3,j2st3-q1,1
      jj=j+q1
      st3(j)=st3(jj)
  333 continue
      is(0)=ijj
c
c Premodulation Filtering
c
      sx=2*(float(ijj)-1.5)
      do 340 j=j1,j2,1
      sph(j)=sph(j)+sx*hd(j)
      if(sph(j).gt.2.0)sph(j)=sph(j)-ahd
      if(sph(j).lt.0.0)sph(j)=sph(j)+ahd
  340 continue
c
c st0 Conversion
c
```

```
      do 350 j=j2st0-q1+1,j2st0,1
      a1=cos(pi*sph(j))
      a2=sin(pi*sph(j))
      st0(j)=cmplx(a1,a2)
  350 continue
c
c Tx Filtering
c
      do 370 j=j2st2-q1+1,j2st2,1
      aa=(0.0,0.0)
      do 360 jj=0,j3,1
      aa=aa+ft(jj)*st0(j-jj)
  360 continue
      st2(j)=aa
  370 continue
c
c Rx Filtering
c
      do 374 j=j2st3-q1/2,j2st3,4
      aa=(0.0,0.0)
      do 372 jj=0,j4,1
      aa=aa+fr(jj)*st2(j-jj)
  372 continue
      st3(j)=aa
  374 continue
      if(l.eq.10.and.l1.eq.10.and.ijj.eq.1)then
      write(0,48)(sph(i),i=j1,j2)
      write(0,49)(st0(i),i=j1st0,j2st0)
      write(0,51)(st2(i),i=j1st2,j2st2)
      write(0,52)(st3(i),i=j1st3,j2st3)
      else
      endif
c
c End of Tx: Post-amble of zero-data
c
      if(ilmc.ne.1)then
      iff=tr1+(tr3+tr4)/2
      else
      iff=tr1
      endif
      do 680 if=1,iff,1
c
c Left Shift
c
      do 600 j=j1,j2-q1,1
      jj=j+q1
      sph(j)=sph(jj)
  600 continue
      do 610 j=j1st0,j2st0-q1,1
      jj=j+q1
      st0(j)=st0(jj)
  610 continue
      do 620 j=j1st2,j2st2-q1
      jj=j+q1
      st2(j)=st2(jj)
  620 continue
      do 625 j=j1st3,j2st3-q1
      jj=j+q1
      st3(j)=st3(jj)
  625 continue
```

```
c
c Premodulation Filtering
c
      do 627 j=j1,j2,1
      sph(j)=sph(j)-0.0d+00*hd(j)
      if(sph(j).gt.2.0)sph(j)=sph(j)-ahd
      if(sph(j).lt.0.0)sph(j)=sph(j)+ahd
  627 continue
c
c Convert to st0
c
      do 630 j=j2st0-q1+1,j2st0,1
      a1=cos(pi*sph(j))
      a2=sin(pi*sph(j))
      st0(j)=cmplx(a1,a2)
  630 continue
c
c Tx Filtering
c
      do 650 j=j2st2-q1+1,j2st2,1
      aa=(0.0,0.0)
      do 640 jj=0,j3,1
      aa=aa+ft(jj)*st0(j-jj)
  640 continue
      st2(j)=aa
  650 continue
c
c Rx Filtering
c
      do 670 j=j2st3-q1/2,j2st3,4
      aa=(0.0,0.0)
      do 660 jj=0,j4,1
      aa=aa+fr(jj)*st2(j-jj)
  660 continue
      st3(j)=aa
  670 continue
      if(l.eq.10.and.l1.eq.10.and.ijj.eq.1)then
      write(0,48)(sph(i),i=j1,j2)
      write(0,49)(st0(i),i=j1st0,j2st0)
      write(0,51)(st2(i),i=j1st2,j2st2)
      write(0,52)(st3(i),i=j1st3,j2st3)
      else
      endif
  680 continue
c
c Determine initial state si(-1) from st3 at
c end-point (j2st3-q1) and thus the initial state
c
      write(0,300)st3(j2st3-q1)
  300 format('st3 = ',f10.6,3h + ,1hj,f10.6)
      pre=real(st3(j2st3-q1))
      pim=aimag(st3(j2st3-q1))
      if(abs(pre).gt.abs(pim).and.pre.gt.0.0)then
      is(-1)=0
      isold=1
      elseif(abs(pre).gt.abs(pim).and.pre.lt.0.0)then
      is(-1)=2
      isold=1+2*(4**nnn)
      elseif(abs(pre).lt.abs(pim).and.pim.gt.0.0)then
      is(-1)=1
      isold=1+4**nnn
      else
      is(-1)=3
      isold=1+3*(4**nnn)
      endif
c
c Rx Array Values
c
      co0(isold,ijj)=st3(j2st3-q1/2)
      co1(isold,ijj)=st3(j2st3)
c
c New State calculation for ics
c
      isnew=is(-1)+is(1)+is(0)-3
      if(isnew.lt.0)isnew=isnew+4
      if(isnew.gt.3)isnew=isnew-4
      isnew=(4**nnn)*isnew
      do 50 j=0,nnn-1,1
      isnew=isnew+(4**j)*is(j)
   50 continue
      ics(isold,ijj)=isnew
   80 continue
   90 continue
  100 continue
c Output.
c         (a) Everything o/p to .absout!
c         (b) Look-up tables o/p to file2, no format
c
      iss=4*s
      write(6,108)tr3,tr4
  108 format(i2,2x,i2)
      write(6,110)(ft(i),i=0,j3),(fr(i),i=0,j4),
     1((co0(i,j),j=0,3),i=0,(iss-1)),
     1((co1(i,j),j=0,3),i=0,(iss-1))
  110 format(f25.20,1x,f25.20)
      write(6,115)((ics(i,j),j=0,3),i=0,(iss-1))
  115 format(i4)
      write(0,120)q,q1,tr1,tr2,iss,(hd(i),i=j1,j2-1),
     1tr3,(ft(i),i=0,j3-1),tr4,(fr(i),i=0,j4-1),
     2((co0(i,j),j=0,3),i=0,(iss-1)),((co1(i,j),j=0,3),i=0,(iss-1)),
     3((ics(i,j),j=0,3),i=0,(iss-1))
  120 format('No. samples per T for Premod. Filter:',i5/
     1'No. samples per T for phase response:',i5/
     2'Length of Premod Filter: -',i2,' to +',i2,' symbols'/
     3'No. States: ',i3//
     4'Premod. Filter Characteristics:'/
     512(8(5x,f10.6)/)//
     6'Symbol length of Tx Channel Filter',i2/
     a'Tx Channel Filter:'/2(4(5x,f10.6,3h + ,1hj,f10.6)/)//
     7'Symbol length of Rx Channel Filter',i2/
     b'Rx Channel Filter:'/2(4(5x,f10.6,3h + ,1hj,f10.6)/)//
     8'Array of mid-points,co0:'/
     916(4(5x,f10.6,3h + ,1hj,f10.6)/)//
     a'Array of end-points,co1:'/
     b16(4(5x,f10.6,3h + ,1hj,f10.6)/)//
     c'Array of Final States,ics:'/4(16(2x,i3)/)////)
      stop
      end
```

PROGRAM FOR NEAR MAXIMUM LIKELIHOOD SYSTEM 1 DETECTION, FOR CODED 8PSK

```
JOB Z8150NR,:EUXXX,CP76(P0000,TD1280)
FTN5(DB-0/PMD)
LIBRARY(PROCLIB,*)
NAC(FTN5)
LGO.
££££S
C                PROGRAM CONV-8PSK_NML1A
C
C
C
C
C THIS PROGRAM SIMULATES THE TRANSMISSION OF CONVOLUTIONALLY ENCODED
C (RATE-2/3) BINARY SYMBOLS USING 8PSK MODULATION OVER AN AWGN CHANNEL

C WHICH INTRODUCES NO SIGNAL DISTORTION (MEMORYLESS CHANNEL). A NEAR
C MAXIMUM LIKELIHOOD PROCESS IS USED AT THE RECEIVER TO PERFORM THE
C DECODING/DETECTION PROCESS. CONVOLUTIONAL CODE 2 (WITH A CODE MEMORY

C OF 6 BITS) PROPOSED BY J. HUI AND R.J. FANG, (ICC 1981), IS USED.
C FOR MORE DETAILS SEE THE PROGRAM DOCUMENTATION ENTITLED 'SIMULATION

C OF CODED 8PSK OVER A DISTORTIONLESS CHANNEL'.
C
C
C
C
C DECLARE ALL VARIABLES
C
C
C
      PROGRAM CONV(INPUT,OUTPUT,TAPE1=INPUT,TAPE2=OUTPUT)
      DIMENSION IS(2,70),IG(3,2,4),IX(32,2,70),CX(32),
     1IA(3),IB(3),ISS(2,1),ICONV(4,3),IBB(3),CXX(32,4),IXX(32,2,70)
      REAL CC,AR,AI,RR,RI,AAR,AAI,D,ER,AEPB,W,WI,WR,MAP(8,2)
      INTEGER IQ,M,L,K,N,IE,IB1,IC,QQ,PS,IV
C
C
C
C
C INITIALISE VARIABLES
C
C
C
C
      IQ=7
      M=1
      L=45000
      K=16
      N=65
      PS=0
      IV=0
      P=0.547
C
C CODE 4
C
      IG(1,1,1)=1
      IG(1,1,2)=1
      IG(1,1,3)=1
      IG(1,1,4)=0
      IG(1,2,1)=1
      IG(1,2,2)=0
      IG(1,2,3)=1
      IG(1,2,4)=1
      IG(2,1,1)=0
      IG(2,1,2)=0
      IG(2,1,3)=0
      IG(2,1,4)=1
      IG(2,2,1)=1
      IG(2,2,2)=0
      IG(2,2,3)=1
      IG(2,2,4)=0
      IG(3,1,1)=0
      IG(3,1,2)=0
      IG(3,1,3)=0
      IG(3,1,4)=0
      IG(3,2,1)=0
      IG(3,2,2)=1
      IG(3,2,3)=1
      IG(3,2,4)=0
      AI=ATAN(1.0)
      DO 30 I=1,8,1
      MAP(I,1)=2*COS((I-1)*AI+(AI/2))
      MAP(I,2)=2*SIN((I-1)*AI+(AI/2))
   30 CONTINUE
      ICONV(1,1)=0
      ICONV(1,2)=0
      ICONV(1,3)=0
      ICONV(2,1)=IG(1,2,1)
      ICONV(2,2)=IG(2,2,1)
      ICONV(2,3)=IG(3,2,1)
      ICONV(3,1)=IG(1,1,1)
      ICONV(3,2)=IG(2,1,1)
      ICONV(3,3)=IG(3,1,1)
      DO 60 I=1,3,1
      IF(IG(I,1,1).EQ.IG(I,2,1))GOTO 40
      ICONV(4,I)=1
      GOTO 50
   40 ICONV(4,I)=0
   50 CONTINUE
   60 CONTINUE
      WRITE(2,600)
C
C
C
C
C CALL RANDOM GENERATOR ROUTINE BEFORE ALL PROGRAM LOOPS AND
C GENERATE NEXT PAIR OF SYMBOLS
C
C
C
C
      CALL G05CBF(IQ)
      DO 800 LM=1,M,1
      P=P-0.00
      IE=0
      IB1=0
      IC=0
      DO 20 I=1,2,1
      DO 10 J=1,N,1
      IS(I,J)=1
   10 CONTINUE
   20 CONTINUE
      DO 125 II=1,K,1
      DO 120 I=1,2,1
      DO 110 J=1,N,1
      IX(II,I,J)=1
  110 CONTINUE
  120 CONTINUE
      CX(II)=1.0E+06
  125 CONTINUE
      CX(1)=0.0
      DO 671 LLL=1,5,1
      DO 670 LL=1,L,1
      DO 160 I=1,2,1
      NN=N-1
      DO 155 J=1,NN,1
      JJ=J+1
      IS(I,J)=IS(I,JJ)
  155 CONTINUE
  160 CONTINUE
      DO 168 I=1,K,1
      DO 165 IL=1,2,1
      NN=N-1
      DO 162 J=1,NN,1
      JJ=J+1
      IX(I,IL,J)=IX(I,IL,JJ)
  162 CONTINUE
  165 CONTINUE
  168 CONTINUE
      DO 200 I=1,2,1
      W=G05DAF(-1.0,1.0D+00)
      IF(W)170,170,180
  170 IS(I,N)=0
      GOTO 190
  180 IS(I,N)=1
  190 CONTINUE
  200 CONTINUE
C
C
C
C
C USE IG(I,IL,J) TO CALCULATE IA(I),(I=1,2,3). CONVERT VECTOR IA INTO

C VARIABLE IV BY PERFORMING A BINARY TO DECIMAL CONVERSION. USE
C MAP(J) TO MAP THIS VALUE ONTO THE TWO QUADRATURE
C COMPONENTS TO BE TRANSMITTED,AR & AI
C
C
C
C
      DO 250 I=1,3,1
      IA(I)=0
      DO 240 J=1,2,1
      LN=N+1
      DO 230 IL=1,4,1
      LN=LN-1
      PS=IS(J,LN)*IG(I,J,IL)
      IF(PS.EQ.IA(I))GOTO 210
      IA(I)=1
      GOTO 220
  210 IA(I)=0
  220 CONTINUE
  230 CONTINUE
  240 CONTINUE
  250 CONTINUE
      IV=1+IA(3)+IA(2)+IA(2)+IA(1)+IA(1)+IA(1)+IA(1)
      AR=MAP(IV,1)
      AI=MAP(IV,2)
C
C
C
C
C THE QUADRATURE COMPONENTS,AR &AI,ARE NOW TRANSMITTED AND ARE
C SUBJECTED TO THE AWGN COMPONENTS,WR &WI,WHICH ARE GENERATED
C USING A RANDOM NUMBER GENERATOR WITH A GAUSSIAN PDF,WITH IT'S
C STANDARD DEVIATION GIVEN BY P
C
C
      WR=G05DDF(0.0,P)
      RR=AR+WR
      WI=G05DDF(0.0,P)
      RI=AI+WI
C
C
C NEAR MAXIMUM LIKELIHOOD DECODING/DETECTION IS NOW PERFORMED.
C THE COSTS OF EACH OF THE 4 EXPANSIONS, (0,0),(0,1),(1,0),
C AND (1,1) ARE CALCULATED FOR EACH OF THE INITIAL IX.
C THIS IS DONE BY CODING AND MAPPING THE EXPANDED VECTORS
C AND THEN FINDING THE EUCLIDEAN DISTANCE BETWEEN THIS
C AND THE SIGNAL ACTUALLY RECEIVED FOR EACH EXPANSION.
C
C
      DO 360 I=1,K,1
      DO 340 I2=1,3,1
      IB(I2)=0
      DO 330 J=1,2,1
      LN=N
      DO 320 IL=2,4,1
      LN=LN-1
      PS=IX(I,J,LN)*IG(I2,J,IL)
      IF(PS.EQ.IB(I2))GOTO 300
      IB(I2)=1
      GOTO 310
  300 IB(I2)=0
  310 CONTINUE
  320 CONTINUE
  330 CONTINUE
  340 CONTINUE
      DO 358 JJ=1,4,1
      DO 355 I2=1,3,1
      IBB(I2)=IB(I2)
      IF(ICONV(JJ,I2)-IBB(I2))342,345,342
  342 IBB(I2)=1
      GOTO 350
  345 IBB(I2)=0
  350 CONTINUE
  355 CONTINUE
      IV=1+IBB(3)+IBB(2)+IBB(2)+IBB(1)+IBB(1)+IBB(1)+IBB(1)
      AAR=MAP(IV,1)
      AAI=MAP(IV,2)
      CXX(I,JJ)=((RR-AAR)*(RR-AAR))+((RI-AAI)*(RI-AAI))+CX(I)
C
C MAG/SUM COST
C
```

```
C     CXX(I,JJ)=ABS(RR-AAR)+ABS(RI-AAI)+CX(I)
  358 CONTINUE
  360 CONTINUE
C
C
C
C
C THE EXPANSION ASSOCIATED WITH THE MINIMUM COST CXX(I,J) IS
C FOUND AND THE ELEMENTS IN THE LEFT-MOST POSITIONS OF IXX  ARE
C THE DETECTED VALUES CORRESPONDING TO THE ELEMENTS IS(1,1)
C AND IS(2,1) IN THE TRANSMITTED SIGNAL.
C ALL IX WHICH DO NOT CONTAIN THE DETECTED VALUES
C ARE DISCARDED BY ASSIGNING VERY HIGH COSTS TO THEM.
C
C
      CC=10.0E+06
      DO 400 I=1,K,1
      DO 390 J=1,4,1
      IF(CXX(I,J)-CC)370,380,380
  370 CC=CXX(I,J)
      III=I
      JJJ=J
  380 CONTINUE
  390 CONTINUE
  400 CONTINUE
      DO 410 J=1,2,1
      NN=N-1
      DO 405 IL=1,NN,1
      IXX(1,J,IL)=IX(III,J,IL)
  405 CONTINUE
  410 CONTINUE
      CX(1)=CC
      CXX(III,JJJ)=100.0E+06
      IF(JJJ-2)415,420,425
  415 IXX(1,1,N)=0
      IXX(1,2,N)=0
      GOTO 440
  420 IXX(1,1,N)=0
      IXX(1,2,N)=1
      GOTO 440
  425 IF(JJJ-4)430,435,435
  430 IXX(1,1,N)=1
      IXX(1,2,N)=0
      GOTO 440
  435 IXX(1,1,N)=1
      IXX(1,2,N)=1
  440 CONTINUE
      ISS(1,1)=IXX(1,1,1)
      ISS(2,1)=IXX(1,2,1)
      DO 470 I=1,K,1
      IF(IX(I,1,1)-IXX(1,1,1))450,445,450
  445 IF(IX(I,2,1)-IXX(1,2,1))450,460,450
  450 DO 455 J=1,4,1
      CXX(I,J)=100.0E+06
  455 CONTINUE
  460 CONTINUE
  470 CONTINUE
C
C
C SELECT THE [K-1] REMAINING VECTORS WHICH HAVE
```

```
C THE SMALLEST COSTS.
C
C
      DO 590 I=2,K,1
      CC=10.0E+06
      DO 510 II=1,K,1
      DO 500 J=1,4,1
      IF(CXX(II,J)-CC)480,490,490
  480 CC=CXX(II,J)
      III=II
      JJJ=J
  490 CONTINUE
  500 CONTINUE
  510 CONTINUE
      CX(I)=CC
      CXX(III,JJJ)=100.0E+06
      DO 520 J=1,2,1
      NN=N-1
      DO 515 IL=1,NN,1
      IXX(I,J,IL)=IX(III,J,IL)
  515 CONTINUE
  520 CONTINUE
      IF(JJJ-2)530,540,550
  530 IXX(I,1,N)=0
      IXX(I,2,N)=0
      GOTO 580
  540 IXX(I,1,N)=0
      IXX(I,2,N)=1
      GOTO 580
  550 IF(JJJ-4)560,570,570
  560 IXX(I,1,N)=1
      IXX(I,2,N)=0
      GOTO 580
  570 IXX(I,1,N)=1
      IXX(I,2,N)=1
  580 CONTINUE
  590 CONTINUE
C
C
C TRANSFER THE IXX BACK INTO THE IX VECTORS.
C
C
      CC=CX(1)
      DO 598 I=1,K,1
      DO 595 J=1,2,1
      DO 592 IL=1,N,1
      IX(I,J,IL)=IXX(I,J,IL)
  592 CONTINUE
  595 CONTINUE
      CX(I)=CX(I)-CC
  598 CONTINUE
C
C
C THE NEXT SECTION TESTS FOR ERRORS IN THE DETECTED PAIR OF
C DIGITS. THE BIT ERROR COUNT,IE,IS INCREMENTED WHENEVER A
C BIT ERROR OCCURS. IF THE NUMBER OF CORRECTLY DETECTED
C BINARY SYMBOLS SINCE THE LAST ERROR IS GREATER OR EQUAL
C TO 20,THE BURST ERROR COUNTER,IB1,IS INCREMENTED ON
C THE OCCURRENCE OF AN ERROR. OTHERWISE,(IF AN ERROR
C HAS OCCURRED),THE COUNT OF CORRECTLY DETECTED SYMBOLS,IC,
C IS SET TO ZERO. IN ADDITION,WHEN THE FIRST ERROR OCCURS,
C IB1 IS SET TO ONE.
C
C
C
C
      DO 660 I=1,2,1
      IC=IC+1
      IF(IS(I,1)-ISS(I,1))605,650,605
  605 IE=IE+1
      IF(IE.NE.1)GOTO 610
      IB1=1
      GOTO 625
  610 IF(IC-20)630,630,620
  620 IB1=IB1+1
  625 CONTINUE
  630 IC=0
  650 CONTINUE
  660 CONTINUE
  670 CONTINUE
  671 CONTINUE
C
C
C
C
C THE ERROR RATE,ER,AND THE AVERAGE NUMBER OF ERRORS PER BURST
C AEPB,ARE NOW CALCULATED. THE SNR IS ALSO CALCULATED AND
C THE RESULTS ARE SENT TO THE OUTPUT.
C
C
C
C
      ER=(FLOAT(IE))/FLOAT(L+L)/5.0
      IF(IB1.EQ.0)GOTO 680
      AEPB=(FLOAT(IE))/(FLOAT(IB1))
      GOTO 690
  680 AEPB=0
  690 CONTINUE
      SNR=10.0*ALOG10(2.0 /(P*P))
  600 FORMAT(1H ,10X,4H SNR,10X,10HERROR RATE,
     110X,16HERRORS PER BURST)
      WRITE(2,700)SNR,ER,AEPB
  700 FORMAT(1H ,7X,F9.5,7X,E12.5,13X,F9.5)
  800 CONTINUE
C
C
C
C
C A NUMBER OF IMPORTANT PARAMETERS ARE PRINTED OUT
C
C
C
C
      WRITE(2,900)(MAP(J,1),J=1,8),(MAP(J,2),J=1,8),P,IQ,L,K,N
  900 FORMAT(1H ,10X,'MAP1 = ',8F9.5/1H ,10X,
     1'MAP2 = ',8F9.5/1H ,10X,
     2'P = ',F6.4,5X,'IQ = ',I3,5X,'L = ',I6,5X,'K = ',I2,
     45X,'N = ',I2////)
      WRITE(2,950)(((IX(I,J,MM),MM=1,33),J=1,2),I=1,8)
  950 FORMAT(1H ,10X,66I1/)
      STOP
      END
```

B6 PROGRAM FOR NOISE-ADAPTIVE, VITERBI-TYPE, DETECTION, FOR CODED 8PSK

```
c
c        PROGRAM CONV-8PSK_VIT3F
c
c
c This program simulates the use of the VA detection
c scheme on the conv. code/phase mapping modulation
c using a variable number of states and a variable
c number of expansions per initial state. The
c restrictions are probabilistic in nature:
c
c         (a) No. states restricted by max. cost
c             constraint, cxmax
c
c         (b) No. expansions for a given initial state
c             constrained by;
c
c             either (i) hard limit on No. boundary
c                        relative to Rx sample
c                or (ii) a set of cost thresholds
c
c In addition a hard limit on the max. No. states can be set.
c
c Declarations
c
      library 'nagf'
      integer j4(0:2),ig(3,2,4),icc(64),
     1ib(3),ibb(3),iconv(4,3),j3(2,0:2),icp(64,600),ic02(64),
     2ffin(0:63),fexp(0:63,0:3),isinit(0:63),
     3isinit2(0:63),cst(0:63,0:3),cot(0:63,0:3),jold(0:63),
     4jnew(0:63),ifull(0:63),icheck(0:63),ix(0:63,65),
     5ixd(2),is(65,2)
      real map(8,2),cx(0:63),carr(8),xx(0:63,0:3),ccc(64),
     1cth(4)
      double precision g05ddf,g05daf,p
      character*3 file1,file2
      open(0,defer=.true.,prompt=.true.)
      write(0,)"Run Parameters File"
      read(0,)file1
      write(0,)"Graphics File"
      read(0,)file2
      open(0,defer=.false.)
      open(1,file=file1,form='formatted',mode='in')
      open(2,file=file2,form='formatted',mode='out')
      read(1,*)iq,m,l,l1,n,p,pp,ilim,cxmax,ismax,
     1ja,(cth(i),i=1,4)
c
c Code Initialisation
c
c CODE 1
c
      ig(1,1,1)=0
      ig(1,1,2)=1
      ig(1,1,3)=0
      ig(1,2,1)=1
      ig(1,2,2)=0
      ig(1,2,3)=1
      ig(2,1,1)=1
      ig(2,1,2)=1
      ig(2,1,3)=1
      ig(2,2,1)=0
      ig(2,2,2)=0
      ig(2,2,3)=1
      ig(3,1,1)=0
      ig(3,1,2)=0
      ig(3,1,3)=0
      ig(3,2,1)=0
      ig(3,2,2)=1
      ig(3,2,3)=0
c
c CODE 3
c
c     ig(1,1,1)=1
c     ig(1,1,2)=0
c     ig(1,1,3)=1
c     ig(1,1,4)=1
c     ig(1,2,1)=1
c     ig(1,2,2)=0
c     ig(1,2,3)=0
c     ig(1,2,4)=1
c     ig(2,1,1)=0
c     ig(2,1,2)=1
c     ig(2,1,3)=0
c     ig(2,1,4)=1
c     ig(2,2,1)=1
c     ig(2,2,2)=0
c     ig(2,2,3)=0
c     ig(2,2,4)=0
c     ig(3,1,1)=0
c     ig(3,1,2)=0
c     ig(3,1,3)=0
c     ig(3,1,4)=0
c     ig(3,2,1)=0
c     ig(3,2,2)=0
c     ig(3,2,3)=1
c     ig(3,2,4)=1
c
c Initialise Coder F.S. Machine
c
      if(ja.eq.15)then
      jaa=1
      else
      jaa=2
      endif
      iconv(1,1)=0
      iconv(1,2)=0
      iconv(1,3)=0
      iconv(2,1)=ig(1,2,1)
      iconv(2,2)=ig(2,2,1)
      iconv(2,3)=ig(3,2,1)
      iconv(4,1)=ig(1,1,1)
      iconv(4,2)=ig(2,1,1)
      iconv(4,3)=ig(3,1,1)
      do 16 i=1,3,1
      if(ig(i,1,1).eq.ig(i,2,1))then
      iconv(3,i)=0
      else
      iconv(3,i)=1
      endif
   16 continue
c
```

```
c State decomposition into symbol values
c
      do 150 i=0,ja,1
      ii=i
      do 30 j=jaa,0,-1
      jj=4**j
      if(ii.ge.3*jj)then
      j4(j)=3
      j3(1,j)=1
      j3(2,j)=0
      ii=ii-(3*jj)
      elseif(ii.lt.3*jj.and.ii.ge.2*jj)then
      j4(j)=2
      j3(1,j)=1
      j3(2,j)=1
      ii=ii-(2*jj)
      elseif(ii.lt.2*jj.and.ii.ge.jj)then
      j4(j)=1
      j3(1,j)=0
      j3(2,j)=1
      ii=ii-jj
      else
      j4(j)=0
      j3(1,j)=0
      j3(2,j)=0
      endif
   30 continue
c
c Partial coding;ie of state j3 alone
c
      do 90 i2=1,3,1
      ib(i2)=0
      do 80 jj=1,2,1
      do 70 il=2,jaa+2,1
      ps=j3(jj,(il-2))*ig(i2,jj,il)
      if(ps-ib(i2))40,50,40
   40 ib(i2)=1
      goto 60
   50 ib(i2)=0
   60 continue
   70 continue
   80 continue
   90 continue
c
c Completion of Coding/element determination
c
      do 140 j=0,3,1
      do 130 i2=1,3,1
      ibb(i2)=ib(i2)
      if(iconv((j+1),i2).eq.ibb(i2))then
      ibb(i2)=0
      else
      ibb(i2)=1
      endif
  130 continue
c
c Set up state/element link vectors and Tx FS machine
c
      jold(i)=j4(jaa)
      if(ja.eq.15)then
      imm=4
      else
      imm=16
      endif
      cst(i,j)=4*(i-(imm*j4(jaa)))+j
      jnew(cst(i,j))=j
      cot(i,j)=1+ibb(3)+2*ibb(2)+4*ibb(1)
  140 continue
  150 continue
c
c Initialise constellation mapping
c
      ai=atan(1.0)
      do 160 i=1,8,1
      map(i,1)=2*cos((i-1)*ai+(ai/2.0))
      map(i,2)=2*sin((i-1)*ai+(ai/2.0))
  160 continue
c
c SNR loop
c
      call g05cbf(iq)
      write(0,600)
      do 800 lm=1,m,1
      p=p-pp
      ie=0
      ib1=0
      ic=0
      ss1=0.0
      es1=0.0
      do 739 i=1,ja+1,1
      icc(i)=0
  739 continue
      istemp=1
      is1=0
      isep=n-1
      do 180 i=1,n,1
      do 165 ij=0,ja,1
      ix(ij,i)=0
  165 continue
      do 170 j=1,2,1
      is(i,j)=0
  170 continue
  180 continue
      do 190 i=1,ja,1
      cx(i)=10.0e+06
      isinit(i)=-1
  190 continue
      cx(0)=0.0
      isinit(0)=0
c
c Tx loop
c
      do 671 lll=1,ll,1
      do 670 ll=1,l,1
c
c Left shift
c
      isep=isep+1
      if(isep.gt.n)isep=1
      isbp=isep+1
      if(isbp.gt.n)isbp=1
c
c Data Generation
c
      w=g05daf(-2.0d+00,2.0d+00)
      if(w.lt.-1.0d+00)then
      isx=0
      is(isep,1)=0
      is(isep,2)=0
      elseif(w.ge.-1.0d+00.and.w.lt.0.0d+00)then
      isx=1
      is(isep,1)=0
      is(isep,2)=1
      elseif(w.ge.0.0d+00.and.w.lt.1.0d+00)then
      isx=2
      is(isep,1)=1
      is(isep,2)=1
      else
      isx=3
      is(isep,1)=1
      is(isep,2)=0
      endif
c
c Conv. coding/mapping
c
      iv=cot(is1,isx)
      is1=cst(is1,isx)
      ar=map(iv,1)
      ai=map(iv,2)
c
c Noise addition
c
      wr=g05ddf(0.0d+00,p)
      rr=ar+wr
      wi=g05ddf(0.0d+00,p)
      ri=ai+wi
c
c Rx:
c Threshold test the Rx sample to allow expansion validity
c testing later on. The VA is used in a reconfigurable sense with
c a variable No. of states and a variable No. of expansion
c per initial state.
c
c Threshold testing
c
      if(rr.gt.0.0.and.ri.ge.0.0.and.abs(rr).gt.abs(ri))then
      ivv=1
      elseif(rr.gt.0.0.and.ri.gt.0.0.and.abs(ri).ge.abs(rr))then
      ivv=2
      elseif(rr.le.0.0.and.ri.gt.0.0.and.abs(ri).gt.abs(rr))then
      ivv=3
      elseif(rr.lt.0.0.and.ri.gt.0.0.and.abs(rr).ge.abs(ri))then
      ivv=4
      elseif(rr.lt.0.0.and.ri.le.0.0.and.abs(rr).gt.abs(ri))then
      ivv=5
      elseif(rr.lt.0.0.and.ri.lt.0.0.and.abs(ri).ge.abs(rr))then
      ivv=6
      elseif(rr.ge.0.0.and.ri.lt.0.0.and.abs(ri).gt.abs(rr))then
      ivv=7
      else
      ivv=8
      endif
c
c Incremental cost determination
c
      do 260 j=1,8,1
      carr(j)=(rr-map(j,1))**2+(ri-map(j,2))**2
  260 continue
c
c VA
c
c Reset final state flags
c
      do 420 i=0,ja,1
      ffin(i)=0
      do 410 j=0,3,1
      fexp(i,j)=-1
  410 continue
  420 continue
c
c Initial state/expansions loop
c
      if(ll.gt.11000)then
      write(0,1010)(isinit(i),i=0,15)
 1010 format('isinit=',16i3/)
      else
      endif
      istemp2=istemp
      istemp=0
      do 439 i=0,ja,1
      if(isinit(i).ne.-1)istemp=istemp+1
  439 continue
      do 1111 j=2,ja+1,1
      if(istemp.ge.j)then
      ic02(j)=ic02(j)+1
      else
      if(ic02(j).ne.0)icp(j,ic02(j))=icp(j,ic02(j))+1
      ic02(j)=0
      endif
 1111 continue
      ss1=ss1+float(istemp)
      icc(istemp)=icc(istemp)+1
      do 440 ist=0,ja,1
c
c Check on existence of initial state
c
      if(isinit(ist).ne.-1)then
c
c If flexible ilim allocation is in operation
c threshold-test initial state's cost.
c
      if(cx(ist).le.cth(4))then
      ilim=4
      elseif(cx(ist).gt.cth(4).and.cx(ist).le.cth(3))then
      ilim=3
      elseif(cx(ist).gt.cth(3).and.cx(ist).le.cth(2))then
      ilim=2
      else
      ilim=1
      endif
      do 430 iex=0,3,1
c
c Check on legality of state transition
c
      iv1=cot(ist,iex)
      ivd=iabs(iv1-ivv)
      if(ivd.gt.4)ivd=8-ivd
      if(ivd.le.ilim)then
      es1=es1+1.0
c
c Transfer vector/state linkage and flag existence of
c final state
c
      fexp(cst(ist,iex),jold(ist))=isinit(ist)
      ffin(cst(ist,iex))=1
c
c Costing
c
      xx(cst(ist,iex),jold(ist))=cx(ist)+carr(cot(ist,iex))
      if(ll.gt.11000)then
      write(0,1020)ist,iex,cst(ist,iex),jold(ist),
     1xx(cst(ist,iex),jold(ist))
 1020 format('ist=',i3,2x,'iex=',i3,2x,'cs=',i3,2x,'jold=',
     1i3,2x,'xx=',f8.5)
      else
      endif
      else
      endif
  430 continue
      else
      endif
  440 continue
      if(ll.gt.11000)then
      write(0,1030)((fexp(i,j),j=0,3),i=0,15),(ffin(i),i=0,15)
 1030 format('fexp='/4(10x,16i3/)/'ffin=',16i3)
      else
      endif
c
c Selection
c
c Reset isinit and ifull flags
c
      do 450 i=0,ja,1
      ifull(i)=0
      isinit(i)=-1
  450 continue
      do 470 isf=0,ja,1
      if(ffin(isf).eq.1)then
      cc=10.0e+06
      do 460 j=0,3,1
      if(fexp(isf,j).ne.-1)then
      if(xx(isf,j).lt.cc)then
      cc=xx(isf,j)
      jchos=j
      else
      endif
      else
      endif
  460 continue
      isinit2(isf)=fexp(isf,jchos)
      cx(isf)=xx(isf,jchos)
      ifull(isinit2(isf))=1
      else
      endif
  470 continue
      if(ll.gt.11000)then
      do 1043 i=0,15,1
      write(0,1040)isinit2(i),cx(i),ifull(isinit2(i))
 1040 format('isinit2=',i3,3x,'cx=',f8.5,3x,'ifull=',i3)
 1043 continue
      else
      endif
c
c Restoration of unique vector/state relationship
c
      do 480 i=0,ja,1
      icheck(i)=0
  480 continue
      ifull1=0
      do 510 isf=0,ja,1
      if(ffin(isf).eq.1)then
      if(icheck(isinit2(isf)).eq.0)then
      isinit(isf)=isinit2(isf)
      icheck(isinit(isf))=1
      ix(isinit(isf),isep)=jnew(isf)
      else
      iflag=0
      do 500 j=ifull1,ja,1
      if(iflag.ne.1)then
      if(ifull(j).eq.0)then
      do 490 ij=1,n,1
      ix(j,ij)=ix(isinit2(isf),ij)
  490 continue
      ix(j,isep)=jnew(isf)
      isinit(isf)=j
      ifull(j)=1
      iflag=1
      ifull1=j+1
      else
      endif
      else
      endif
  500 continue
      endif
      else
      endif
  510 continue
      if(ll.gt.11000)then
      write(0,1050)(isinit(i),i=0,15)
 1050 format('isinit=',16i3)
      else
      endif
c
c Detection
c
      cc=10.0e+06
      do 520 isf=0,ja,1
      if(ffin(isf).eq.1)then
      if(cx(isf).lt.cc)then
      cc=cx(isf)
      ii=isf
      else
      endif
      else
      endif
  520 continue
      if(ix(isinit(ii),isbp).eq.0)then
      ixd(1)=0
      ixd(2)=0
      elseif(ix(isinit(ii),isbp).eq.1)then
      ixd(1)=0
      ixd(2)=1
      elseif(ix(isinit(ii),isbp).eq.2)then
      ixd(1)=1
      ixd(2)=1
      else
      ixd(1)=1
      ixd(2)=0
      endif
      if(ll.gt.11000)then
      write(0,1060)ixd(1),ixd(2),
     1is(isbp,1),is(isbp,2)
 1060 format('ixd=',2i2,3x,'is=',2i2)
      else
      endif
c
c Cost size reduction
c
      cc=cx(ii)
      do 540 i=0,ja,1
      cx(i)=cx(i)-cc
  540 continue
      icount=0
      do 542 i=0,ja,1
      if(cx(i).gt.cxmax)isinit(i)=-1
      if(isinit(i).ne.-1)icount=icount+1
  542 continue
c
c Check No. states does not exceed ismax
c
      if(icount.gt.ismax)then
      do 733 i=1,(icount-ismax),1
      cc=0.0
      do 731 ij=0,ja,1
      if(isinit(ij).ne.-1)then
      if(cx(ij).gt.cc)then
      cc=cx(ij)
      ii=ij
      else
      endif
      else
      endif
  731 continue
      isinit(ii)=-1
  733 continue
      else
      endif
c
c Error Count
c
```

```
      do 695 i=1,2,1
      ic=ic+1
      if(is(isbp,i).ne.ixd(i))then
      ie=ie+1
      if(ie.ne.1)goto 683
      ib1=1
      goto 685
  683 if(ic.gt.20)then
      ib1=ib1+1
      else
      endif
  685 continue
      ic=0
      else
      endif
  695 continue
  670 CONTINUE
  671 CONTINUE
C
C
C
C
C THE ERROR RATE,ER,AND THE AVERAGE NUMBER OF ERRORS PER BURST
C AEPB,ARE NOW CALCULATED. THE SNR IS ALSO CALCULATED AND
C THE RESULTS ARE SENT TO THE OUTPUT.
C
C
C
C
      ER=(FLOAT(IE))/(FLOAT(L)*FLOAT(L1)*2.0)
      IF(IB1.EQ.0)GOTO 680
      AEPB=(FLOAT(IE))/(FLOAT(IB1))
      GOTO 690
  680 AEPB=0
  690 CONTINUE
      eee=2.0/(p*p)
      SNR=10.0*alog10(eee)
      es1=es1/(float(l)*float(l1))
      ss1=ss1/(float(l)*float(l1))
      do 737 i=1,ja+1,1
      if(icc(i).ne.0)then
      ccc(i)=(float(icc(i))*100.0)/(float(l1)*float(l))
      else
      ccc(i)=0.0
      endif
  737 continue
  600 FORMAT(1H ,10X,4H SNR,10X,10HERROR RATE,
     110X,16HERRORS PER BURST)
      write(0,700)SNR,ER,AEPB
  700 FORMAT(1H ,7X,F9.5,7X,E12.5,13X,F9.5)
  800 CONTINUE
C
C
C
C
C A NUMBER OF IMPORTANT PARAMETERS ARE PRINTED OUT
C
C
C
C
```

```
      do 1011 j=2,ja+1,1
      do 1009 i=1,300,1
      write(2,*)i,icp(j,i)
 1009 continue
 1011 continue
      write(0,900)(MAP(J,1),J=1,8),(MAP(J,2),J=1,8),P,PP,IQ,L,L1,N
  900 FORMAT(1H ,10X,'MAP1 = ',8F9.5/1H ,10X,
     1'MAP2 = ',8F9.5/1H ,10x,
     2'P = ',F6.4,5X,'PP = ',F6.4,5X,'IQ = ',I3,5X,
     3'L = ',I6,5X,'L1 = ',I3,5X,
     45X,'N = ',I2////)
      write(0,940)(cth(i),i=1,4),cxmax,ismax
  940 format('ilim Cost Thresholds',4f9.5//'Max. Cost='f9.5/
     1'Max. No. States=',i3/)
      write(0,945)isep,((is(i,j),j=1,2),i=1,n)
  945 format('isep=',i3/'Tx Data'/2(5x,65i1/)/)
      write(0,950)((ix(i,j),j=1,n),i=0,ja)
  950 format('Rx Vectors'/64(5x,65i1/))
      write(0,955)(cx(i),i=0,ja)
  955 format('cx='/64(10x,f10.7/))
      write(0,960)(isinit(i),i=0,ja)
  960 format('isinit='/4(10x,16i3/))
      write(0,930)((cst(i,j),j=0,3),i=0,ja),((cot(i,j),j=0,3),i=0,ja)

  930 format('Tx code FS Machine final state look-up Table'/
     116(5x,16i3/)/'Tx code FS Machine o/p look-up Table'/16(5x,16i3/))

      write(0,933)(jold(i),i=0,ja),(jnew(i),i=0,ja)
  933 format('jold='/4(10x,16i2/),/'jnew='/4(10x,16i2/))
      write(0,934)ss1,es1
  934 format('Av. No. States/Interval=',f9.5/
     1'Av. No. Expansions/Interval=',f9.5)
      do 936 i=1,ja+1,1
      write(0,937)i,ccc(i)
  937 format('State Count :',i5,2x,'% Occurrence :',f9.5)
  936 continue
      stop
      end
```

B9 PROGRAM FOR VITERBI DETECTION, FOR CODED 8PSK USING CODE 1 OF TABLE 2.5.1

```
CJOB Z8150P3,:EUXXX,CP76(P0000,TD1280)
CFTN5(DB=0/PMD)
CLIBRARY(PROCLIB,*)
CNAG(FTN5)
CLGO.
C££££S
C      PROGRAM PSKVIT(INPUT,OUTPUT,TAPE1=INPUT,TAPE2=OUTPUT)
C              PROGRAM CONV-8PSK_VIT3
C
C
C
C
C THIS PROGRAM SIMULATES THE TRANSMISSION OF CONVOLUTIONALLY ENCODED
C (RATE-2/3) BINARY SYMBOLS USING 8PSK MODULATION OVER AN AWGN CHANNEL

C WHICH INTRODUCES NO SIGNAL DISTORTION (MEMORYLESS CHANNEL). THE
C VITERBI ALGORITHM  IS USED AT THE RECEIVER TO PERFORM THE
C DECODING/DETECTION PROCESS. A CONVOLUTIONAL CODE (WITH A CODE MEMORY

C OF 4 BITS) PROPOSED BY J. HUI AND R.J. FANG, (ICC 1981), IS USED.
C FOR MORE DETAILS SEE THE PROGRAM DOCUMENTATION ENTITLED 'SIMULATION

C OF CODED 8PSK OVER A DISTORTIONLESS CHANNEL'.
C
C
C
C
C DECLARE ALL VARIABLES
C
C
C
      LIBRARY 'NAGF'
      DIMENSION IS(2,85),IG(3,2,3),IX(16,2,85),IRES(3),ICHOS(3),
     1CX(16),IA(3),IB(4,3),ISS(2,1),XX(4),IBB(3),CXX(4),ICONV(4,3),
     1ISD(2,4),ISSD(2,1),IS1(2)
      REAL CC,AR,AI,RR,RI,AAR,AAI,ER,AEPB,W,WI,WR,MAP(8,2)
      INTEGER IQ,M,L,K,N,IE,IB1,IC,PS,IV,E(4,4),EX(4,4)
      DOUBLE PRECISION G05DDF,G05DAF,P
C
C
C
C
C INITIALISE VARIABLES
C
C
C
C
      IQ=30
      M=1
      K=16
      L=400
      N=65
      PS=0
      IV=0
      P=0.00
C
C CODE 1
C
      IG(1,1,1)=0
      IG(1,1,2)=1
      IG(1,1,3)=0
      IG(1,2,1)=1
      IG(1,2,2)=0
```

```
   IG(1,2,3)=1
   IG(2,1,1)=1
   IG(2,1,2)=1
   IG(2,1,3)=1
   IG(2,2,1)=0
   IG(2,2,2)=0
   IG(2,2,3)=1
   IG(3,1,1)=0
   IG(3,1,2)=0
   IG(3,1,3)=0
   IG(3,2,1)=0
   IG(3,2,2)=1
   IG(3,2,3)=0
   AI=ATAN(1.0)
   DO 30 I=1,8,1
   MAP(I,1)=2*COS((I-1)*AI+(AI/2))
   MAP(I,2)=2*SIN((I-1)*AI+(AI/2))
30 CONTINUE
   ICONV(1,1)=0
   ICONV(1,2)=0
   ICONV(1,3)=0
   ICONV(2,1)=IG(1,2,1)
   ICONV(2,2)=IG(2,2,1)
   ICONV(2,3)=IG(3,2,1)
   ICONV(3,1)=IG(1,1,1)
   ICONV(3,2)=IG(2,1,1)
   ICONV(3,3)=IG(3,1,1)
   DO 45 I=1,3,1
   IF(IG(I,1,1).EQ.IG(I,2,1))GOTO 35
   ICONV(4,I)=1
   GOTO 40
35 ICONV(4,I)=0
40 CONTINUE
45 CONTINUE
   WRITE(1,600)
   CALL G05CBF(IQ)
   DO 800 LM=1,M,1
   DO 20 I=1,2,1
   DO 10 J=1,N,1
   IS(I,J)=0
10 CONTINUE
20 CONTINUE
   DO 60 I=1,16,1
   DO 55 IL=1,2,1
   NN=N-2
   DO 50 J=1,NN,1
   IX(I,IL,J)=0
50 CONTINUE
55 CONTINUE
60 CONTINUE
   DO 65 I=1,4,1
   IX(I,1,N)=0
   IX(I,2,N)=0
65 CONTINUE
   DO 70 I=5,8,1
   IX(I,1,N)=0
   IX(I,2,N)=1
70 CONTINUE
   DO 75 I=9,12,1
   IX(I,1,N)=1
```

```
      IX(I,2,N)=0
   75 CONTINUE
      DO 80 I=13,16,1
      IX(I,1,N)=1
      IX(I,2,N)=1
   80 CONTINUE
      NN=N-1
      DO 90 I=1,13,4
      IX(I,1,NN)=0
      IX(I,2,NN)=0
   90 CONTINUE
      DO 130 I=2,14,4
      IX(I,1,NN)=0
      IX(I,2,NN)=1
  130 CONTINUE
      DO 135 I=3,15,4
      IX(I,1,NN)=1
      IX(I,2,NN)=0
  135 CONTINUE
      DO 140 I=4,16,4
      IX(I,1,NN)=1
      IX(I,2,NN)=1
  140 CONTINUE
      DO 141 I=1,16,1
      CX(I)=1.0E+06
  141 CONTINUE
      CX(1)=0.0
      IS1(1)=0
      ISDD=0
      ISD(1,1)=0
      ISD(2,1)=0
      ISD(1,2)=0
      ISD(2,2)=0
      ISD(1,3)=0
      ISD(2,3)=0
      II=0
      DO 150 J=1,4,1
      DO 145 IL=1,4,1
      II=II+1
      E(IL,J)=II
  145 CONTINUE
  150 CONTINUE
C
C
C CALL RANDOM GENERATOR ROUTINE BEFORE ALL PROGRAM LOOPS AND
C GENERATE NEXT PAIR OF SYMBOLS
C
C
      P=P-0.00
      IE=0
      IB1=0
      IC=0
      DO 671 LLL=1,1,1
      DO 670 LL=1,L,1
      DO 160 I=1,2,1
      NN=N-1
      DO 155 J=1,NN,1
      JJ=J+1
      IS(I,J)=IS(I,JJ)
  155 CONTINUE
  160 CONTINUE
      DO 166 I=1,16,1
      DO 164 IL=1,2,1
      NN=N-1
      DO 162 J=1,NN,1
      JJ=J+1
      IX(I,IL,J)=IX(I,IL,JJ)
  162 CONTINUE
  164 CONTINUE
  166 CONTINUE
      DO 169 I=1,2,1
      ISD(I,3)=ISD(I,2)
      ISD(I,2)=ISD(I,1)
  169 CONTINUE
      DO 200 I=1,2,1
      W=G05DAF(-1.0D+00,1.0D+00)
      IF(W)170,170,180
  170 IS(I,N)=0
      GOTO 190
  180 IS(I,N)=1
  190 CONTINUE
  200 CONTINUE
      ID=IS(1,N)+IS(1,N)+IS(2,N)
      IF(ID-2)206,202,204
  202 ID=3
      GOTO 206
  204 ID=2
  206 CONTINUE
C
C DIFF. ENCODE ID
C
      ISDD=ID-ISDD
      IF(ISDD.LT.0)ISDD=ISDD+4
      IF(ISDD.EQ.0)THEN
      ISD(1,1)=0
      ISD(2,1)=0
      ELSEIF(ISDD.EQ.1)THEN
      ISD(1,1)=0
      ISD(2,1)=1
      ELSEIF(ISDD.EQ.2)THEN
      ISD(1,1)=1
      ISD(2,1)=1
      ELSE
      ISD(1,1)=1
      ISD(2,1)=0
      ENDIF
C
C     ISD(1,1)=IS(1,N)
C     ISD(2,1)=IS(2,N)
C
C
C
C
C
C USE IG(I,IL,J) TO CALCULATE IA(I),(I=1,2,3). CONVERT VECTOR IA INTO

C VARIABLE IV BY PERFORMING A BINARY TO DECIMAL CONVERSION. USE
C MAP(J) TO MAP THIS VALUE ONTO THE TWO QUADRATURE
C COMPONENTS TO BE TRANSMITTED,AR & AI
C
C
```

```
C
C
      DO 250 I=1,3,1
      IA(I)=0
      DO 240 J=1,2,1
      DO 230 IL=1,3,1
      PS=ISD(J,IL)*IG(I,J,IL)
      IF(PS.EQ.IA(I))GOTO 210
      IA(I)=1
      GOTO 220
  210 IA(I)=0
  220 CONTINUE
  230 CONTINUE
  240 CONTINUE
  250 CONTINUE
      IV=1+IA(3)+(2*IA(2))+(4*IA(1))
C
C SUDDEN PHASE SHIFT
C
C     IF(LL.GT.500)THEN
C     IV=IV+4
C     IF(IV.GT.8)IV=IV-8
C     ELSE
C     ENDIF
C
C
      AR=MAP(IV,1)
      AI=MAP(IV,2)
C
C
C
C
C THE QUADRATURE COMPONENTS,AR &AI,ARE NOW TRANSMITTED AND ARE
C SUBJECTED TO THE AWGN COMPONENTS,WR &WI,WHICH ARE GENERATED
C USING A RANDOM NUMBER GENERATOR WITH A GAUSSIAN PDF,WITH IT'S
C STANDARD DEVIATION GIVEN BY P
C
C
C
C
      WR=G05DDF(0.0D+00,P)
      RR=AR+WR
      WI=G05DDF(0.0D+00,P)
      RI=AI+WI
C
C
C DETECTION PROCESS: EACH GROUP OF 4 IX'S CORRESPONDING TO
C THE E( ,I,J)'S IS TAKEN IN TURN. THE IX'S ARE EXPANDED
C AND THEIR COSTS ARE CALCULATED. THEY ARE THEN RANKED TO
C FORM THE GROUP OF FOUR POINTERS EX(I,J, ).
C
C FOR EACH SUBGROUP OF 4 E( ,I,J)'S RAW VERSIONS OF IB(I)
C ARE CALCULATED FROM THE APPROPRIATE UNEXPANDED IX'S. THEN
C THE PRE-CALCULATED ICONV(I,J) ARE APPENDED TO GAIN FINAL
C VERSIONS OF IB(I) CORRESPONDING TO THE REQUIRED EXPANSION.
C FOR EACH EXPANSION, CORRESPONDING TO ONE OF 4 EX(I,J, ),
C THE CODED RESULTS ARE MAPPED AND THE NEW COSTS ARE CALCULATED.
C THE FOUR RESULTS CORRESPONDING TO A PARTICULAR EX(I,J, )
C ARE COMPARED, AND THE BEST VECTOR IS CHOSEN, WHERE
C EX(I,J, ) POINTS TO IT.
```

```
C
C
      DO 435 J=1,4,1
      DO 283 IJ=1,4,1
      II=E(IJ,J)
      DO 280 I2=1,3,1
      IB(IJ,I2)=0
      DO 275 JJ=1,2,1
      LN=N
      DO 270 IL=2,3,1
      LN=LN-1
      PS=IX(II,JJ,LN)*IG(I2,JJ,IL)
      IF(PS-IB(IJ,I2))255,260,255
  255 IB(IJ,I2)=1
      GOTO 265
  260 IB(IJ,I2)=0
  265 CONTINUE
  270 CONTINUE
  275 CONTINUE
  280 CONTINUE
  283 CONTINUE
      DO 325 JJ=1,4,1
      DO 305 IJ=1,4,1
      II=E(IJ,J)
      DO 300 I2=1,3,1
      IBB(I2)=IB(IJ,I2)
      IF(ICONV(JJ,I2)-IBB(I2))285,290,285
  285 IBB(I2)=1
      GOTO 295
  290 IBB(I2)=0
  295 CONTINUE
  300 CONTINUE
      IV=1+IBB(3)+IBB(2)+IBB(2)+IBB(1)+IBB(1)+IBB(1)+IBB(1)
      AAR=MAP(IV,1)
      AAI=MAP(IV,2)
      XX(IJ)=CX(II)+((RR-AAR)*(RR-AAR))+((RI-AAI)*(RI-AAI))
  305 CONTINUE
C
C RANK THE 4 CALCULATED COSTS TO FIND THE VALUE OF E(I,J,JJ),I=1 TO 4

C
      CC=+10.0E+06
      DO 320 IJ=1,4,1
      IF(XX(IJ)-CC)310,315,315
  310 CC=XX(IJ)
      III=IJ
  315 CONTINUE
  320 CONTINUE
      EX(J,JJ)=E(III,J)
      CXX(JJ)=XX(III)
  325 CONTINUE
C
C
C THE NEXT SECTION FINDS THOSE IX'S POINTED TO BY THE 4 E'S
C WHICH ARE NOT INCLUDED AMONGST THE EX'S BECAUSE OF
C DUPLICATION IN THE EX'S. THESE ARE STORED IN IRES(I)
C I=1 TO 3
C
C
      IRES(1)=0
      IRES(3)=0
      ID=0
      DO 355 IZZ=1,4,1
      III=0
      DO 340 IZ=1,4,1
      IF(E(IZZ,J)-EX(J,IZ))335,330,335
  330 III=1
  335 CONTINUE
  340 CONTINUE
      IF(III)350,345,350
  345 ID=ID+1
      IRES(ID)=E(IZZ,J)
  350 CONTINUE
  355 CONTINUE
C
C
C THE NEXT SECTION INITIALISES A STORE TO NOTE THOSE IX'S
C WHICH HAVE BEEN ASSIGNED PERMANENTLY TO EX'S. THEN THE IX
C POINTED TO BY EX(I,J,1) IS EXPANDED AND IT'S COST IS PLACED
C IN IT'S CX POSITION. IT'S DESIGNATION IS ALSO STORED IN
C THE STORE,ICHOS(I),NOTED ABOVE. THE PROCESS THEN MOVES ON
C TO EX(I,J,2). IF IT IS NOT INCLUDED IN ICHOS(I), THE IX IT
C POINTS TO IS EXPANDED ETC AS BEFORE. OTHERWISE A 'SPARE'
C IX IS FOUND FROM AMONG THE IRES(I).
C
C
      ICHOS(1)=0
      ICHOS(2)=0
      ICHOS(3)=0
      II=EX(J,1)
      IX(II,1,N)=0
      IX(II,2,N)=0
      ICHOS(1)=II
      CX(II)=CXX(1)
      IZ=1
      II=EX(J,2)
      IF(II-ICHOS(1))375,360,375
  360 EX(J,2)=IRES(IZ)
      III=II
      II=IRES(IZ)
      IZ=IZ+1
      DO 370 IJ=1,2,1
      NN=N-1
      DO 365 IL=1,NN,1
      IX(II,IJ,IL)=IX(III,IJ,IL)
  365 CONTINUE
  370 CONTINUE
  375 CONTINUE
      IX(II,1,N)=0
      IX(II,2,N)=1
      ICHOS(2)=II
      CX(II)=CXX(2)
      II=EX(J,3)
      IF(II-ICHOS(1))380,385,380
  380 IF(II-ICHOS(2))400,385,400
  385 EX(J,3)=IRES(IZ)
      III=II
      II=IRES(IZ)
      IZ=IZ+1
      NN=N-1
      DO 390 IL=1,NN,1
      IX(II,IJ,IL)=IX(III,IJ,IL)
  390 CONTINUE
  395 CONTINUE
  400 CONTINUE
      IX(II,1,N)=1
      IX(II,2,N)=0
      ICHOS(3)=II
      CX(II)=CXX(3)
      II=EX(J,4)
      IF(II-ICHOS(1))405,415,405
  405 IF(II-ICHOS(2))410,415,410
  410 IF(II ICHOS(3))430,415,430
  415 EX(J,4)=IRES(IZ)
      III=II
      II=IRES(IZ)
      DO 425 IJ=1,2,1
      NN=N-1
      DO 420 IL=1,NN,1
      IX(II,IJ,IL)=IX(III,IJ,IL)
  420 CONTINUE
  425 CONTINUE
  430 CONTINUE
      IX(II,1,N)=1
      IX(II,2,N)=1
      CX(II)=CXX(4)
  435 CONTINUE
C
C
C FROM THE 16 VECTOR PAIRS,IX,THE ONE WITH THE LOWEST COST
C IS FOUND AND THE ELEMENTS IN THE FIRST POSITION OF THE
C VECTOR PAIR ARE TAKEN TO BE THE DETECTED BITS.
C
C
      CC=+10.0E+06
      DO 455 I=1,16,1
      IF(CX(I)-CC)445,450,450
  445 CC=CX(I)
      II=I
  450 CONTINUE
  455 CONTINUE
      ISSD(1,1)=IX(II,1,1)
      ISSD(2,1)=IX(II,2,1)
      IS1(2)=ISSD(2,1)+ISSD(1,1)+ISSD(1,1)
      IF(IS1(2)-2)446,442,444
  442 IS1(2)=3
      GOTO 446
  444 IS1(2)=2
  446 CONTINUE
C
C
C DIFF. DECODE
C
C
      INN=IS1(1)+IS1(2)
      IS1(1)=IS1(2)
      IF(INN.GT.3)INN=INN-4
      IF(INN.EQ.0)THEN
      ISS(2,1)=0
      ELSEIF(INN.EQ.1)THEN
      ISS(1,1)=0
      ISS(2,1)=1
      ELSEIF(INN.EQ.2)THEN
      ISS(1,1)=1
      ISS(2,1)=1
      ELSE
      ISS(1,1)=1
      ISS(2,1)=0
      ENDIF
C
C     ISS(1,1)=IX(II,1,1)
C     ISS(2,1)=IX(II,2,1)
C
C
C NOW THE EX'S ARE TRANSFERRED BACK TO THE E'S AND CX(1) IS
C SUBTRACTED FROM ALL THE CX'S
C
      DO 465 J=1,4,1
      DO 460 IJ=1,4,1
      E(J,IJ)=EX(J,IJ)
  460 CONTINUE
  465 CONTINUE
      CC=CX(II)
      DO 475 I=1,16,1
      CX(I)=CX(I)-CC
  475 CONTINUE
C
C
C THE NEXT SECTION TESTS FOR ERRORS IN THE DETECTED PAIR OF
C DIGITS. THE BIT ERROR COUNT,IE,IS INCREMENTED WHENEVER A
C BIT ERROR OCCURS. IF THE NUMBER OF CORRECTLY DETECTED
C BINARY SYMBOLS SINCE THE LAST ERROR IS GREATER OR EQUAL
C TO 20,THE BURST ERROR COUNTER,IB1,IS INCREMENTED ON
C THE OCCURRENCE OF AN ERROR. OTHERWISE,(IF AN ERROR
C HAS OCCURRED),THE COUNT OF CORRECTLY DETECTED SYMBOLS,IC,
C IS SET TO ZERO. IN ADDITION,WHEN THE FIRST ERROR OCCURS,
C IB1 IS SET TO ZERO.
C
C
C
C
      DO 660 I=1,2,1
      IC=IC+1
      IF(IS(I,1)-ISS(I,1))605,650,605
  605 IE=IE+1
      IF(IE.NE.1)GOTO 610
      IB1=1
      GOTO 625
  610 IF(IC-20)630,630,620
  620 IB1=IB1+1
  625 CONTINUE
  630 IC=0
  650 CONTINUE
  660 CONTINUE
  670 CONTINUE
  671 CONTINUE
C
```

```
C
C
C THE ERROR RATE,ER,AND THE AVERAGE NUMBER OF ERRORS PER BURST
C AEPB,ARE NOW CALCULATED. THE SNR IS ALSO CALCULATED AND
C THE RESULTS ARE SENT TO THE OUTPUT.
C
C
C
C
      ER=(FLOAT(IE))/(FLOAT(L+L))/1.0
      IF(IB1.EQ.0)GOTO 680
      AEPB=(FLOAT(IE))/(FLOAT(IB1))
      GOTO 690
  680 AEPB=0
  690 CONTINUE
      EE=2.0/(P*P)
      SNR=10.0*ALOG10(EE)
  600 FORMAT(1H ,10X,4H SNR,10X,10HERROR RATE,
     110X,16HERRORS PER BURST)
      WRITE(1,700)SNR,ER,AEPB
  700 FORMAT(1H ,7X,F9.5,7X,E12.5,13X,F9.5)
  800 CONTINUE
C
C
C
C
C A NUMBER OF IMPORTANT PARAMETERS ARE PRINTED OUT
C
C
C
C
      WRITE(1,900)(MAP(J,1),J=1,8),(MAP(J,2),J=1,8),P,IQ,L,K,N
  900 FORMAT(1H ,10X,'MAP1 = ', 8F9.5/1H ,10X,
     1'MAP2 = ',8F9.5/1H ,10X,
     2'P = ',F6.4,5X,'IQ = ',I3,5X,'L = ',I6,5X,'K = ',I2,
     45X,'N = ',I2////)
      WRITE(1,950)(((IX(I,IL,J),J=1,65),IL=1,2),I=1,16)
  950 FORMAT(1H ,10X,65I1/)
      WRITE(1,960)(CX(I),I=1,16)
  960 FORMAT(1H ,10X,F10.6)
      STOP
      END
C££££S
C****
```

REFERENCES

1. Clark, A.P., *"Principles of Digital Data Transmission"*, 2nd Ed., (Pentech Press, 1983).

2. Clark, A.P., *"Advanced Data Transmission Systems"*, (Pentech Press, 1977).

3. Feher, K., *"Digital Communications. Satellite/Earth Station Engineering"*, (Prentice-Hall, 1983).

4. Clark, A.P., *"Digital Modems for Land Mobile Radio"*, IEE Proc. F., Commun., Radar & Signal Process., 1985, 132, (5), pp.348-362.

5. Cheung, S.W., *"Report on the Feasibility Study on the CERS Modem Design"*, Dept. of Electronic and Electrical Engin., Loughborough University of Technology, Jan., 1983.

6. Communications Group, *"Modems for the CERS Project"*, Dept. of Electronic and Electrical Engin., Loughborough University of Technology, Feb., 1984.

7. Aftelak, S.B., *"Detection Processes for Digital Satellite Modems"*, Dept. of Electronic and Electrical Engin., Loughborough University of Technology, Feb., 1984.

8. Barton, S.K., et al., *"Communications Engineering Research. Final Report of Project Definition Phase (PD1) Study of Experimental Payload and Earth Stations"*, SERC, Rutherford Appleton Lab., March, 1984.

9. Griffiths, J.W.R., et al., *"Communications Development. Satellite Modem and Optical Fibre Link"*, Project UNIVERSE Report No.2O, Dec., 1984.

1O. Clark, A.P., et al., *"Modulation/Demodulation"*, IEE Electronics Division Colloquium on CERS - Communication Engin. Research Satellite, Apr., 1984, No.6.

11. Harris, R.A., and Ulrich, S., *"Transmission Considerations for TDMA Systems Using Small Earth Terminals"*, ESA Journal, 1981, Vol. 5, pp.15-31.

12. Hui, J., and Fang, R.J.F., *"Convolutional Code and Signal Waveform Design for Band-Limited Satellite Channels"*, ICC'81, 1981 IEEE Int. Conf. on Communications, Denver, CO , U.S.A., June 1981, Vol. 3, pp.47.5/1-1O.

13. Le-Ngoc, T., and Feher, K., *"Performance of an IJF-OQPSK Modem in Cascaded Nonlinear and Regenerative Satellite Systems"*, IEEE Trans., 1983, COM-31, (2), pp.296-3O1.

14. Wu, W.W., and Shimbo, O.S., *"On-Board Process: An Overview"*, NTC'81, 1981 IEEE Nat. Telecom. Conf., New Orleans, LA , U.S.A., Nov.-Dec. 1981, pp.F7.1/1-5.

15. Reisenfeld, S., *"On-Board Processing for a 30/20 GHz Communications Satellite"*, ICC'82, 1982 IEEE Int.Conf. on Communications, Philadelphia, PA , U.S.A., June 1982, pp.5E3/1-4.

16. Huang, J.C.Y., *"Simulation Study of DQPSK INTELSAT V Regenerative/ Non-Regenerative Satellite Systems"*, NTC'81, 1981 IEEE Nat. Telecom. Conf. New Orleans, LA , U.S.A., Nov-Dec. 1981, pp. G10.6/1-5.

17. Riris, A., *"QPSK-Direct-Phase Regenerator with Imperfect Reference Carrier"*, IEE Proc. F., Commun., Radar & Signal Process., 1984, 131,(1), pp.15-18.

18. Harris, R.A., *"Transmission Aspects of the European Communications Satellite (ECS) System"*, ESA Journal, 1978, Vol. 2, pp.259-277.

19. Clark, Jr., G.C., and Cain, J.B., *"Error-Correction Coding for Digital Communications"*, (Plenum Press, 1981).

20. Ungerboeck, G., *"Channel Coding with Multilevel/Phase Signals"*, IEEE Trans., 1982, IT-28, (1), pp.55-67.

21. Lebowitz, S.H., and Rhodes, S.A., *"Performance of Coded 8PSK Signaling for Satellite Communications"*, ICC'81, 1981, IEEE Int.Conf. on Communications, Denver, CO , U.S.A., June 1981, Vol. 3, pp.47.4/1-8.

22. Fang, R.J.F., *"A Bandwidth and Power Efficient Modulation System"*, Innovations in Telecommunications, 1982, Pt.A, pp.29-57.

23. Rhodes, S.A., Fang, R.J.F., and Chang, P.Y., *"Coded Octal Phase Shift Keying in TDMA Satellite Communications"*, COMSAT Tech. Review, 1983, Vol. 13, (2), pp.221-258.

24. Taylor, D.P., and Chan, H.C., *"A Simulation Study of Two Bandwidth-Efficient Modulation Techniques"*, IEEE Trans., 1981, COM-29, (3), pp.267-275.

25. Wilson, S.G., Schottler, P.J., and Sleeper, H.A., *"Rate 3/4 16-PSK Phase Codes"*, ICC'82, 1982 IEEE Int. Conf. on Communications, Philadelphia, PA , U.S.A., June 1982, pp.6F.1/1-5.

26. Biglieri, E., *"High-Level Modulation and Coding for Nonlinear Satellite Channels"*, IEEE Trans., 1984, COM-32, (5), pp.616-626.

27. Clark, A.P., and Ser, W., *"Improvement in Tolerance to Noise Through the Transmission of Multilevel Coded Signals"*, 1981, IERE Conf. Proc. 49, pp.129-141.

28. Clark, A.P., *"Minimum-Distance Decoding of Binary Convolutional Codes"*, Computers and Digital Techniques, 1978, Vol. 1 (4), pp.190-196.

29. Huang, J.C.Y., and Feher, K., *"Performance of Bandlimited QPSK, OKQPSK and MSK Signals Through Cascaded Nonlinearities"*, ICC'79, 1979 IEEE Conf. on Communications, Pt. II, Boston, MA , U.S.A., June 1979, pp.34.4/1-5.

30. MuraKami, S., Furuya, Y., Matsuo, Y., and Sugiyama, M., *"Optimum Modulation and Channel Filters for Nonlinear Satellite Channels"*, IEEE Trans., 1979, COM-27, (12), pp.1810-1819.

31. Morais, D.H., and Feher, K., *"The Effects of Filtering and Limiting on the Performance of QPSK, Offset QPSK, and MSK Systems"*, IEEE Trans., 1980, COM-28, (12), pp.1999-2009.

32. Fang, R.J.F., *"Quaternary Transmission over Satellite Channels with Cascaded Nonlinear Elements and Adjacent Channel Interference"*, IEEE Trans., 1981, COM-29, (5), pp.567-581.

33. Osborne, W.P., and Luntz, M.B., *"Coherent and Non-coherent Detection of CPFSK"*, IEEE Trans., 1974, COM-22, (8), pp.1023-1036.

34. Muilwijk, D., and Schadé, J.H., *"Correlative Phase Modulation for Fixed Satellite Services, Pt. 1 & Supplementary Report"*, ESA Contract Reports 1981/1983.

35. Le-Ngoc, T., Feher, K., and Van, H.P., *"New Modulation Techniques for Low-Cost Power and Bandwidth Efficient Satellite Earth Stations"*, IEEE Trans., 1982, COM-30, (1), pp.275-283.

36. Le-Ngoc, T., and Feher, K., *"Performance of IJF-OQPSK Modulation Schemes in the Presence of Noise, Interchannel and Cochannel Interference"*, NTC'81, 1981 IEEE Nat. Telecom. Conf., New Orleans, LA , U.S.A., Nov-Dec. 1981, Vol. 1, pp.B7.6/1-5.

37. Anderson, J.B., and Taylor, D.P., *"A Bandwidth-Efficient Class of Signal-Space Codes"*, IEEE Trans., 1978, IT-24, (6), pp. 703-712.

38. Aulin, T., and Sundberg, C.-E., *"On the Minimum Euclidean Distance for a Class of Signal Space Codes"*, IEEE Trans., 1982, IT-28, (1), pp.43-55.

39. Aulin, T., and Sundberg, C.-E., *"Minimum Euclidean Distance and Power Spectrum for a Class of Smoothed Phase Modulation Codes with Constant Envelope"*, IEEE Trans., 1982, COM-30,(7), pp. 1721-1729.

40. Mazur, B.A., and Taylor D.P., *"Demodulation and Carrier Synchronization of Multi-h Phase Codes"*, IEEE Trans., 1981, COM-29, (3), pp.257-266.

41. Wilson, S.G., and Gaus, R.C., *"Power Spectra of Multi-h Phase Codes"*, IEEE Trans., 1981, COM-29, (3), pp.250-256.

42. Hirade, K., Murota, K., and Hata, M., *"GMSK Transmission Performance in Land Mobile Radio"*, GLOBECOM'82, 1982, IEEE Global Communications Conf., Miami, FL , U.S.A., Nov.-Dec. 1982, Vol. 1, pp.328-333.

43. Hirade, K., and Murota, K., *"A Study of Modulation for Digital Mobile Telephony"*, Conf. Rec., 29th Vehic. Technol., 1979, pp.13-19.

44. Murota, K., and Hirade, K., *"GMSK Modulation for Digital Mobile Telephony"*, IEEE Trans., 1981, COM-29, (7), pp.1044-1050.

45. Nakajima, S. and Furuya, N., *"Gaussian Filtered and Amplitude Limited MSK"*, Trans. IECE of Japan, 1981, Vol. E64, (11), pp.716-723.

46. de Jager, F., and Dekker, C.B., *"Tamed Frequency Modulation, A Novel Method to Achieve Spectrum Efficiency in Digital Transmission"*, IEEE Trans., 1978, COM-26 (5), pp.534-542.

47. Muilwijk, D., and Noordanus, J., *"Digital Phase Modulation Methods Giving a Band-Limited Spectrum for Satellite Communications"*, Proc. 8th AIAA Communications Satellite Systems Conf., Orlando, FL , April, 1980.

48. Chung, K.S., and Zegers, L.E., *"Generalized Tamed Frequency Modulation"*, Proc. ICASSP'82, 1982 IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Paris, France, May 1982, pp.1805-1808.

49. Aulin, T., Persson, B., Rydbeck, N., and Sundberg, C.-E., *"Spectrally-Efficient Constant-Amplitude Digital Modulation Schemes for Communication Satellite Applications, Vols. I & II"*, ESA Contract Report, May 1982.

50. Lender, A., *"Correlative Level Coding for Binary-Data Transmission"*, IEEE Spectrum, Feb. 1966, pp.104-115.

51. Kabal, P., and Pasupathy, S., *"Partial-Response Signaling"*, IEEE Trans., 1975, COM-23, (9), pp.921-934.

52. Deshpande, G.S., and Wittke, P.H., *"Optimum Pulse Shaping in Digital Angle Modulation"*, IEEE Trans., 1981, COM-29, (2), pp.162-168.

53. Deshpande, G.S., and Wittke, P.H., *"Correlative Encoded Digital FM"*, IEEE Trans., 1981, COM-29, (2), pp.156-161.

54. Aulin, T., and Sundberg, C.-E., *"Continuous Phase Modulation-Part I: Full Response Signaling"*, IEEE Trans., 1981, COM-29, (3), pp.196-209.

55. Aulin, T., Rydbeck, N., and Sundberg, C.-E., *"Continuous Phase Modulation - Part II: Partial Response Signaling"*, IEEE Trans., 1981, COM-29, (3), pp.210-225.

56. Aulin, T., and Sundberg, C.-E., *"CPM - An Efficient Constant Amplitude Modulation Scheme"*, submitted to Int. J. of Satellite Communications, April, 1984.

57. Aulin, T., and Sundberg, C.-E., *"CPM - The Effect of Filtering and Hard Limiting"*, submitted to Int. J. of Satellite Communications, June, 1984.

58. Lindell, G., Sundberg, C.-E., and Aulin, T., *"Minimum Euclidean Distance for Combinations of Short Rate 1/2 Convolutional Codes and CPFSK Modulation"*, IEEE Trans., 1984, IT-30, (3), pp.509-519.

59. Lindell, G., and Sundberg, C.-E., *"Multilevel Continuous Phase Modulation with High Rate Convolutional Codes"*, GLOBECOM'83, 1983 IEEE Global Communications Conf., San Diego, CA , U.S.A., Nov. 1983, pp.30.2/1-6.

60. Lindell, G., Sundberg, C.-E., and Svensson, A., *"Error Probability of Multilevel CPM with High Rate Convolutional Codes"*, Proc. The Int. Zurich Seminar on Digital Communications, Zürich, Switzerland, March, 1984, pp.F1.1/93-100.

61. Aulin, T., Sundberg, C.-E., and Svensson, A., *"Viterbi Detectors with Reduced Complexity for Partial Response Continuous Phase Modulation "*, NTC'81, 1981 IEEE Nat. Telecom. Conf., New Orleans, LA , U.S.A., Nov.-Dec. 1981, pp.A7.6/1-7.

62. Muilwijk, D., *"Correlative Phase Shift Keying - A Class of Constant Envelope Modulation Techniques"*, IEEE Trans., 1981, COM-29, (3), pp.226-236.

63. Viterbi, A.J., *"Convolutional Codes and Their Performance in Communication Systems"*, IEEE Trans., 1971, COM-19, (5), pp.751-772.

64. Clark, A.P., Harvey, J.D., and Driscoll, J.P., *"Improved Detection Processes for Distorted Digital Signals"*, 1977, IERE Conf. Proc. 37, pp.125-136.

65. Clark, A.P., Harvey, J.D., and Driscoll, J.P., *"Near-Maximum-Likelihood Detection Processes for Distorted Digital Signals"*, The Radio and Electronic Engineer, 1978, Vol. 48, (6), pp. 301-309.

66. Clark, A.P., and Fairfield, M.J., *"Detection Processes for a 9600 bit/s Modem"*, The Radio and Electronic Engineer, 1981, Vol. 51, (9), pp.455-465.

67. Clark, A.P., Kwong, C.P. and Harvey, J.D., *"Detection Processes for Severely Distorted Digital Signals"*, IEE Proc. J. Electronic Circuits and Systems, 1979, Vol. 3 (1), pp.27-37.

68. Clark, A.P., *"Distance Measures for Near-Maximum-Likelihood Detection Processes"*, IEE Proc. E., 1981, Vol. 128, (3), pp.114-122.

69. Clark, A.P., Ip, S.F.A. and Soon, C.W., *"Pseudobinary Detection Processes for a 9600 bit/s Modem"*, IEE Proc. F., Commun., Radar & Signal Process., 1982, Vol. 129, (5), pp.305-314.

70. Clark, A.P. and Clayden, M., *"Pseudobinary Viterbi Detector"*, IEE Proc. F., Commun., Radar & Signal Process., 1984, Vol.131, (2), pp.208-218.

71. Clark, A.P., Abdullah, S., Jayasinghe, S.G., and Sun, K.H., *"Pseudobinary and Pseudoquaternary Detection Processes for Linearly Distorted Multilevel QAM Signals"*, IEEE Trans., 1985, COM-33, (7), pp.639-645.

72. Gill, A., *"Introduction to the Theory of Finite-State Machines"*, (McGraw-Hill, 1962).

73. Bhargava, V.K., Haccoun, D., Matyas, R., and Nuspl, P., *"Digital Communications by Satellite. Modulation, Multiple Access and Coding"*, (John Wiley, 1981).

74. Lin, S., and Costello Jr., D.J., *"Error Control Coding: Fundamentals and Applications"*, (Prentice-Hall, 1983).

75. Najdi, H.Y., *"Digital Data Transmission Over Voice Channels"*, Ph.D. Thesis, Loughborough University, 1982.

76. Simmons, S.J., and Wittke, P.H., *"Low Complexity Decoders for Constant Envelope Digital Modulations"*, IEEE Trans., 1983, COM-31, (12), pp.1273-1280.

77. Forney Jr., G.D., *"Convolutional Codes I: Algebraic Structure"*, IEEE Trans., 1970, IT-16, (6), pp.720-738.

78. Forney Jr., G.D., *"Structural Analysis of Convolutional Codes Via Dual Codes"*, IEEE Trans., 1973, IT-19, (4), pp.512-518.

79. Peterson, W.W., and Weldon Jr., E.J., *"Error Correcting Codes"*, 2nd Ed., (MIT Press, 1972).

80. Schalkwijk, J.P.M., and Vinck, A.J., *"Syndrome Decoding of Binary Rate-1/2 Convolutional Codes"*, IEEE Trans., 1976, COM-24, (9), pp.977-985.

81. Schalkwijk, J.P.M., Vinck, A.J., and Post, K.A., *"Syndrome Decoding of Binary Rate k/n Convolutional Codes"*, IEEE Trans., 1978, IT-24, (5), pp.553-562.

82. Reed, I.S., and Truong, T.K., *"New Syndrome Decoding Techniques for the (n,k) Convolutional Codes"*, IEE Proc. F., Commun., Radar & Signal Process., 1984, Vol. 131, (4), pp.412-416.

83. Reed, I.S., and Truong, T.K., *"Error-Trellis Syndrome Decoding Techniques for Convolutional Codes"*, IEE Proc. F., Commun., Radar & Signal Process., 1985, Vol. 132, (2), pp.77-83.

84. Ng, W.H., and Goodman, R.M.F., *"An Efficient Minimum-Distance Decoding Algorithm for Convolutional Error Correcting Codes"*, Proc. IEE, 1978, Vol. 125, (2), pp.97-103.

85. Ng, W.H., and Goodman, R.M.F., *"Analysis of the Computational and Storage Requirements for the Minimum-Distance Decoding of Convolutional Codes"*, Proc. IEE, 1979, Vol. 126(1), pp.29-34.

86. Goodman, R.M.F., and Winfield, A.F.T., *"Soft-Decision Minimum-Distance Sequential Decoding Algorithm for Convolutional Codes"*, IEE Proc. F., Commun., Radar & Signal Process., 1981, Vol. 128, (3), pp.179-186.

87. Chevillat, P.R., and Costello Jr., D.J., *"A Multiple Stack Algorithm for Erasure free Decoding of Convolutional Codes"*, IEEE Trans., 1977, COM-25, (12), pp.1460-1470.

88. Vermeulen, F.L., *"Low Complexity Decoders for Channels with Intersymbol Interference"*, Ph.D. Thesis, Stanford University, CA , U.S.A., 1975.

89. Shenoy, A., and Johnson, P., *"Serial Implementation of Viterbi Decoders"*, COMSAT Tech. Review, 1983, Vol. 13,(2), pp.315-330.

90. Rader, C.M., *"Memory Management in a Viterbi Decoder"*, IEEE Trans., 1981, COM-29, (9), pp.1399-1401.

91. Texas Instruments Inc., *"TMS32010 User's Guide. 16/32-bit Digital Signal Processor"*, France, 1983.

92. Texas Instruments Inc., *"TMS32020 User's Guide, Preliminary"*, Houston, Texas, U.S.A., Jan. 1985.

93. Aftelak, S.B., and Clark, A.P., *"Adaptive Reduced-State Viterbi-Algorithm Detector"*, submitted to J. of the Institute of Radio and Electronic Engineers, July 1985.

94. Kobayashi, H., *"Correlative Level Coding and Maximum-Likelihood Decoding"*, IEEE Trans., 1971, IT-17, (5), pp.586-594.

95. Clark, A.P., *"The Transmission of Digitally-Coded Speech Signals by Means of Random Access Discrete Address Systems"*, Ph.D. Thesis, Imperial College, London, 1969.

96. Clements, A., *"The Application of Iterative Techniques to Adaptive Detection Processes"*, Ph.D. Thesis, Loughborough University, 1976.