



National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, tests publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

Comparison of Optimum Sequence and Optimum
Symbol-by-Symbol Detection in Convolutional Codes Decoding

Georgios Stamatelos

A Thesis

— In

The Department

of

Electrical Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering at
Concordia University
Montréal, Québec, Canada

April 1987

© Georgios Stamatelos, 1987


Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ISBN 0-315-44234-4



ABSTRACT

Comparison of Optimum Sequence and Optimum
Symbol-by-Symbol Detection in Convolutional Codes Decoding

Georgios Stamatelos

An optimal algorithm, originally proposed for symbol-by-symbol detection of pulse amplitude modulated sequences, is considered in this work, for optimal decoding of convolutional codes. This algorithm is similar and directly comparable to the well known Viterbi decoding algorithm, whose optimality criterion is focused on sequence detection. Due to the lack of an analytical error expression for the optimal symbol-by-symbol algorithm, computer simulations were conducted in order to observe their relative error performance.

In chapter 1 a general digital communications system employing coding is discussed, along with the channel models in use here. Convolutional codes were assumed to be the coding scheme employed in this work, thus, are also reviewed in this chapter. An extensive description of the Viterbi algorithm follows in Chapter 2, exposing the reasons for its implementation in an increased diversity of areas. The application of the symbol-by-symbol algorithm in convolutional codes decoding is then presented, as a consequence of the shared property of con-

stituting, in their general form, a solution to a variety of digital estimation problems. Concluding this chapter, Viterbi decoding of convolutional codes transmitted over channels with intersymbol interference is also discussed.

In Chapter 3 the structure of the simulation that performs the comparison between the Viterbi and the symbol-by-symbol algorithm in convolutional codes decoding is presented, along with some well-known statistical techniques, aiming at supplying with reliability and effectiveness the conducted simulations.

Finally, simulation results and a first theoretical approach to the error performance of the symbol-by-symbol algorithm conclude this work.

ACKNOWLEDGEMENT

I would like to thank Dr. J. F. Hayes for teaching and guiding me through the various aspects of work that a Master's thesis implies.

As well, I wish to thank Mrs. Lynne White for her kindness.

Table of Contents

Title Page	i
Signature Page	ii
Abstract	iii
Acknowledgements	v
Table of Contents	vi
List of Figures	viii
 Chapter 1 : Channel Models and Convolutional Codes	 1
1.1 Introduction	1
1.1.1 AWGN Channel, 1.1.2 Discrete Memoryless Channel,	
1.1.3 BSC, 1.1.4 Intersymbol Interference Channels	5
1.2 Signaling with Coded Waveforms	8
1.2.1 Convolutional Codes, 1.2.2 Trellis Codes, 1.2.3 Optimal	
Decoding of Convolutional Codes, Soft and Hard decisions	17
 Chapter 2 : The Viterbi Algorithm, and Related Techniques	 17
2.1 The Viterbi Algorithm	23
2.1.1 The Shift-register Model, 2.1.2 Decoding of Convolutional	
and Trellis Codes, 2.1.3 Demodulation of signals with ISI,	
2.1.4 The Algorithm	
2.1.5 Dynamic Programming Principle of optimality and the VA	
2.1.6 Formal Statement of the VA, its Implementation	
2.1.7 Complexity, 2.1.8 Analysis of Performance	43
2.2 Related Algorithms	49
2.2.1 Historical Background of Convolutional Codes Decoding	
2.2.2 Sequential and Feedback Decoding	

2.2.3 Comparison of Sequential and Viterbi Decoding	
2.2.4 Related Algorithms in the ISI area	54
2.3 The Optimal Symbol-by-Symbol Algorithm	57
2.3.1 Hard-decoding of Convolutional Codes 2.3.2 Optimum Sequence Detection 2.3.3 Optimum Symbol-by-Symbol Detection	
2.3.4 Formal Description of the Algorithm	
2.3.5 Merges and Complexity Requirements	69
2.4 MLSE of convolutional Codes transmitted over an ISI channel	72
2.4.1 Nonencoded Transmission 2.4.2 Encoded Transmission	
2.4.3 Implementation of the Decoder	79
Chapter 3 : Simulation and Results	85
3.1 Simulation of the Noisy Conditions of the Channel	88
3.1.1 The Gaussian Random Number Generator	89
3.2 On the Statistical Analysis of the Simulation, Results	91
3.2.1 Choice of sample size, the Independent replication method	
3.2.2 The Single Run Method 3.2.3. The Regenerative Method	98
3.3 Monte Carlo Techniques	100
3.3.1 The antithetic Variate method 3.3.2 Its Implementation	
3.3.3 The Control Variate Method	105
3.4 Results of the Simulation	108
Conclusions	115
Bibliography	117
Appendices	119
A Error correcting capability under S-b-S decoding	119
B A first approach to the error performance of the S-by-S algorithm	122
C Proof of equation B-11	131

LIST OF FIGURES

- Fig. 1.1** Block diagram of a digital communications system ()
- Fig. 1.2** Discrete memoryless channel.
- Fig. 1.3** Binary symmetric channel . .
- Fig. 1.4** Model of digital communications system with channel encoding and decoding.
- Fig. 1.5** Rate- k/n convolutional encoder.
- Fig. 1.6** $L = 3, k = 1, n = 3$ convolutional encoder.
- Fig. 1.7** State diagram for the $L = 3, k = 1, n = 3$ convolutional code.
- Fig. 1.8** Tree diagram for the $L = 3, k = 1, n = 3$ convolutional code.
- Fig. 1.9** Trellis diagram for the $L = 3, k = 1, n = 3$ convolutional code.
- Fig. 1.10** (a) Trellis source decoder and (b) the corresponding trellis diagram.
- Fig. 2.1** General model.
- Fig. 2.2** Shift-register model.
- Fig. 2.3** Model of PAM system subject to intersymbol interference.
- Fig. 2.4a** State diagram of a four state shift-register process.
- Fig. 2.4b** Trellis diagram of a four state shift-register process.
- Fig. 2.5** (a) Trellis labeled with branch lengths; $M = 4, k = 5$.
(b) Recursive determination of the shortest path via the VA.
- Fig. 2.6** The merging phenomenon - survivors at σ_{k+3} pass through 00 at σ_k .
- Fig. 2.7** Typical correct path x (heavy line) and estimated path \hat{x} (lighter line) in the trellis showing three error events.

Fig. 2.8 Typical correct path x (heavy line) and time- k incorrect subset for a four state trellis.

Fig. 2.9 Path search in sequential decoding.

Fig. 2.10 Decoding a rate-1/3 convolutional code via the stack algorithm.

Fig. 2.11 Optimal symbol-by-symbol decoding of the ω_1^{th} symbol (here $\omega_1^{th} = 4$)
(a) $a_1 = 1$ (b) $a_4 = 0$

Fig. 2.12 General transmission/reception scheme of PAM signals.

Fig. 2.13 General transmission/reception scheme of PAM and convolutionally encoded signals.

Fig. 2.14 (a) Decision between the paths 000 and 100 of the rate-1/3 $L = 3$, code.
(b) Expanded trellis description of the rate-1/3 $L = 3$, code.

Fig. 3.1 Most general flow chart.

Fig. 3.2 Regenerative states σ_0, σ_k (heavy lines: survivors).

Fig. 3.3 Performance of the rate-1/3, $K=4, 6$ and 8 codes with Viterbi soft-decoding ($\bar{Q}=8$ and by K is denoted the constraint length).

Fig. 3.4 Bit error rate versus E_b / N_0 for 1/2 Viterbi decoding and hard-quantized received data with 32-bit paths, $K=3$ through 8.

Fig. 3.5 Viterbi decoding for the rate-1/3, $L=3$ convolutional code, hard and soft decisions.

Fig. 3.6 Comparative error-performance of the two decoders, hard and soft decisions.

CHAPTER 1

Channel Models and Convolutional Codes

1.1 Introduction

The basic elements of a digital communications system are illustrated by fig.

1.1. The information source generates messages which are to be transmitted to the receiver. Depending on the type of information source, these signals can be analog or digital. Analog signals are first converted to digital form, usually into a sequence of binary digits ('source encoder' process). This digital sequence is to be transmitted through a channel to the intended receiver. A 'channel encoder' introduces redundancy in the information sequence for the purposes of combating the detrimental effects of noise and interference in the channel. Following this device a 'digital modulator' converts the digital information sequence into waveforms that are compatible with the characteristics of the channel. In order to recover the information sequence at the receiving end, the previously mentioned processes are reversed.

The channel is the media that allows communication to be established between the transmitting/receiving parts. In general it is not ideal, instead, introduces noise and other interference that corrupt the signal's transmission. Several models describe the characteristics of different types of channels and their statistical behavior. Those presented in the following sections constitute general approximations in wide use, that exhibit the memoryless property as their common characteristic.

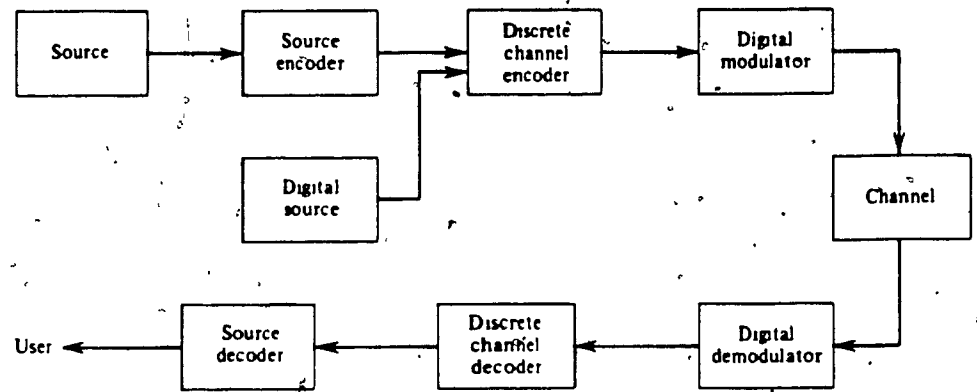


Fig. 1.1 Block diagram of a digital communications system

1.1.1 Additive White Gaussian Noise Channel .

A composite discrete-time channel, consisted of the modulator, the demodulator and the waveform channel, is characterized by an input alphabet, an output alphabet and a set of conditional probabilities relating the possible inputs to the possible outputs. The additive white Gaussian noise channel (AWGN) is a discrete-time memoryless channel, that can be modeled as having a finite input alphabet $X = \{x_0, x_1, \dots, x_{q-1}\}$, an output Y that can assume any value of the real line, i.e., $Y = \{-\infty, \infty\}$ and a set of conditional probability density functions

$$P(y | X = x_k) \quad k = 0, 1, \dots, q-1$$

that can be defined from its output formulation as :

$$Y = X + G$$

where G is a zero mean Gaussian random variable with variance σ^2 and $X = x_k$, $k = 0, 1, \dots, q-1$. For a given X , it follows that Y is Gaussian with mean x_k and variance σ^2 . That is,

$$P(y | X = x_k) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(y - x_k)^2 / 2\sigma^2} \quad (1.1)$$

For any given input sequence X_i , $i = 0, 1, \dots, n$, there is a corresponding output sequence

$$Y_i = X_i + G_i \quad i = 0, 1, \dots, n$$

and the joint conditional probability

$$P(y_1, y_2, \dots, y_n | X = u_1, X = u_2, \dots, X = u_n) = \prod_{k=1}^n P(Y = y_k | X = u_k)$$

1.1.2 Discrete Memoryless Channels.

Quantization of the output of the channel to one of Q levels, simply transforms the AWGN channel to a finite-input, finite-output alphabet channel.

As a consequence of its derivation from the AWGN channel, for which individual observables are independent, is called *discrete memoryless channel* (DMC). It is characterized by an input alphabet $X = \{a_0, a_1, \dots, a_{q-1}\}$, an output alphabet $Y = \{b_0, b_1, \dots, b_{Q-1}\}$, and a set of qQ conditional probabilities

$$P(Y = y_i | X = x_j) = P(y_i | x_j)$$

where $i = 0, 1, \dots, Q-1$ and $j = 0, 1, \dots, q-1$. In fig. 1.2, a binary input - eight level output DMC is shown.

If the input to a DMC is a sequence of n symbols u_1, u_2, \dots, u_n , selected from the alphabet X and the corresponding output is the sequence v_1, v_2, \dots, v_n , of symbols from the alphabet Y , the joint conditional probability

$$P(Y = v_1, Y = v_2, \dots, Y = v_n | X = u_1, X = u_2, \dots, X = u_n) = \prod_{k=1}^n P(Y = v_k | X = u_k)$$

The last expression restates the memoryless property of the channel.

1.1.3 Binary Symmetric Channel.

The simplest DMC has binary input and output symbols and may be derived from a binary-input AWGN channel by utilizing a two level quantizer. The quantizer, whose output is b_1 for nonnegative inputs and b_2 otherwise, is generally called a *hard quantizer* (or limiter) in contrast with a multilevel quantizer which is usually called a *soft quantizer*. The resulting hard-quantized output is the binary-symmetric channel (BSC). It has the conditional distribution diagram of fig. 1.3, with $p = P(Y = b_2 | X = a_1) = P(Y = b_1 | X = a_2)$, generally called the crossover probability, being the same as the symbol error probability for an uncoded digital communication system.

1.1.4 Intersymbol Interference channels.

Signal transmission over a band-limited non-ideal channel at high symbol rates is limited by *intersymbol interference (ISI)*. A symbol rate equal to or exceeding W (the channel's finite bandwidth constraint) results in Intersymbol Interference among a number of adjacent symbols, i.e., the transmitted pulses tend to be spread and overlap each other. Compensation for the ISI is done by designing the band-limited signals for no Intersymbol Interference for transmission rates less than $2W$, or by using a class of physically realizable pulses that are called *partial-response signals*, for transmission rates of $2W$ that allow correlation at the sampling instances but in a deterministic or 'controlled' manner, which can be

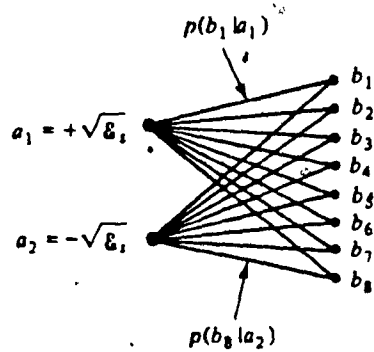


Fig. 1.2 Discrete memoryless channel.

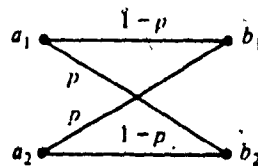


Fig. 1.3 Binary symmetric channel.

taken into account at the receiver.

Demodulation of signals corrupted by ISI is done

1. by means of equalization techniques - a whole class of devices, usually in the form of a transversal filter with its tap coefficients adjusted to minimize either the worst-case intersymbol interference at the equalizer's output, or the mean square error value of the estimation.

2. by performing *maximum-likelihood sequence estimation (MLSE)*, implemented by means of the Viterbi algorithm.

Equalization techniques constitute a clearly suboptimum solution to the problem of maximum likelihood demodulation of signals under ISI conditions. Their advantage lies in ease of implementation whereas, optimal decisions guaranteed by the application of the Viterbi algorithm, imply also the considerable decoding effort the latter technique requires.

1.2 Signaling with coded waveforms.

Arbitrary small probability of error in digital signaling can be achieved by increasing the number of orthogonal signaling waveforms and consequently expanding the required channel bandwidth. Bandwidth efficient signaling waveforms that attain comparable error performance can be generated from binary sequences, which in turn are obtained by encoding the binary information sequence. Fig. 1.4 illustrates the basic elements of a digital communications system employing coded waveforms [11].

The process of encoding constitutes a one-to-one mapping from a set of message vectors (or sequences) into a set of code vectors, an act that introduces redundancy. There are two types of encoding the information.

In *block encoding* blocks of k information bits are encoded into corresponding blocks of n bits ($n > k$). Each block of n bits from the encoder constitutes a code word contained in a set of $M = 2^k$ possible code words. The code rate is defined as the ratio k/n and is denoted by R_c .

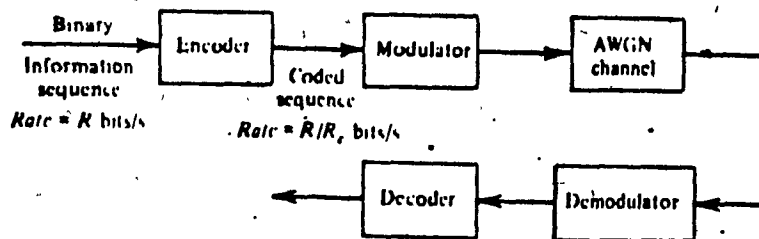


Fig. 1.4 Model of digital communications system with channel encoding and decoding.

In *convolutional encoding*, the encoder may be viewed as a 'sliding window' process that introduces correlation into the information sequence, or equivalently as a linear finite-state shift-register with an output sequence consisting of a selected set of linear combinations of the input sequence. The number of output bits from the shift-register for each input bit is a measure of the redundancy in the code and the reciprocal of this quantity is again defined as the code rate R_c . In the sequel we focus our attention on the coded output of the latter process, the convolutional codes and their analytical description.

1.2 Convolutional Codes.

Convolutional codes constitute a class of codes, whose output symbol sequence can be expressed as the convolution of input sequence with the generator sequences. This convolution is performed by passing the information sequence through a linear finite-state shift register.

In general the shift register process consists of L (k -bit) stages (L is the constraint length of the code) and n algebraic function generators, as shown in fig. 1.5. The input data to the encoder which is assumed to be binary, is shifted into and along the shift register k bits at a time. If n generally is the number of output bits for each k bits shift, the code is defined as an $R_c = \frac{k}{n}$ -rate code, consistently with the definition of the code rate for a block code.

Both block and convolutional codes can be described by a *generator matrix*, the encoder's operation in a matrix form. A generator matrix of a convolutional code is semi-infinite, since encoding operation is continuously performed over the input sequence, which is generally assumed to be semi-infinite in length. A functionally equivalent representation can be accomplished by specifying a set of n vectors, one vector for each of the n modulo-2 adders. A 1 in the i^{th} position of the vector indicates that the corresponding stage in the shift register is connected to the modulo 2 adder, whereas a 0 indicates that such a connection does not exist.

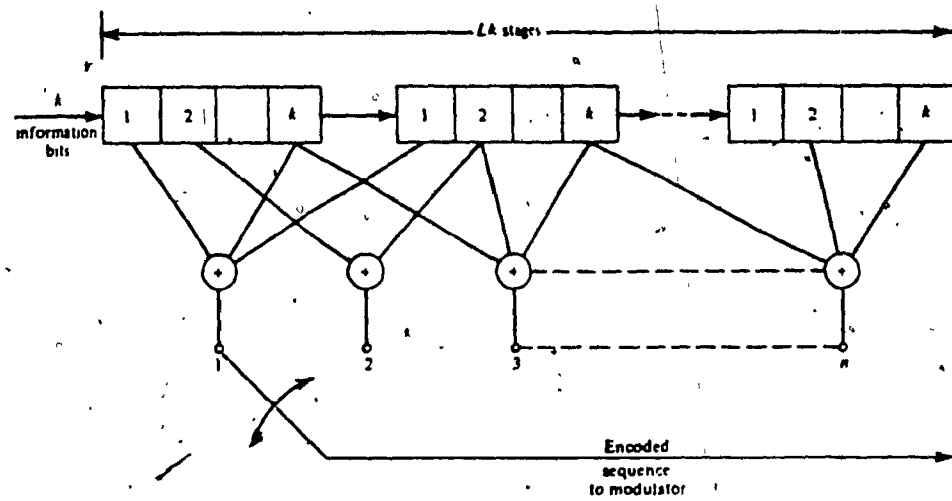


Fig. 1.5 Rate- k/n convolutional encoder.

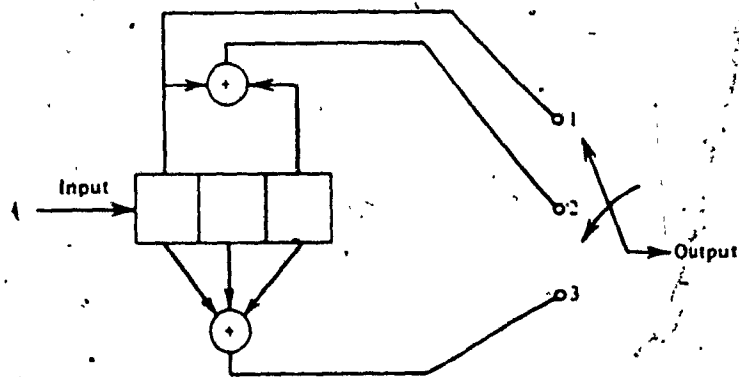


Fig. 1.6 $L=3$, $k=1$, $n=3$ convolutional encoder.

The following $L=3, k=1, n=3$, convolutional encoder, shown in fig. 1.6, serves both as a general example and an encoding process model for the purposes of a comparative error performance simulation with two different decoding techniques. Its function generators are :

$$g_1 = [100]$$

$$g_2 = [101]$$

$$g_3 = [111]$$

Alternative methods that are often used to describe a convolutional code are, the state diagram, the tree diagram, and the trellis diagram. By adopting the convention of denoting the *state* of a rate $\frac{k}{n}$ convolutional encoder by the latest $k(L-1)$ binary symbols, the *state diagram* can be used to represent the input-output relation of the linear finite-state shift register model of fig. 1.5. It is simply a graph of the possible states of the encoder and the possible transitions from one state to another. For example the state diagram for the encoder shown in fig. 1.6, is illustrated in fig. 1.7, where the letters a, b, c, d, represent the four possible states of that encoder i.e., the four possible contents of the shift register, namely 00, 01, 10, 11.

A tree diagram for the same binary encoder is illustrated in fig. 1.8. Depending on whether an incoming bit is a 0 or 1, two new branches are added to last existing edges of this tree-like construction, along with the indication of the coded output of such a shift.

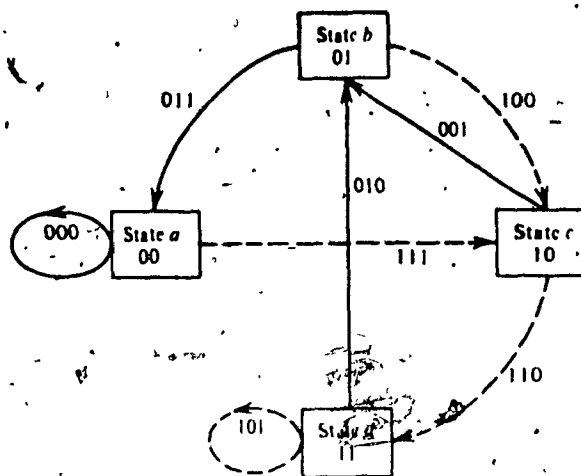


Fig. 1.7 State diagram for the $L = 3, k = 1, n = 3$ convolutional code.

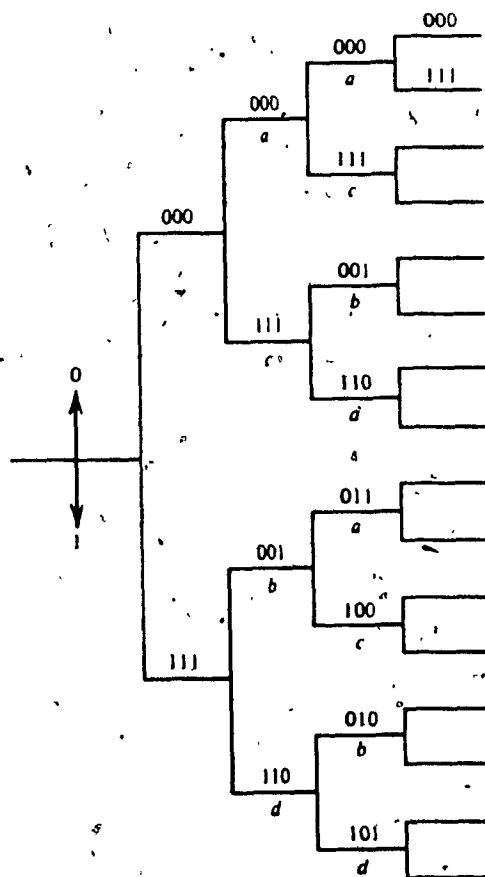


Fig. 1.8 Tree diagram for the $L = 3, k = 1, n = 3$ convolutional code.

From this diagram it becomes clear that after the third stage, the structure becomes repetitive, due to the fact the constraint length of the specific code is three ($L = 3$). By observing that all branches emanating from nodes having the same state are identical, in the sense that they generate identical output sequences, the merging of all these nodes produces a more compact diagram, called a *trellis*. The trellis diagram for the $1/3$ -rate, $L = 3$, code of fig. 1.6 is illustrated in fig. 1.9.

One subclass of convolutional codes is that of the *systematic convolutional codes*. As with systematic block codes, systematic convolutional codes have the property that the data symbols are transferred unchanged among the coded symbols. For a systematic convolutional code, in each branch the first k symbols are data symbols followed by $n-k$ parity or coded symbols. In this subclass of codes belongs the $L = 3$, $R_c = \frac{1}{3}$ -rate convolutional code, illustrated in fig. 1.6.

Forney has shown [5] that systematic feed-forward convolutional codes do not perform as well as nonsystematic convolutional codes. Viterbi in [18] shows that for asymptotically large L , the performance of a systematic code of constraint length L is approximately the same as that of a nonsystematic code of constraint length $L(1 - R_c)$ where R_c is the rate of the code.

• The Transfer Function of a Convolutional Code

The distance properties and the error rate performance of a convolutional code can be obtained from its state diagram.

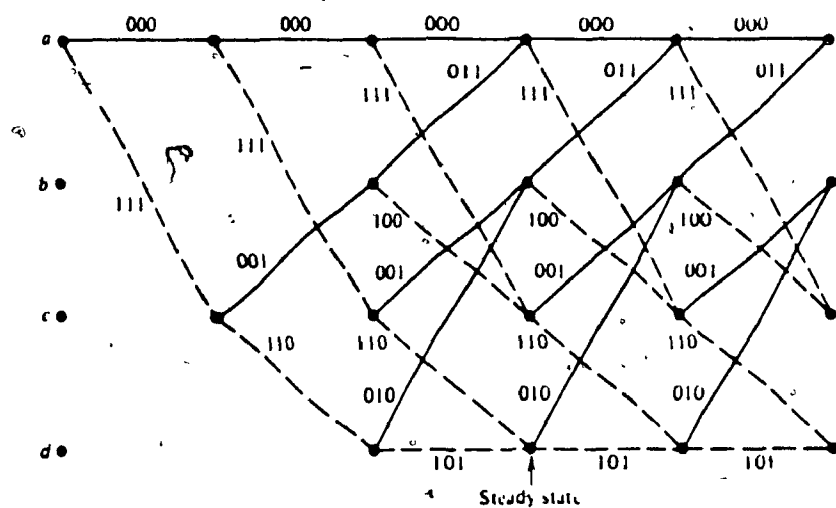


Fig. 1.9 Trellis diagram for the $L = 3, k = 1, n = 3$ convolutional code.

Assuming that the all-zero sequence is encoded and denoting by d the Hamming distance (Hamming distance is the number of corresponding elements in which two sequences differ) of an output branch-sequence, the solution of a set of state equations derived from the state diagram, leads to the formulation of the transfer function as :

$$T(D) = \sum_{d=d_{free}}^{\infty} a_d D^d$$

where a_d represents the number of paths of Hamming distance d from the all-zero path that merge with the all-zero path at a given node, and d_{free} : the minimum distance of the code. The 1/3 -rate, $L = 3$, code of fig. 1.6, can similarly be described by :

$$T(D) = \frac{D^6}{1 - 2D^2} = \sum_{d=6}^{\infty} a_d D^d$$

where $a_d = 2^{(d-6)/2}$ for d even, & $a_d = 0$ for d odd. Its minimum free distance is therefore 6 .

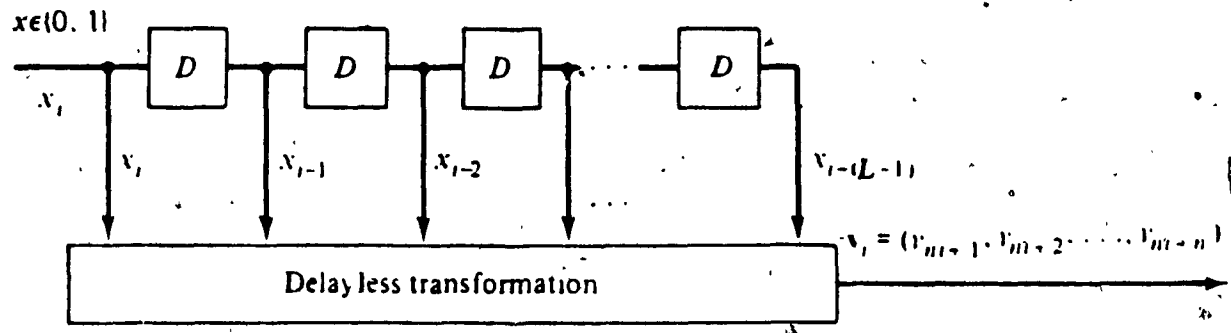
The transfer function can be easily modified to supply additional information for the number of branches and the number of information bits 1 incorporated in any given path. It is also directly used in the establishment of bounds concerning the error probability of a Viterbi decoder (Chapter 2) operating on a convolutional code transmitted over a binary-input, memoryless channel [18].

1.2.2 Trellis Codes

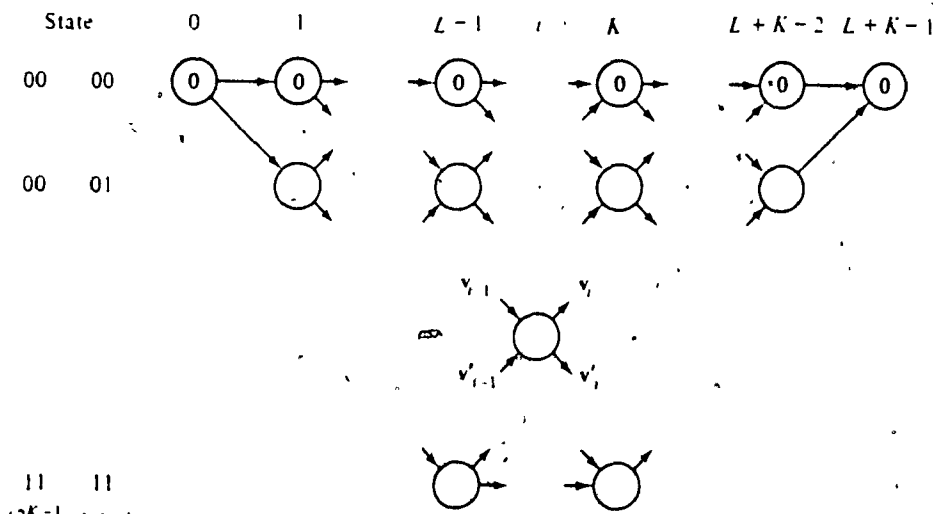
Just as linear block codes are a subclass of block codes, convolutional codes are a subclass of a broader class of codes which are called *trellis codes*. Rate $\frac{k}{n}$ trellis encoders also emit n channel symbols each time k source bits enter the register. Trellis codes are generalized convolutional codes generated by the same shift register encoder as convolutional codes, but with arbitrary delayless nonlinear operations replacing the linear combinatorial logic of the latter [18]. Whether fixed or time-varying, they can conveniently be described and analyzed by means of a trellis diagram. Fig. 1.10a shows a trellis source encoder and fig. 1.10b shows the corresponding trellis diagram for this binary trellis code with $L = 1$ delay elements and a delayless transformation.

1.2.3 Optimal decoding of convolutional codes with hard and soft decisions

In practical communication systems, when transmission occurs over channels that can be modeled as AWGN, we rarely process the actually received analog voltages, due to the computational burden that such a procedure implies. The common practice is quantization of the received voltages in order to facilitate digital processing by the receiver.



(a)



(b)

Fig. 1.10 (a) Trellis source decoder and (b) the corresponding trellis diagram.

If binary quantization is used, we say that a *hard decision* has been made on the correlator output, as to which level was actually sent, and consequently we approximate the noisy conditions of the channel by those of the BSC with crossover probability p . Consequently Hamming distance is the appropriate measure of likelihood.

When coding is used, it is desirable to keep an indication of how reliable the decision was, by computing a distance (or metric) which specifies how far from the decision threshold the demodulator output is. By allowing this quantity to take more than two possible values, say n , *soft-decoding* is performed and the model of the channel is that of the DMC with an output alphabet of size n . Hard quantization of the received data usually entails a loss of about 2dB in signal-to-noise ratio, compared with infinitely fine quantization.

Optimum decoding of convolutional codes is done by means of the Viterbi algorithm (Chap. 2), i.e., given that the received sequence is Y , this algorithm finds the most likely transmitted sequence $X^{(j)}$ that maximizes the *a posteriori* likelihood $P(Y/X^{(j)})$ computed for each possible j . When hard decisions are employed, under the BSC assumption the maximum likelihood decoder reduces to a minimum distance decoder which computes the Hamming distance from the error-corrupted received vector $Y = \{y_1, y_2, \dots, y_L\}$ to each possibly transmitted code vector $X_j = \{x_1, x_2, \dots, x_L\}$ and decides in favor of the closest code vector. In this case the crossover probability of the BSC is expressed as :

$$p = Q(\sqrt{2E_s / N_0})$$

where $Q(\cdot)$ is the Gaussian Integral function

$$Q(\beta) = \int_{\beta}^{\infty} e^{-x^2/2} \frac{dx}{\sqrt{2\pi}}$$

Now, let's assume that the coded sequence r is consisted of N vectors, each of them having n elements (n the number of output bits at each encoder's shift) denoted by $x_{ij}^{(r)}$ ($i = 1, \dots, N$ $j = 1, \dots, n$), and that binary coherent PSK modulation is employed for their transmission. The demodulator output is:

$$y_{jm} = 2aE(2x_{jm} - 1) + v_{jm}$$

where a represents the attenuation factor, E the transmitted signal energy for each code bit and v_{jm} the Gaussian noise voltage added by the channel. When soft-decoding is employed (AWGN channel), the demodulator output is statistically described by (see eq. 1.1)

$$p(y_{jm} / x_{jm}^{(r)}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-|y_{jm} - 2aE(2x_{jm}^{(r)} - 1)|^2 / 2\sigma^2}$$

where $\sigma^2 = 2EN_0$ is the variance of the additive Gaussian noise.

Then a measure of the likelihood of the possibly transmitted vector $X^{(r)}$ can be established by noting the memoryless properties of the AWGN channel in an

accumulative form as :

$$U^{(r)} = \sum_{j=1}^N \sum_{m=1}^M v_{jm} (2x_{jm}^{(r)} - 1)$$

Once the likelihood metrics were previously defined for both the decoding modes (hard and infinitely soft) the Viterbi algorithm can be further applied as a dynamic programming technique which enables the selection of their maximum, avoiding the unnecessary exhaustive computation of them all.

CHAPTER 2

The Viterbi Algorithm and Related Algorithms

2.1 The Viterbi Algorithm.

The Viterbi Algorithm (VA) was proposed in 1967 [19] as a method of decoding convolutional codes but since then has been applied on a variety of digital estimation problems, as a recursive dynamic programming technique which is used to decode digital data sequences with correlation between symbol intervals. This correlation may be introduced (in our field of interest) by a convolutional encoder or the presence of intersymbol interference in the channel.

The VA can be shown to be optimum in the sense of minimizing probability of error in detecting a sequence of symbols (MLSE). Moreover, its complexity grows linearly with the length of the sequence to be detected.

The following sections describe the analysis and the performance of the algorithm, exposing its general modeling and implementation in real situation problems.

2.1.1 The shift-register model, formal statement of the problem.

In its most general form, the VA may be viewed as a solution to the problem of maximum *a posteriori* probability (MAP) estimation of the state sequence of a finite-state discrete-time Markov process observed in memoryless noise [3].

The underlying Markov process is characterized as follows. Time is discrete. The state σ_k at time k is one of a finite number M of states m , $1 \leq m \leq M$ i.e., the state space Σ is simply $\{1, 2, \dots, M\}$. Initially we shall

assume that the process runs only from time 0 to time N and that the initial and final states σ_0 and σ_N are known ; the state sequence is then represented by a finite vector $\sigma = (\sigma_0, \dots, \sigma_N)$. Extension to infinite sequences is trivial.

The process is Markov, in the sense that the probability $P(\sigma_{k+1} | \sigma_0, \sigma_1, \dots, \sigma_k)$ of being in state σ_{k+1} at time $k+1$, given all states up to time k , depends only on the state σ_k at time k :

$$P(\sigma_{k+1} | \sigma_0, \sigma_1, \dots, \sigma_k) = P(\sigma_{k+1} | \sigma_k).$$

The transitions probabilities $P(\sigma_{k+1} | \sigma_k)$ may be time varying.

Let's define the transition ξ_k at time k as the pair of states (σ_{k+1}, σ_k) :

$$\xi_k = (\sigma_{k+1}, \sigma_k).$$

We let Ξ be the set of transitions $\xi_k = (\sigma_{k+1}, \sigma_k)$ for which $P(\sigma_{k+1} | \sigma_k) \neq 0$, and $|\Xi|$ their number. Clearly $|\Xi| \leq M^2$. There is evidently a one-to-one correspondence between state sequences σ and transition sequences $\xi = (\xi_0, \dots, \xi_{N-1})$. (we write $\sigma \rightleftharpoons \xi$).

The process is assumed to be observed in memoryless noise: that is, there is a sequence z of observations z_k in which z_k depends probabilistically only on the transition ξ_k at time k .

$$P(z | \sigma) = P(z | \xi) = \prod_{k=0}^{N-1} P(z_k | \xi_k)$$

The sequence z can be described as the output of some memoryless channel whose input sequence is ξ (see fig. 2.1). Again, the channel may be time varying in the sense that $P(z_k | \xi_k)$ may be a function of k . This formulation also implies the case in which z_k depends probabilistically on an output y_k of the process at time k , where y_k is in turn a deterministic function of the transition ξ_k or the state σ_k . (we write $y_k = f(\sigma_k)$). This case is of practical interest since it is frequently encountered in digital transmission systems and suggests the following common model for their study.

Assuming an input sequence $u = (u_0, u_1, \dots)$, where each u_k is generated independently according to some probability distribution $P(u_k)$ and can take on one of a finite number of values, say m . There is a noise-free signal sequence y , not observable in which each y_k is some deterministic function of the present and the v previous inputs :

$$y_k = f(u_k, \dots, u_{k-v})$$

The observed sequence z is the noise corrupted output of a memoryless channel whose input is y . Such a procedure is called a *shift-register process*, since it can be modeled by a shift register of length v with inputs u_k , shown in fig. 2.2. To complete the correspondence to our general Markovian model discussed previously, we define :

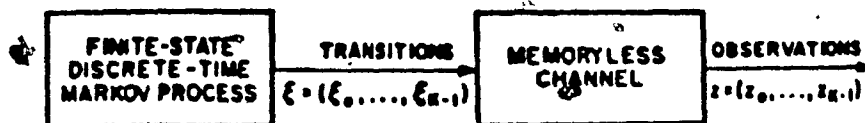


Fig. 2.1 General model.

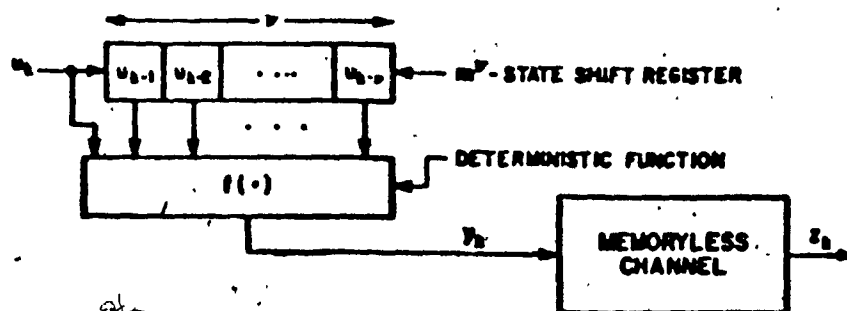


Fig. 2.2 Shift-register model.

1) the state

$$\sigma_k = (u_{k-1}, \dots, u_{k-s})$$

2) the transition

$$\xi_k = (\sigma_{k+1}, \sigma_k) = (u_k, \dots, u_{k-s})$$

The number of states is thus $|\Sigma| = m^s$, and of transitions, $|\Xi| = m^{s+1}$. If the input sequence 'starts' at time 0 and stops at time $N - s$, i.e.

$$u = (\dots, 0, u_0, u_1, \dots, u_{N-s}, 0, 0, \dots)$$

then this shift-register process effectively starts at time 0 and ends at time N with $\sigma_0 = \sigma_N = (0, 0, \dots, 0)$.

Reception of a certain sequence (signal + noise) imposes the problem of finding the most likely information sequence that was shifted into the finite-state shift-register. The VA constitutes an efficient answer to this problem which can be more formally stated as follows. Given a sequence z of observations of a discrete-time finite-state Markov process in memoryless noise, find the state sequence σ for which the *a-posteriori* probability $P(\sigma|z)$ is maximum. Alternately, find the transmission sequence ξ for which $P(\xi|z)$ is maximum (since $\sigma \rightleftharpoons^{1-1} \xi$). In the shift-register model this is also the same as finding the most probable input sequence u , since $u \rightleftharpoons^{1-1} \sigma$. It is well known that this MAP rule minimizes the error probability in detecting the whole sequence (the block-

message-, or word-error probability), and thus is optimum in this sense.

The shift-register process and the MAP estimation of the most probable input sequence \mathbf{u} , model a number of situations encountered in digital transmission, including the decoding of convolutional codes and demodulation of signals transmitted over channels with intersymbol interference. The Viterbi algorithm, being an efficient solution to the MAP estimation problem, applies to these cases above as a natural consequence. Their modeling is explained in the following sections.

2.1.2 Decoding of Convolutional and Trellis Codes.

A rate- $1/n$ binary convolutional encoder can be modeled as a shift register process circuit exactly like that of fig. 2.2, where the inputs u_k are information bits and the outputs y_k are blocks of n bits, $y_k = (p_{1k}, \dots, p_{nk})$, each of which is a parity check on (modulo 2 sum of) some subset of the $v+1$ information bits (u_k, \dots, u_{k-v}) . When the encoded sequence (codeword) \mathbf{y} is sent through a memoryless channel, we have precisely the model of fig. 2.2. Fig. 1.5 shows a particular rate $\frac{1}{3}$ code with $v = L - 1 = 2$. The general case of the $\frac{k}{n}$ -rate convolutional code has also been modeled as shift register processes [2].

Other codes of great interest can also fall into this general setup. Trellis codes were previously introduced as generalized convolutional codes generated by the same shift-register encoder but with arbitrary delayless nonlinear operations replacing the linear combinatorial logic of the latter. Ungerboeck's codes belong

to this class of codes. In 1982, he introduced a coded modulation technique suitable for band-limited channels, that can be viewed as binary convolutional codes with 'mapping by set partitioning' [20]. The later principle involves the partition of finite signal constellations into subsets, where the minimum distance between elements of the subsets is greater than that within the full signal constellation.

In order to transmit n bits/symbol by means of a two dimensional modulation scheme (QAM), a constellation of 2^{n+1} points (different signals) is used, partitioned into 4 or 8 subsets. 1 or 2 incoming bits/symbol enter a rate-1/2 or -2/3 binary convolutional encoder, and the resulting 2 or 3 coded bits/symbol specify which subset is to be used. The remaining incoming bits specify which point from the selected subset is to be used. Decoding of these codes is assumed to be performed by the Viterbi algorithm.

The model is again that of the shift-register process in fig. 2.2 . The deterministic function f plays a different role than the respective one in the convolutional codes' model, encompassing subset selection and signal specification functions.

Forney et al. [21] regarded the two dimensional constellations of Ungerboeck codes, as finite sets chosen from an infinite rectangular grid, with the subset sequence being determined by a convolutional encoder. He also proposed that, construction of 'block codes' (being viewed as finite multidimensional constellations drawn from infinite multidimensional lattices), can be accomplished in the same way and be represented by trellis diagrams.

Finally, Forney in [22], presents a unifying study on a large class of lattices and trellis codes under the name of 'coset codes', and proposes a squaring technique for their construction. The structure of these codes is such that they can be represented by 'trellis diagrams' (see Sec. 1.2.2), which naturally lead to maximum likelihood decoding algorithms (VA).

2.1.3 Demodulation of Signals with Intersymbol Interference

The concept of the Intersymbol Interference was previously discussed in Chapter 1, as a situation encountered in digital transmission through analog channels. The input sequence u , discrete-time and discrete-valued as in the shift register model, is used to modulate some continuous waveform which is transmitted through a channel and then sampled. Ideally, samples z_k would equal the corresponding u_k , but in fact are perturbed both by noise and by neighboring inputs u_k . The latter effect is called Intersymbol Interference.

In such cases the output can be modeled as

$$z_k = y_k + n_k$$

where y_k is a deterministic function of a finite number of inputs, say, $y_k = f(u_k, \dots, u_{k-L})$, and n_k is a white Gaussian noise sequence. Such a model is again identical to that of fig 2.2.

In Pulse Amplitude Modulation (PAM) the signal sequence y may be taken as the convolution of the input sequence u with some discrete-time channel

Impulse-response sequence (h_0, h_1, \dots):

$$y_k = \sum_i h_i u_{k-i}$$

If $h_i = 0$ for $i > v$ (finite impulse response) then we obtain our shift register model. An illustration of such a model in which ISI spans three time units ($v = 2$) appears in fig. 2.3.

2.1.4 The Algorithm

We now show that the MAP sequence estimation problem previously stated is formally identical to the problem of finding the shortest route through a certain graph. The VA then arises as a natural recursive solution [2].

There exist a state diagram, like that of fig 2.4a, associated with a discrete-time finite state Markov process (the similarities with the corresponding one describing a convolutional code, previously discussed in Sec. 1.2.1, are obvious).

A more redundant description of the same process, called a *trellis*, represents the same process by corresponding nodes to distinct states at a given time and branches to transitions to some new state at the next instant of time. (fig. 2.4b). The trellis begins and ends at the known states σ_0 and σ_N . Its most important property is that to every possible state sequence σ there corresponds a unique path through the *trellis*, and vice versa.

We show now, that given a sequence of observations z every path may be assigned a *metric* (whose physical representation is length) proportional to

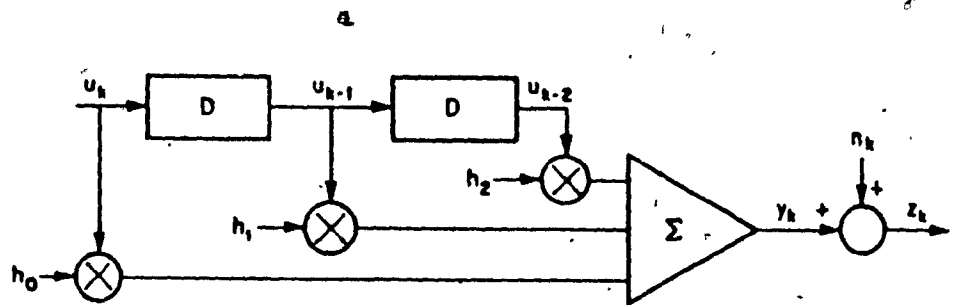


Fig. 2.3 Model of PAM system subject to intersymbol interference.

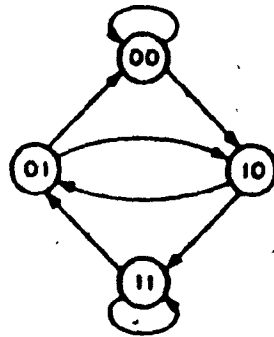


Fig. 2.4a State diagram of a four state shift-register process.

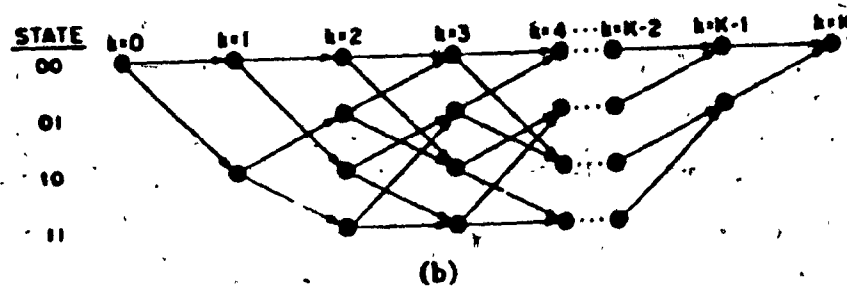


Fig. 2.4b Trellis diagram of a four state shift-register process.

$-\ln P(\sigma, z)$ where σ is the state sequence associated with this path. This allows the solution of the problem of finding the state sequence for which $P(\sigma, z)$ is maximum, or equivalently for which $P(\sigma, z) = P(\sigma/z)P(z)$ is maximum, by finding the path whose metric $-\ln P(\sigma, z)$ is minimum, since $\ln P(\sigma, z)$ is a monotonic function of $P(\sigma, z)$ and there is a one-to-one correspondence between paths and sequences. We simply observe that due to the Markov and memoryless properties, $P(\sigma, z)$ factors as follows :

$$P(\sigma, z) = P(\sigma | z)P(z) = \prod_{k=0}^{N-1} P(\sigma_{k+1} | \sigma_k) \prod_{k=0}^{N-1} P(z_k | \sigma_{k+1}, \sigma_k)$$

Hence if we assign each branch (transition) the metric

$$\lambda(\xi_k) = -\ln P(\sigma_{k+1} | \sigma_k) - \ln P(z_k | \xi_k)$$

then the total 'length' of the path corresponding to some σ is

$$-\ln P(\sigma, z) = \sum_{k=0}^{N-1} \lambda(\xi_k)$$

Having defined a measure for the likelihood of each path, an inefficient way for finding the most likely of them could be the exhaustive computation of their metrics, comparisons and selection of their minimum. The virtue of the VA is that it performs the same task much more efficiently by noting the accumulative formulation of the last expression and applying the dynamic programming principle of optimality, which simply states that from a set of converging paths only the shortest of them could potentially become part of the shortest path connecting the initial and the final point.

2.1.5 Dynamic programming principle of optimality and the VA .

The dynamic programming characteristics of the VA enable the selection of the minimum distance path (from an exponentially increasing set of paths) by a linearly increasing amount of computations. Omura [15] was the first to show that the VA was equivalent to a dynamic programming solution to the shortest route problem.

The concept of the dynamic programming involves computational procedures and techniques for finding an optimum path or a trajectory between two points in a graph. This sense of optimality can assume different qualities (length, cost, etc) in order to respond to questions imposed by real situation problems.

Hayes in [3] discusses a 'pedestrian' example (a shortest route problem in an academic setting) for demonstrating a basic principle of dynamic programming, the *principle of optimality*. The hero of this story instead of calculating the lengths of all the possible paths and selecting the shortest that lead to his destination, defines a sequence of finite sets of critical points, naturally formulated by the process of his journey. By keeping a record of only the shortest paths that lead to the next cluster of passages, he approaches his destination maintaining the certainty that the final comparison of the alternative routes at the converging point would reveal the shortest.

Using again the trellis description of a shift-register process and interpreting a possibly transmitted sequence as a 'path' and any one of its k segments-length subsequences ($k \geq v$) as 'branches', the previously discussed 'principle of optimality' imposed on the VA, states that if there is a set of branches $\{b\}$ such

that for any legitimate path containing one of the branches, any other branch in the set may be substituted with the result being another legitimate path, then a decision can be made on the best element out of that set given the corresponding portion of the received sequence, independent of the rest of the received sequence. The best such branch is called *survivor*, and its distance from the corresponding portion of the received path, possibly normalized, is called its *metric*. For further steps of decoding, the *survivor* may be taken as a proxy for the whole set.

2.1.6 Formal statement of the VA, its implementation .

Denoting by σ^k a segment $(\sigma_0, \sigma_1, \dots, \sigma_k)$ that consists of the states up to time k . In the trellis σ^k corresponds to a path segment starting at the node σ_0 and terminating at σ_k .

Its metric has the form :

$$\lambda(\sigma^k) = \sum_{i=0}^{k-1} \lambda(\xi_i)$$

For any time $k \geq 0$, there are M survivors in all, one for each σ^k , represented as $\delta(\sigma_k)$. According to the previously exposed principle, the shortest complete path should begin with one of these survivors. If it did not, but contained another path segment, say $\delta'(\sigma_k)$, then we could replace $\delta'(\sigma_k)$ by the survivor $\delta(\sigma_k)$ to get an even shorter path - contradiction, since we assumed that we deal with the shortest complete path. Thus at any time k , we need to remember only the M survivors and their metrics $\Lambda(\sigma_k) = \lambda(\delta(\sigma_k))$. All that is needed to get from time k

to time $k+1$, is the extension of all time- k survivors by one time unit, computation of the metrics ($\Lambda(\sigma_k)$) of the extended path segments and selection of the shortest extended paths terminating at each specific node σ_{k+1} . Recursion proceeds indefinitely without the number of survivors ever exceeding M .

The algorithm is illustrated for a simple four-state trellis covering 5 time units in fig. 2.5. Fig. 2.5(a) shows the complete trellis with each branch labeled with a length. Fig. 2.5(b) shows the five recursive steps by which the algorithm determines the shortest path from the initial to the final node. At each stage only the 4 (or fewer) survivors are shown, along with their metrics.

A formal description of the VA follows:

Storage:

$$\begin{aligned} k & \quad \text{(time index)} \\ \sigma(\sigma_k), 1 \leq \sigma_k \leq M, & \quad \text{(survivor terminating in } \sigma_k) \\ \Lambda(\sigma_k), 1 \leq \sigma_k \leq M, & \quad \text{(survivor length)} \end{aligned}$$

Initialization:

$$\begin{aligned} k &= 0; \\ \sigma(\sigma_0) &= \sigma_0, \quad \sigma(m) \text{ arbitrary}, \quad m \neq \sigma_0 \\ \Lambda(\sigma_0) &= 0, \quad \Lambda(m) = \infty, \quad m \neq \sigma_0 \end{aligned}$$

Recursion:

$$\Lambda(\sigma_{k+1}, \sigma_k) = \Lambda(\sigma_k) + \lambda(\xi_k) \quad \text{for all } \xi_k = (\sigma_{k+1}, \sigma_k).$$

Find

$$\Lambda(\sigma_{k+1}) = \min_{\sigma_k} \Lambda(\sigma_{k+1}, \sigma_k) \quad \text{for each } \sigma_{k+1};$$

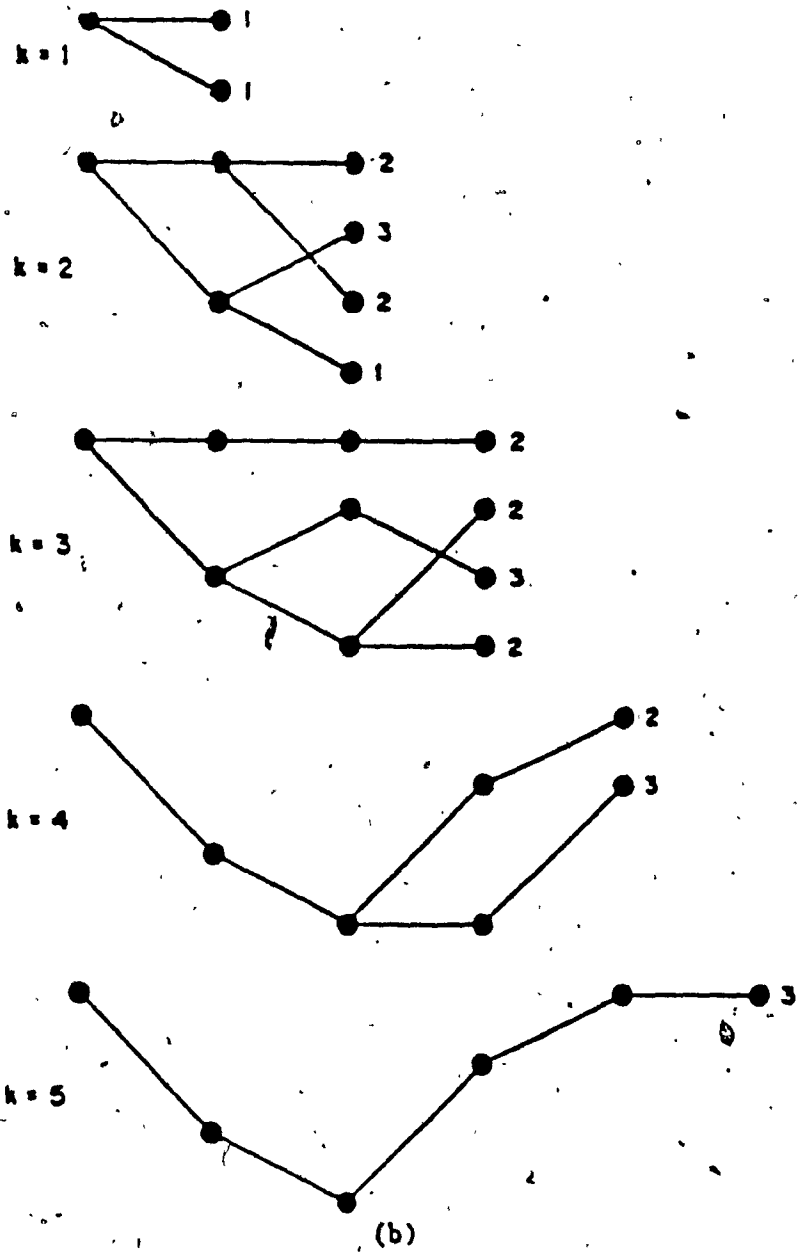


Fig. 2.5 (b) Recursive determination of the shortest path via the V.A.

Store $\Lambda(\sigma_{k+1})$ and the corresponding survivors $\delta(\sigma_{k+1})$.

Set k to $k+1$ and repeat until $k = N$.

In practical applications, when the length N can become very large, a decoder based on the algorithm described above, never actually decides upon the most likely path. It always retains a set of M paths after each decoding step. One way of selecting a single most likely path is to periodically force the shift-register process into a prearranged state by inputting v known symbols. Then the Viterbi decoder can select the survivor terminating at the known state.

Another alternative is the truncation of survivors to some length δ . That is, the algorithm must come to a definite decision on nodes up to time $k - \delta$ at time k . The event that is clearly exploited in this case is the *merge phenomenon*, a random situation in which all the survivors of a certain state, exhibit the same history at a certain depth δ , passing through the same node(s). Fig. 2.6 illustrates this event.

In general, if the truncation depth δ is chosen large enough, there is a high probability that all the time- k survivors will go through the same nodes up to time $k - \delta$, so that the initial segment of the maximum-likelihood path is known up to time $k - \delta$ and can be put out as the algorithm's decision; in this case truncation costs nothing. In the rare case when survivors disagree, any reasonable strategy for determining the algorithm's time- $k - \delta$ decision will work [6]: choose an arbitrary time- $k - \delta$ node, or the node associated with the shortest survivor, or a node chosen by majority vote, etc.

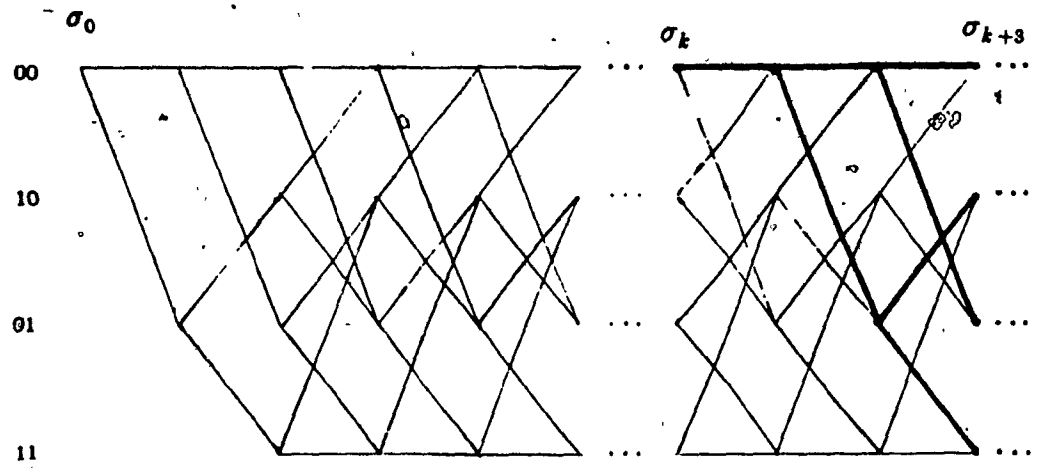


Fig. 2.6 The merging phenomenon - survivors at σ_{k+3} pass through 00 at σ_k .

If δ is large enough, the effect on performance is negligible. Also if k becomes large, it is necessary to renormalize the metrics $\Lambda(m)$ from time to time by subtracting a constant from all of them.

Focusing our attention again at the field of convolutional codes' decoding by means of the VA, random coding analysis done by Forney [5], has shown that a truncation length $\delta \geq 5L$ (L is the constraint length of the code) is sufficient to ensure that the additional error probability due to truncation is negligible.

More specifically, under a BSC assumption with crossover probability p , the final result of the bit-error expression is :

$$P_b(E) \approx \frac{1}{k} [B_{d_{free}} 2^{d_{free}} p^{d_{free}/2} + A_{d(\delta)} 2^{d(\delta)} p^{d(\delta)/2}]$$

where the first term represents the bit errors made by a standard decoder, whereas the second represents the decoding errors due to truncation.

Also $d(\delta)$ is the smallest function power of D in the generating function

$$\sum_{i=1}^{2^{K-1}} T_i(X, Y, Z), \quad A_{d(\delta)} \text{ is the number of terms of length } \delta \text{ and weight } d(\delta), \text{ and}$$

$B_{d_{free}}$ is the number of nonzero information bits on all weight d_{free} paths. From the previous error expression it is clear that if $d(\delta) > d_{free}$, the second term is negligible compared with the first term, and the additional error probability due to truncation is small compared to the error probability of a standard decoder.

The previous expression can be used to define minimum truncations lengths for various codes. It can also be generalized to other DMCs and the unquantized AWGN channel in a similar way.

Finally a practical rule in the form $\delta \geq 5v$, where v in this case represents the channel's memory, should hold for the value of the truncation length δ in maximum likelihood sequence detection under ISI conditions and PAM transmission. The number of the undetected errors due to truncation is expressed as : (Forney [4])

$$[1 - (1/v + 1)]^\delta$$

2.1.7 Complexity

The complexity of the algorithm is easily estimated, in terms of memory requirements and amount of computations.

i). Memory :

the algorithm requires M storage locations, one for each state, where each location must be capable of storing a metric-value $\Lambda(m)$ and a truncated survivor listing $\sigma(m)$ of δ symbols.

ii). Computation :

In each unit of time the algorithm must take $|E|$ additions, one for each transition, and M comparisons among the $|E|$ results. Thus the amount of storage is proportional to the number of states, and the amount of computations to the number of transitions. With a shift-register model, $M = m^v$ and $|E| = m^{v+1}$, so that the complexity increases exponentially with the length v of the shift-register.

2.1.8 Analysis of Performance.

In many cases, tight upper and lower bounds for error probability can be derived. Even when the VA is not actually implemented, calculation of its performance shows how far the performance of less complex schemes is from ideal, and often suggests simple suboptimum schemes that attain nearly optimal performance. [2].

The key concept in performance analysis is that of an error event. Let σ be the actual state sequence, and δ the state sequence actually chosen by the VA. Over a long time σ and δ will typically diverge and remerge a number of times as illustrated in fig. 2.7. Each distinct separation is called an error event. Error events may in general be of unbounded length if σ is infinite, but the probability of an infinite error event will usually be zero [2].

The importance of error events is that they are probabilistically independent of one another and they allow us to calculate error probability per unit time, which is necessary since usually the probability of any error in MAP estimation of a block of length N goes to 1 as N goes to infinity. Instead we calculate the probability of an error event starting at some given time, given that the starting state is correct, i.e., that an error event is not already in progress at that time.

Given the correct path σ , the set E_k of all possible error events starting at some time k is a treelike trellis which starts at σ_k and each of whose branches ends on the correct path, as illustrated in fig. 2.8, for the trellis of fig. 2.4b. In coding theory this is called the incorrect subset (at time k).



Fig. 2.7 Typical correct path x (heavy line) and estimated path z (lighter line) in the trellis showing three error events.

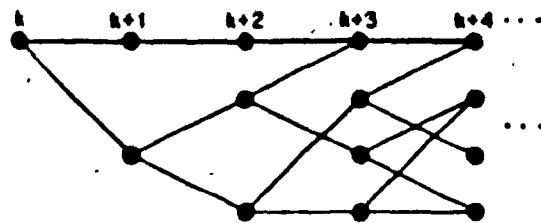


Fig. 2.8 Typical correct path x (heavy line) and time- k incorrect subset for a four state trellis.

The probability of any particular error event is easily calculated; it is simply the probability that the observations will be such that over the time span during which σ is different from σ , σ is more likely than σ . If the error event has length τ , this is simply a two-hypothesis decision problem between two sequences of length τ , and typically has a standard solution.

The probability $P(E_k)$ that any error event in E_k occurs can then upper-bounded by a union bound, i.e., by the sum of the probabilities of all error events in E_k . While this sum may well be infinite, it is typically dominated by one or a few large leading terms representing particularly likely error events, whose sum then forms a good approximation to $P(E_k)$.

On the other hand, a lower bound to error-event probability, again frequently tight, can be obtained by a genie argument. [2]. Take the particular error event that has the greatest probability of all those in E_k , (denoted by $\max P(\text{error event})$). Suppose that a friendly genie tells you that the true state sequence is one of two possibilities: the actual correct path, or the incorrect path corresponding to that error event. Even with this side of information, you will still make an error if the incorrect path is more likely given z , so your probability of error is still no better than the probability of this particular error event. In the absence of the genie, the error probability must be worse still, since one of the strategies we have, given the genie's information, is to ignore it. In summary, the probability of any particular error event is a lower bound to $P(E_k)$.

In conclusion, the probability of any error event starting at time k may be upper-bounded and lower-bounded as follows:

$$\max P(\text{error event}) \leq P(E_k) \leq \max P(\text{error event}) + \text{other terms.}$$

i). Convolutional Codes Error Performance.

The upper and lower bounds of the previous expression are close to each other in the case of convolutional codes.

a. For binary convolutional codes on symmetric memoryless channels the principal result [7] is that $P(E_k)$ is approximately given by

$$P(E_k) \approx N_d 2^{-dD}$$

where d is the free distance, i.e., the minimum Hamming distance of any path in the incorrect subset E_k from the correct path; N_d is the number of these paths; and D is the Bhattacharyya distance

$$D = \log_2 \sum_z P(z|0)^{1/2} P(z|1)^{1/2}$$

where the sum is over all outputs z in the channel output space Z .

b. On Gaussian channels

$$P(E_k) \approx N_d \exp(-dRE_k/N_0)$$

where E_k/N_0 is the signal-to-noise ratio per information bit. The tightness of this bound is confirmed by simulations [8].

ii). *Intersymbol Interference Channels.*

The principal result, for PAM in white Gaussian noise, is that $P(E_k)$ can be tightly bounded as follows :

$$K_L Q(d_{\min}/2\sigma) \leq P(E_k) \leq K_U Q(d_{\min}/2\sigma)$$

where K_L and K_U are small constants, $Q(x)$ is the Gaussian error probability function defined earlier, σ^2 is the noise variance, and d_{\min} is the Euclidean distance between any two distinct signals [4]. This result implies that on most channels intersymbol interference need not lead to any significant degradation in performance [†], which comes as rather a surprise.

For example, with the most common partial response systems, the VA recovers the 3-dB loss sustained by conventional decoders relative to full-response systems [4], [9]. Simple suboptimum processors [4], can do nearly as well.

It seems most likely that the greatest effect of the VA on digital modulation systems will be to reveal those instances in which conventional detection techniques fall significantly short of optimum, and to suggest effective suboptimum methods of closing the gap. PAM channels that cannot be linearly equalized without excessive noise enhancement due to nulls or near nulls in the transmission band are the likeliest candidates for nonlinear techniques of this kind.

[†] It is shown in [4] that in the absence of ISI the error probability $P(E_k)$ can be written $P(E_k) = K_A Q(s_{\min}/2\sigma)$ with $K_A = 2(m-1)/m$ and m the levels of PAM. Then the probability of error of the same system under ISI conditions differs at most by the ratio K_U/K_A from that of an m -level system without ISI. In decibels such a difference is small and goes to zero as SNR goes to infinity.

2.2 Related algorithms

A few other decoding techniques are presented here that have been applied on the topics of convolutional coding and transmission under ISI conditions, supplying alternative views and comparative results to the efficient signaling problem in these areas and the performance of the VA.

2.2.1 Historical Background of Convolutional Codes Decoding [10]

The first practical decoding algorithm for long randomly chosen convolutional codes, called sequential decoding, was proposed by Wozencraft and Reiffen and subsequently refined by Fano. Other types of sequential decoding followed. In 1963, Massey proposed a less efficient but simpler-to-implement decoding technique called 'threshold decoding'. Another type of sequential decoding algorithm called 'stack algorithm' was published in 1966 by Zigangirov and rediscovered by Jellinek.

Viterbi in 1967, introduced a new algorithm particularly effective in decoding convolutional codes of short constraint length. Finally Heller proposed another sequential decoding algorithm under the name 'feedback decoding'.

2.2.2 Sequential and Feedback Decoding [11]

Decoding a $\frac{k}{n}$, L, convolutional code by means of the VA, requires the computation of 2^{kL} metrics at each node of the trellis and the storage of $2^{k(L-1)}$ metrics and $2^{k(L-1)}$ surviving sequences, each of which may be about skL bits long. This

amount of computations is performed independently of the noisy conditions of the channel, resulting to a computational burden that reduce the VA impractical for decoding of convolutional codes with a large constraint length. The main characteristic of the sequential techniques listed below, is a decoding effort adaptation to the noise level.

The Fano sequential decoding algorithm searches for the most probable path through the tree or trellis by examining one path at a time. The increment added to the metric along each branch is proportional to the probability of the received signal for that branch, just as in Viterbi decoding, with the exception that an additional negative constant is added to each branch metric. The value of this constant is selected such that the metric of the correct path will increase on the average while the metric of any incorrect path will decrease on the average. By comparing the metric of a candidate path with a moving (increasing) threshold, Fano's algorithm detects and discards incorrect paths.

The decoder is usually forced to start on the correct path by the transmission of a few known bits of data. Then it proceeds forward from node to node, taking the most probable branch at each node and increasing the threshold is never more than some preselected value, say τ , below the metric. Now suppose that the additive noise (for soft-decision decoding) or demodulation errors When the decoder deviates, due to the noise level, into following an error path that appears more probable than the correct one for a certain length, the observation of the metric's decrement forces a back up situation in which alternative paths through the tree are examined, in an attempt to find another path that exceeds

the threshold τ_0 . If it is successful in finding an alternative path it continues along this path, always selecting the most probable branch at each node. (fig. 2.9) On the other hand, if no path exists that exceeds the threshold τ_0 , the threshold is reduced by an amount τ and the original path is retraced. If the original path does not stay above the new threshold the decoder resumes its backward search for other paths. This procedure is repeated, with the threshold reduced by τ for each repetition until the decoder finds a path that remains above the adjusted threshold.

In comparison with the VA, the Fano sequential decoding exhibits comparable error performance, a significantly larger decoding delay but also reduced storage requirements.

The stack algorithm is another type of sequential decoding, in which a list is maintained of the shortest partial paths found to date, the path on the top of the list is extended and its successors reordered in the list until some path is found that reaches the terminal node, or else decreases without limit (fig. 2.10). That some path will eventually do so is ensured in coding applications by the subtraction of a bias term such that the length of the correct path tends to decrease while that of all incorrect paths tends to increase.

In a comparison of the stack algorithm with the VA, the stack algorithm requires fewer metric computations, but this computational savings is offset to a large extent by the computations involved in reordering the stack after each iteration. When compared with the Fano algorithm, the stack algorithm is computationally simpler since there is no retracing over the same path as is done in

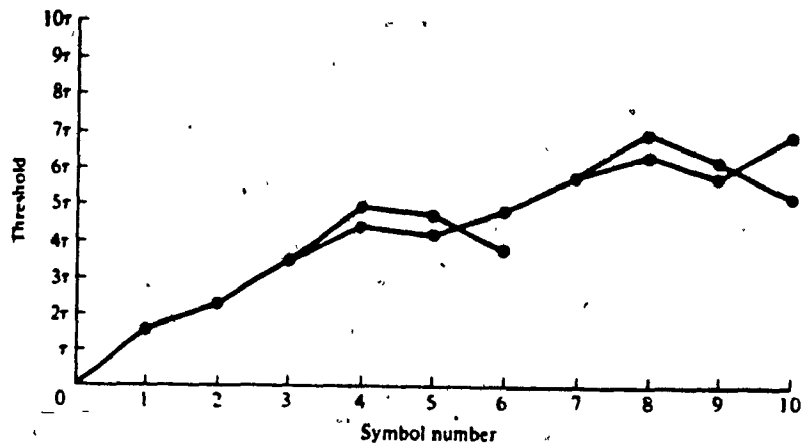
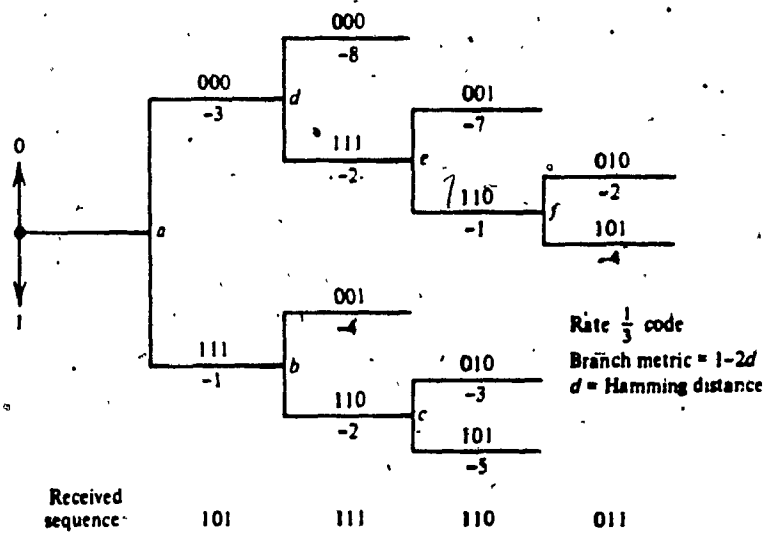


Fig. 2.9 Path search in sequential decoding.



Stack with accumulated path metrics

Step a	Step b	Step c	Step d	Step e	Step f
-1	-2	-3	-2	-1	-2
-3	-3	-3	-3	-3	-3
	-4	-4	-4	-4	-4
		-5	-5	-5	-4
			-6	-7	-5
				-8	-7
					-8

Fig. 2.10 Decoding a rate-1/3 convolutional code via the stack algorithm.

the Fano algorithm. On the other hand, the stack algorithm requires more storage than the Fano algorithm.

Feedback decoding is another technique in which the decoder makes a hard decision on the information bit at stage j based on metrics computed from stage j to $j + m$, where m is a preselected positive integer. Thus the decision on the information bit is either 0 or 1 depending on whether the minimum Hamming distance path which begins at stage j and ends at stage $j + m$ contains a 0 or 1 in the branch emanating from stage j . Once a decision is made on the information bit at stage j only that part of the tree which stems from the bit selected at stage j is kept and the remaining part is discarded. The next step is to extend the part of the tree that has survived to stage $j + 1 + m$ and consider the paths from stage $j + 1$ to $j + 1 + m$ in deciding on the bit at stage $j + 1$. This procedure is repeated at every stage. The parameter m is simply the number of stages in the tree that the decoder looks ahead before making a hard decision, which is usually selected in the range $L \leq m \leq 2L$ significantly smaller than the VA decoding delay ($5L$).

Instead of computing metrics as described above, a feedback decoder for the BSC may be efficiently implemented by computing the syndrome from the received sequence and using a table lookup method for correcting errors. This method is similar to the syndrome decoding technique applied to decoding block codes. For some convolutional codes, the feedback decoder simplifies to a form called a 'majority logic decoder', or a threshold decoder.

2.4.3 Comparison of Sequential and Viterbi Decoding

The main advantage of sequential decoders is that their complexity is relatively independent of constraint length, which can typically be made quite large to provide a very small probability of undetected errors. On the contrary the computational burden of the VA grows exponentially with any linear increment of the constraint length, limiting its use to codes with relatively small values of L .

Long decoding delays, poor performance in error-bursty conditions of the channel, data buffer overflow and necessary storage of several thousands of received data in soft-decoding (a situation that almost prohibits this alternative), are the main disadvantages of sequential decoding techniques in comparison with the respective actual performance of the Viterbi algorithm [8] .

2.4.4 Related Algorithms in the Area of Intersymbol Interference .

In the intersymbol interference literature, many of the alternatives to VA attempt to find optimum nonlinear algorithms using bit-error probability as the optimality criterion [12]-[14] .

The general principle of several of these algorithms is as follows.

Denoting by z the observation, we first calculate the joint probability $P(\sigma_k, z)$ for every state σ_k in the trellis, or alternately $P(\xi_k, z)$ for every transition ξ_k . This is done by observing that

$$P(\sigma_k, z) = P(\sigma_k, z_0^{k-1})P(z_k^N | \sigma_k, z_0^{k-1}) = P(\sigma_k, z_0^{k-1})P(z_k^N | \sigma_k)$$

since, given σ_k , the outputs z_k^N from time k to N are independent of the outputs z_0^{k-1} from time 0 to $k-1$. Similarly

$$\begin{aligned} P(\xi_k, z) &= P(\sigma_k, \sigma_{k+1}, z) \\ &= P(\sigma_k, z_0^{k-1}) P(\sigma_{k+1}, z_k \mid \sigma_k, z_0^{k-1}) P(z_{k+1}^K \mid \sigma_{k+1}, \sigma_k, z_0^k) \\ &= P(\sigma_k, z_0^{k-1}) P(\sigma_{k+1}, z_k \mid \sigma_k) P(z_{k+1}^K \mid \sigma_{k+1}). \end{aligned}$$

Now we note the recursive formula

$$P(\sigma_k, z_0^{k-1}) = \sum_{\sigma_{k-1}} P(\sigma_k, \sigma_{k+1}, z_0^{k-1}) = \sum_{\sigma_{k-1}} P(\sigma_{k-1}, z_0^{k-2}) P(\sigma_k, z_{k-1} \mid \sigma_{k-1})$$

which allows us to calculate the M quantities $P(\sigma_k, z_0^{k-1})$ from the M quantities $P(\sigma_{k-1}, z_0^{k-2})$ with $|E|$ multiplications and additions using the exponential lengths

$$e^{-\lambda(\xi_{k-1})} = P(\sigma_k \mid \sigma_{k-1}) P(z_{k-1} \mid \xi_{k-1})$$

Similarly we have the backward recursion

$$P(z_k^K \mid \sigma_k) = \sum_{\sigma_{k+1}} P(z_k^K, \sigma_{k+1} \mid \sigma_k) = \sum_{\sigma_{k+1}} P(z_k, \sigma_{k+1} \mid \sigma_k) P(z_{k+1}^K \mid \sigma_{k+1})$$

which has similar complexity. Completion of these forward and backward recursions for all nodes allows $P(\sigma_k, z)$ to be calculated for all nodes.

Now, to be specific, let us consider a shift-register process and let $S(u_k)$ be the set of all states σ_{k+1} whose first component is u_k . Then

$$P(u_k, z) = \sum_{\sigma_{k+1} \in S(u_k)} P(\sigma_{k+1}, z)$$

Since $P(u_k, z) = P(u_k \mid z) P(z)$, MAP estimation of u_k reduces to finding the max-

imum of this quantity. Similarly, if we wish to find the MAP estimate of an output y_k , say, then let $S(y_k)$ be the set of all ξ_k that lead to y_k and compute

$$P(y_k, z) = \sum_{\xi_k \in S(y_k)} P(\xi_k, z)$$

Successive computation of the last expression for all k , $1 \leq k \leq N$, leads to MAP estimation of the transmitted sequence u .

The symbol-by-symbol algorithm proposed by Hayes et. al. [1], belongs to the previously discussed class of algorithms which use bit error probability as the optimality criterion. In the sequel, this algorithm which was originally proposed in decoding pulse amplitude modulated signals under ISI conditions is further extended in convolutional codes decoding.

2.3 The Optimal Symbol-by-Symbol Algorithm.

The following decoding algorithm proposed by Hayes et al. in 1982, performs *optimal symbol-by-symbol* detection of a pulse amplitude modulated (PAM) sequence. It exhibits a structure similar to that of the VA, sharing a lot of its advantages such as effective use of dynamic programming principles, parallel structure, straightforwardness of implementation and 'merging' properties that allow decisions to be made prior to reception of the complete sequence. Their critical difference is the optimality criterion; the *symbol-by-symbol* algorithm is focused on minimization of symbol error probability rather than minimization of the probability of error in whole sequence detection.

The symbol-by-symbol algorithm is here considered for optimal decoding of convolutional codes. Following the analysis originally proposed in [1], both the algorithms are presented along with their common derivation. The notation of some critical terms involved in the description of the VA in Sec. 2.3.1, is different from that previously used, primarily for reasons of a unifying treatment of the two decoding algorithms, which implied an explicitly defined common terminology for both the techniques.

2.3.1 Hard-decoding of Convolutional Codes.

We consider the case of decoding convolutionally encoded sequences transmitted over a Binary Symmetric Channel (BSC) - the probability of a bit error in this channel is p for either 0's or 1's.

The transmitted sequence of length N symbols - each of them assuming one of M possible values - is denoted by $\mathbf{a} = (a_1, a_2, \dots, a_N)$, whereas some possible realization of this sequence, say m , is denoted by $\mathbf{a}^{(m)}$, or equivalently by (a_1, a_2, \dots, a_N) . Similarly we define the vector $\mathbf{y}^{(m)}$ as the coded output of the information vector $\mathbf{a}^{(m)}$ and \mathbf{z} the noisy received vector of length nN .

Since each information symbol a_i is encoded into n coded bits, each element y_i of \mathbf{y} represents a subvector of length n , i.e. $y_i = (y_{i1}, y_{i2}, \dots, y_{in})$. Similarly, $z_i = (z_{i1}, z_{i2}, \dots, z_{in})$, since $z_{ij} = y_{ij} \oplus n_{ij}$, where the symbol \oplus represents modulo 2 addition and the noise component n_{ij} is a random variable binomially distributed with p .

A key term that arises in the MAP decision rule of both the decoding algorithms is the probability density function

$$p(\mathbf{z}/\mathbf{a}^{(m)}) = p(\mathbf{z}/\mathbf{y}^{(m)}) \quad (2.1)$$

since there is a one to one correspondence between $\mathbf{a}^{(m)}$ and $\mathbf{y}^{(m)}$. Under the BSC assumption the probability density function of a particular signal being received given that a particular sequence of nN code bits $\mathbf{y}^{(m)}$ was transmitted is

$$p(\mathbf{z}/\mathbf{y}^{(m)}) = \prod_{i=1}^{nN} p(z_i/y_i^{(m)}) = p^{d^{(m)}} (1-p)^{nN-d^{(m)}} \quad (2.2)$$

where $d^{(m)}$ is the Hamming distance between the two vectors \mathbf{z} and $\mathbf{y}^{(m)}$ (i.e.,

$$d^{(m)} = \sum_{i=1}^{nN} z_i \oplus y_i^{(m)}.$$

Since the process evolves in n -bit steps, the previous expression can be written as :

$$p(z/y^{(m)}) = \prod_{i=1}^N p(z_i/y_i^{(m)}) = \prod_{i=1}^N p^{d_i^{(m)}} (1-p)^{n-d_i^{(m)}} = \prod_{i=1}^N \left(\frac{p}{1-p} \right)^{d_i^{(m)}} (1-p)^n \quad (2.3)$$

where N is the number of branches of the m^{th} path, and $z_i, y_i^{(m)}$ represent subsequences of length n bits, having Hamming distance $d_i^{(m)}$ ($d_i^{(m)} = \sum_{j=1}^n z_{ij} \oplus y_{ij}^{(m)}$).

Denoting by α the ratio $\frac{p}{1-p}$, the previous expression can be written as :

$$p(z/y^{(m)}) = \prod_{i=1}^N \left(\frac{p}{1-p} \right)^{d_i^{(m)}} (1-p)^n = \prod_{i=1}^N \alpha^{\sum_{j=1}^n z_{ij} \oplus y_{ij}^{(m)}} (1-p)^n$$

Noting also that $(1-p)^n$ is just a weighting factor independent of $y_{ij}^{(m)}$ we can write :

$$p(z/y^{(m)}) = c \prod_{i=1}^N \alpha^{\sum_{j=1}^n z_{ij} \oplus y_{ij}^{(m)}} = c \alpha^{\sum_{i=1}^N \sum_{j=1}^n z_{ij} \oplus y_{ij}^{(m)}} \quad (2.4)$$

A set of state vectors σ_k is defined as

$$\sigma_k = \{ a_{k-L+1}, a_{k-L+2}, \dots, a_k \} \quad k = 1, 2, \dots, N. \quad (2.5)$$

where L is the constraint length of the code. Also a sequence of state vectors up to and including the state k is denoted by S^k . Thus

$$S^k = \{ \sigma_1, \sigma_2, \dots, \sigma_k \} = \{ a_1, a_2, \dots, a_k \} \quad k \leq N.$$

Similarly, we define the sequence of noisy symbols

$$z^k = \{ z_1, z_2, \dots, z_k \} \quad k \leq N.$$

where the elements $z_i = \{ z_{i1}, \dots, z_{in} \}$. The same notation is again used for the notation of the sequences a^k and \hat{a}^k . Then, with reference to the exponent in eq. (2.4), we define:

$$U(z^k, S^k) = \sum_{i=1}^k \sum_{j=1}^n z_{ij} \oplus y_{ij} \quad (2.6a)$$

$$V(z_k, \sigma_{k-1}, \sigma_k) = \sum_{j=1}^n z_{kj} \oplus y_{kj} \quad (2.6b)$$

and consequently

$$U(z^k, S^k) = U(z^{k-1}, S^{k-1}) + V(z_k, \sigma_{k-1}, \sigma_k) \quad (2.7)$$

Then eq. (2.4) becomes:

$$p(z/y^{(m)}) = c \alpha^{\sum_{i=1}^N \sum_{j=1}^n z_{ij} \oplus y_{ij}^{(m)}} = c \alpha^{U(z^N, S^N)} \quad (2.8)$$

or equivalently, since there is a one-to-one correspondence between information and encoded sequences:

$$\begin{aligned} p(z \mid a_1 = \hat{a}_1, a_2 = \hat{a}_2, \dots, a_N = \hat{a}_N) \\ = c \alpha^{U(z^N, S^N)} = c \alpha^{[U(z^{N-1}, S^{N-1}) + V(z_N, \sigma_{N-1}, \sigma_N)]} \end{aligned} \quad (2.9)$$

where c is a constant.

where c is a constant.

The previous analysis concerned the transmission over a BSC. Similar is the derivation of the respective expressions concerning the transmission over an Additive White Gaussian Noise (AWGN) channel. Substituting eq.(2) by :

$$p(z/y^{(m)}) = \prod_{i=1}^{nN} \frac{1}{\sqrt{2\pi\sigma}} e^{-[z_i - 2aE(2y_i^{(m)} - 1)]^2 / 2\sigma^2}$$

with σ the variance of the white noise and E the signal energy, the rest of the previous analysis remains the same.

2.3.2 Optimum Sequence Detection.

When optimum sequence detection is attempted we want to find the most probable transmitted sequence that resulted in a particular received one. That is, given the received sequence z we attempt to find the information sequence $\hat{a} = (\hat{a}_1, \hat{a}_2, \dots, \hat{a}_N)$ that maximizes the conditional probability

$$p(a^N = \hat{a}^N / z) \quad (2.10)$$

Since the transmitted signals are independent and identically distributed we have

$$\max_{\hat{a}^N} p(a^N = \hat{a}^N / z) = c \max_{\hat{a}^N} \alpha^{[U(z^N, S^N)]} \quad (2.11)$$

and taking the logarithm of the previous expression we get :

$$\max_{\mathbf{a}^N} \ln p(\mathbf{a}^N = \mathbf{a}^N / \mathbf{z}) = \max_{\mathbf{a}^N} \{ [U(\mathbf{z}^N, \mathbf{S}^N)] \ln \alpha + \ln c \}$$

Noting again that $\ln \alpha, \ln c$ are common terms to all possibly transmitted sequences, the problem of optimum sequence detection comes down to maximizing $U(\mathbf{z}^N, \mathbf{S}^N)$ over all transmitted sequences. The recurrent form of this term (eq. (2.7)), leads to the following expression :

$$\begin{aligned} \max_{\mathbf{a}^N} U(\mathbf{z}^N, \mathbf{S}^N) &= \max_{\sigma_1, \dots, \sigma_N} [U(\mathbf{z}^{N-1}, \mathbf{S}^{N-1}) + V(\mathbf{z}_N, \sigma_{N-1}, \sigma_N)] \\ &= \max_{\mathbf{a}^N} \max_{\sigma_1, \dots, \sigma_{N-1} | \sigma_N} [U(\mathbf{z}^{N-1}, \mathbf{S}^{N-1}) + V(\mathbf{z}_N, \sigma_{N-1}, \sigma_N)] \end{aligned}$$

where the notation $\sigma_1, \dots, \sigma_{N-1} | \sigma_N$ means that σ_N is held fixed while $\sigma_1, \dots, \sigma_{N-1}$ is varied. Continuing, we have

$$\begin{aligned} F(\sigma_N) &= \max_{\sigma_1, \dots, \sigma_N} [U(\mathbf{z}^{N-1}, \mathbf{S}^{N-1}) + V(\mathbf{z}_N, \sigma_{N-1}, \sigma_N)] = \\ &= \max_{\sigma_{N-1} | \sigma_N} \max_{\sigma_1, \dots, \sigma_{N-2} | \sigma_{N-1}, \sigma_N} [U(\mathbf{z}^{N-1}, \mathbf{S}^{N-1}) + V(\mathbf{z}_N, \sigma_{N-1}, \sigma_N)] \end{aligned}$$

Two key observations allow us to proceed :

- i) If σ_{N-1} and σ_N are fixed, $V(\mathbf{z}_N, \sigma_{N-1}, \sigma_N)$ is independent of $\sigma_1, \dots, \sigma_{N-2}$ and
- ii) $U(\mathbf{z}^{N-1}, \mathbf{S}^{N-1})$ is conditionally independent of σ_N , given σ_{N-1} . We have then

$$F(\sigma_N) = \max_{\sigma_{N-1} | \sigma_N} [V(\mathbf{z}_N, \sigma_{N-1}, \sigma_N)] + \max_{\sigma_1, \dots, \sigma_{N-2} | \sigma_{N-1}} U(\mathbf{z}^{N-1}, \mathbf{S}^{N-1})$$

$$= \max_{\sigma_{N-1} | \sigma_N} [V(z_N, \sigma_{N-1}, \sigma_N) + F(\sigma_{N-1})]. \quad (2.12)$$

Noting the recurrent form of the previous expression we can write

$$F(\sigma_k) = \max_{\sigma_{k-1} | \sigma_k} [V(z_k, \sigma_{k-1}, \sigma_k) + F(\sigma_{k-1})]. \quad (2.13)$$

By repeated application of eq.(13), the optimum sequence can be found. At each step of the calculation, the path leading to a particular value of a state is preserved. Note also that since the state σ_k was previously defined by the values of $L - 1$ symbols (eq. (2.5)) and each of them may assume one of M possible values, we generally have M^{L-1} possible realizations of σ_k (this number is reduced during the $L - 1$ initial and final states of the process). Finally, when the transmission of the whole sequence is completed, M survivors are compared at the last stage and the one exposing the maximum value of $U(z^N, S^N)$ is selected as the optimum sequence.

2.3.3 Optimum Symbol-by-Symbol Detection .

In optimal *symbol-by-symbol* detection, the whole received sequence is used in the detection of each symbol separately. The actually transmitted sequence a consists of N symbols; any one of them is denoted by a_ω , where $1 \leq \omega \leq N$. Maximum *a posteriori* estimation of this symbol consists of finding the most likely value of a_ω , denoted by \hat{a}_ω , given the received sequence z . The likelihood of each possible value of a_ω is expressed by the conditional probability : $p(a_\omega = \hat{a}_\omega | z)$. The specific value \hat{a}_ω that maximizes the previous expression

constitutes the optimal decision on the actually transmitted value of a_ω .

In order to proceed with the derivation of a tractable expression for this likelihood, we define a subset of the set of states (see eq. (2.5)). Let $\bar{\sigma}_{jl}; j = 1, 2, \dots, N; l = 1, 2, \dots, M$, denote the same set of states as defined previously except that \hat{a}_ω is set equal to one of its M possible values, i.e., $\hat{a}_\omega = a_l$. Thus $\bar{\sigma}_{jl} = \sigma_j$, for $j \leq \omega$ and $j \geq \omega + L - 1$. For $\omega \leq j < \omega + L - 1$, $\bar{\sigma}_{jl}$ has M possible values depending on the M possible values of \hat{a}_ω .

We also define, similarly to eq. (2.5a),

$$S_l^k = \{ \bar{\sigma}_{1,l}, \bar{\sigma}_{2,l}, \dots, \bar{\sigma}_{kl} \}$$

Then

$$Pr(a_\omega = \hat{a}_\omega | z) =$$

$$\sum_{S_l^N} \frac{Pr(a_1 = \hat{a}_1, \dots, a_\omega = \hat{a}_\omega, \dots, a_N = \hat{a}_N) p(z | a_1 = \hat{a}_1, \dots, a_\omega = \hat{a}_\omega, \dots, a_N = \hat{a}_N)}{p(z)}$$

Noting that $p(a_1 = \hat{a}_1, \dots, a_\omega = \hat{a}_\omega, \dots, a_N = \hat{a}_N) = M^{-N}$ (since all the sequences are equiprobable) and $p(z)$ is independent of \hat{a}_ω , the sufficient statistic for the previous probability is reduced to:

$$\sum_{(a_1, a_2, \dots, a_l, \dots, a_N)} p(z | a_1 = \hat{a}_1, \dots, a_\omega = \hat{a}_\omega, \dots, a_N = \hat{a}_N)$$

Combining eq.(9) and (11) we have

$$\max_{\hat{a}_\omega} p(a_\omega = \hat{a}_\omega | z) = c \max_{\hat{a}_\omega} \sum_{\sigma_{1l}, \dots, \sigma_{Nl}} \alpha^{U(z^N, S^N)} \quad (2.14)$$

In order to calculate the summation in the last expression, we use a similar recurrence to that used previously (eq.(9)) :

$$\sum_{\sigma_{1l}, \dots, \sigma_{Nl}} \alpha^{U(z^N, S^N)} = \sum_{\sigma_{Nl}} \left(\sum_{\sigma_{N-1,l} | \sigma_{Nl}} \left(\max_{\sigma_{1l}, \dots, \sigma_{N-2,l} | \sigma_{N-1,l}, \sigma_{Nl}} \alpha^{U(z^{N-1}, S^{N-1}) + V(z_N, \sigma_{N-1,l}, \sigma_{Nl})} \right) \right)$$

where

$$\sum_{\sigma_{N-1,l} | \sigma_{Nl}}$$

denotes summation over all realizations of the state $\sigma_{N-1,l}$ holding state σ_{Nl} fixed.

Similarly,

$$\sum_{\sigma_{1l}, \dots, \sigma_{N-2,l} | \sigma_{N-1,l}, \sigma_{Nl}}$$

denotes summation over $\sigma_{1l}, \dots, \sigma_{N-2,l}$ holding $\sigma_{N-1,l}, \sigma_{Nl}$ fixed. By the same line of reasoning which led to eq.(12) we have

$$\sum_{\sigma_{1l}, \dots, \sigma_{Nl}} \alpha^{U(z^N, S^N)} = \sum_{\sigma_{Nl}} \left(\sum_{\sigma_{N-1,l} | \sigma_{Nl}} \alpha^{V(z_N, \sigma_{N-1,l}, \sigma_{Nl})} \left(\sum_{\sigma_{1l}, \dots, \sigma_{N-2,l} | \sigma_{N-1,l}, \sigma_{Nl}} \alpha^{U(z^{N-1}, S^{N-1})} \right) \right). \quad (2.15)$$

Defining the expression

$$G(\sigma_{kl}, \hat{a}_\omega) = \sum_{\sigma_{1l}, \dots, \sigma_{k-1,l} | \sigma_{kl}} \alpha^{U(z^k, S^k)} \quad k = 1, \dots, N$$

we can write

$$G(\sigma_k, a_\omega) = \sum_{\sigma_{k-1,l} | \sigma_k} \alpha^{V(x_k, \sigma_{k-1,l}, \sigma_k)} G(\sigma_{k-1,l}, a_\omega) \quad k = 1, \dots, N$$

2.3.4 Formal description of the algorithm.

The steps required to find the optimum value of a_ω can be summarized as follows.

- i) Compute for each $\sigma_{m,l}$ the quantity $G(\sigma_{1,l}, a_\omega)$.
- ii) Using the iterative relationship in eq.(16), compute in succession the quantities $G(\sigma_{2,l}, a_\omega)$, $G(\sigma_{3,l}, a_\omega)$, \dots $G(\sigma_N, a_\omega)$.
- iii) Sum $G(\sigma_N, a_\omega)$ over all states σ_N .
- iv) Repeat the previous computations for each of the M possible values of a_ω and choose a_ω which produces a maximum.

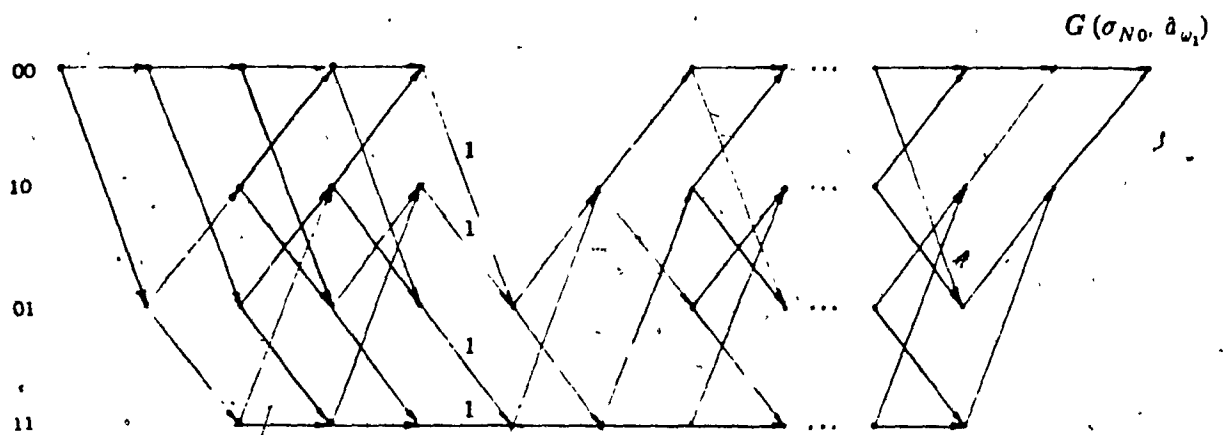
Optimal symbol-by-symbol decoding of the whole sequence is performed by repeating the previous steps for all ω , $1 \leq \omega \leq N$.

As in optimum sequence detection, the finite memory of the channel allows us to define a suitable set of states thereby avoiding complexity that grows exponentially with the length of the transmitted sequence. However, the foregoing calculations must be repeated for each of the N transmitted symbols, implying an N^2 growth in complexity.

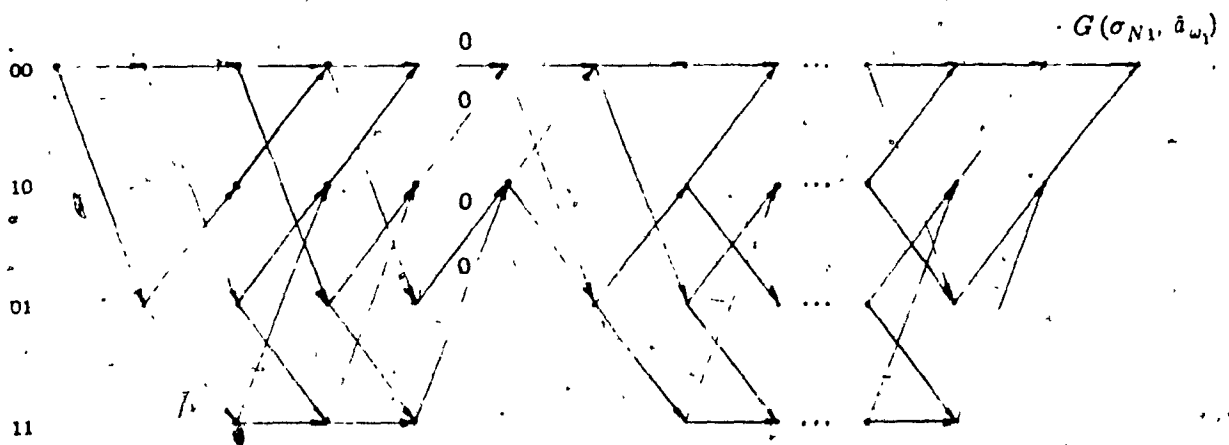
There is a degree of commonality in the calculations for each of the symbols,

which may reduce complexity to some extent.

Consider, for example, the detection of symbols a_{ω_1} and a_{ω_2} where $1 \leq \omega_1 \leq \omega_2 \leq N$. For $j < \omega_1$, the state $\bar{\sigma}_{ji}$ is independent of a_{ω_1} and a_{ω_2} and we have $G(\bar{\sigma}_{ji}, \bar{a}_{\omega_1}) = G(\bar{\sigma}_{ji}, \bar{a}_{\omega_2})$. Thus, in order to detect the symbols a_{ω_1} , $a_{\omega_1+1}, \dots, a_N$, L^m values of $G(\bar{\sigma}_{ji}, \bar{a}_{\omega_1})$ need to be computed for the states $\bar{\sigma}_{ji}$, $j = 1, 2, \dots, \omega_1-1$. When at the ω_1^{th} step, a distinction must be made among the possible values of \bar{a}_{ω_1} . Until a decision is made on \bar{a}_{ω_1} , the quantities $G(\bar{\sigma}_{ji}, \bar{a}_{\omega_1})$ must be computed for all possible values of \bar{a}_{ω_1} . This distinction is depicted in fig. 2.11, for the rate-1/3 $L = 3$ systematic convolutional code of fig. 1.6. In carrying out the computations for the successive states, we carry along $G(\bar{\sigma}_{ji}, \bar{a}_N)$ which is used in the calculations of $G(\bar{\sigma}_{ji}, \bar{a}_k)$; $j, k < N$. This commonality reduces the complexity of computation by one-half.



(a)



(b)

Fig. 2.11 Optimal symbol-by-symbol decoding of the ω_1^A symbol (here $\omega_1^A = 4$). (a) $a_4 = 1$ (b) $a_4 = 0$

In both the decoding techniques we need to compute the likelihoods of each possibly transmitted sequence, in order to decide which sequence or which symbols were actually transmitted. (see eq. (2.11) for optimal sequence detection and eq. (2.14) for optimal symbol-by-symbol detection). Since an information sequence can assume a great number of symbols, a 'brute force' approach to these computations soon becomes impossible. The key observation that minimizes the decoding complexity is the fact that any coded sequence, being the product of a finite-state shift-register machine, can assume, at any time k , one of a finite number of outputs (depending on the certain state of the machine), and merges with other sequences (or 'paths') by exhibiting $L-1$ consecutive identical information symbols at the preencoding level. The previous property combined with the memoryless characteristics of the channel, allows at any time k ,

- 1). the selection of the minimum distance path from a set of merging competitors at any realization of the state σ_k , when Viterbi decoding is performed,
- 2). the summation of the likelihoods of the merging paths at any realization of the state σ_k , when symbol-by symbol decoding is performed,

In both cases independently of the rest of the received sequence.

2.3.5 Merges and Complexity Requirements

"Merges" describe a randomly occurring phenomenon in optimum sequence detection, in which all of the M^{L-1} survivors at state k exhibit the same history, in terms of states succession, at a certain depth δ . Since from eq. (2.13) is assured that the optimum sequence contains one of the survivors at state k , the common

history can be put out as the algorithm's decision at time k . When the depth δ is chosen large enough the probability of survivors' disagreement at time $k-\delta$ is very small and decisions at time k concerning the initial $k-\delta$ symbols are nearly optimum.

An analogous phenomenon is encountered in optimal symbol by symbol decoding. Let $\{a_1, a_2, \dots, a_M\}$ denote the values that a symbol $a_\omega; 1 \leq \omega \leq N$ may assume. As indicated in eq.(16), it is decided that $\hat{a}_\omega = a_l$ if

$$\sum_{\sigma_{Nl}} G(\sigma_{Nl}, \hat{a}_\omega) |_{\hat{a}_\omega = a_l} \geq \sum_{\sigma_{Nl}} G(\sigma_{Nl}, \hat{a}_\omega) |_{\hat{a}_\omega \neq a_l} \quad (2.16)$$

Now suppose that for a particular j such that $j \geq L-1 + \omega$

$$G(\sigma_{jl}, \hat{a}_\omega) |_{\hat{a}_\omega = a_l} \geq G(\sigma_{jl}, \hat{a}_\omega) |_{\hat{a}_\omega \neq a_l} \text{ for every } \sigma_{jl} \quad (2.17)$$

From eq.(15) it follows that for every $k \geq j$

$$G(\sigma_{kl}, \hat{a}_\omega) |_{\hat{a}_\omega = a_l} \geq G(\sigma_{kl}, \hat{a}_\omega) |_{\hat{a}_\omega \neq a_l} \text{ for every } \sigma_{kl} \quad (2.18)$$

Consequently eq.(17) holds.

Thus, if the relationship in eq.(17) holds, then it is not necessary to compute $G(\sigma_{kl}, \hat{a}_\omega); k > j$. The decision $\hat{a}_\omega = a_l$ can be made at time j . In analogy with the Viterbi algorithm, a truncation depth of δL was adopted here as the decoding delay for a symbol's estimation.

The main two expressions concerning the complexity of the two decoders under consideration are reported in [1]. Their interpretation in the present case

is straightforward. The system's memory is $L-1$, a symbol can assume M different values and A_V represents the number of additions necessary to compute $V(z_k, \sigma_{k-1}, \sigma_k)$ for a single pair of states. Then the amount of equivalent additions required for the detection of a single symbol by a Viterbi decoder is :

$$S_1 = M^{L-1} (MA_V + 2M - 1) \quad (2.19)$$

and the storage requirement is

$$B_1 = M^{L-1} (R_1 + D_1 \log_2 M) \text{ bits.}$$

where R_1 is the number of bits necessary to store $F(\sigma_k)$ and D_1 represents the decoding delay.

When symbol-by-symbol decoding is assumed, the amount of additions required for a symbol's detection is :

$$S_2 = M^{L-1} (MA_V + PM + EM + M - 1) (1 + D_2 M) \quad (2.20)$$

where P represents the number of additions equivalent to multiplication, E is the number of additions equivalent to exponentiation and D_2 is the decoding delay in bits.

Also, assuming that R_2 bits are required to store the quantity $G(\sigma_{j1}, a_k)$ with sufficient accuracy, the overall storage requirements for this decoder is :

$$B_2 = L^m R_2 [1 + M_2 L].$$

2.4 On the problem of ML sequence detection of convolutional codes transmitted over a channel with ISI.

2.4.1. Nonencoded Transmission.

In the general case, in which the information sequence of length N is transmitted in a band-limited channel via PAM (Pulse Amplitude Modulation) and is not encoded (fig. 2.12), the received signal has the following form [1]:

$$y(t) = \sum_{j=1}^N a_j h(t - jT) + n(t) \quad 0 \leq t \leq \tau, \quad NT \leq \tau \leq \infty \quad (2.21)$$

and

$$Pr[y(t), 0 \leq t \leq T \mid a_1 = a_1, a_2 = a_2, \dots, a_N = a_N] =$$

$$K \exp \left[\left(-\frac{1}{2N} \right) \int_0^T \left[y(t) - \sum_{i=1}^N a_i h(t - iT) \right]^2 dt \right]. \quad (2.22)$$

Maximization of this quantity, reduces to minimization of:

$$\int_0^T \left[y(t) - \sum_{i=1}^N a_i h(t - iT) \right]^2 dt \quad (2.23)$$

2.4.2 Encoded Transmission

In our case (fig. 2.13) the equivalent of the noiseless sequence $\{a_i\}$ in eq. (2.21), denoted by $\{b_i\}$, consists of N/n successive n -tuple subsequences, the discrete outputs of the convolutional encoder modulated by a pulse amplitude modulation scheme (PAM).

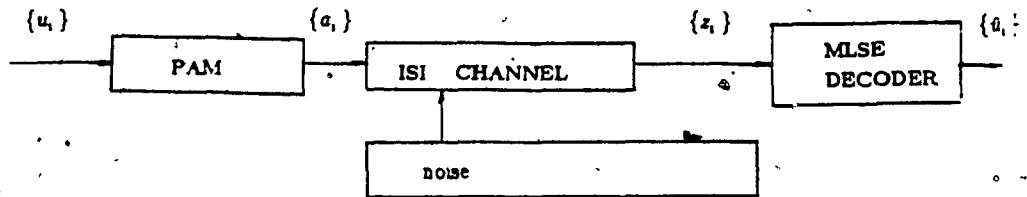


Fig. 2.12 General transmission/reception scheme of PAM signals

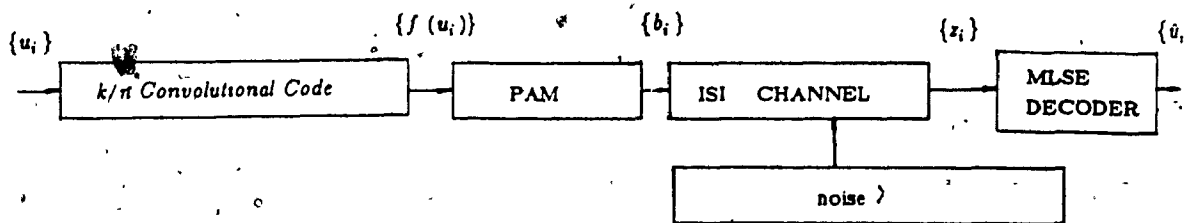


Fig. 2.13 General transmission/reception scheme of PAM and convolutionally-encoded signals.

The formulation of eq. (2.21) as :

$$y(t) = \sum_{j=1}^{N/n} \sum_{i=1}^n a_{(j-1)n+i} h(t - ((j-1)n + i)T) + n(t) = \sum_{j=1}^{N/n} b_j h_j + n(t) \quad (2.24)$$

by means of the vectors, $b_j = [a_{(j-1)n+1}, a_{(j-1)n+2}, \dots, a_{jn}]$ and

$$h_j = \begin{bmatrix} h(t - ((j-1)n + 1)T) \\ \vdots \\ h(t - jnT) \end{bmatrix}$$

shows the sequence's separation into distinct codewords of length n .

Denoting by u the information sequence and assuming a convolutional code with rate $\frac{k}{n}$ (k bits input, n bits output), the encoding procedure can be described by $f(u_j)$ ($f(\cdot)$ represents the coding transform of the information) and subsequently, the vector b_j is the PAM transform of $f(u_j)$:

$$b_j = \text{PAM}[f(u_j)] = [a_{(j-1)n+1}, a_{(j-1)n+2}, \dots, a_{jn}] \quad (2.25)$$

Clearly the sequence $\{a_i\}$ does not have the property of Independence as in [1] where it represents the information sequence, pulse amplitude modulated, since the encoding procedure introduces correlation between successive information symbols allowing certain patterns of a_i symbols to exist. Using this notation eq. (2.22) becomes :

$$\Pr[y(t), 0 \leq t \leq T \mid b_1 = \delta_1, b_2 = \delta_2, \dots, b_N = \delta_N] = \quad (2.26)$$

$$K \exp \left[-\frac{1}{2N} \int_0^T \left[y(t) - \sum_{j=1}^{N/n} \delta_j h_j \right]^2 dt \right].$$

Denoting the ratio N/n by N' , and following the same procedure described in [1], maximization of this quantity reduces to minimization of :

$$\int_0^T (y^2(t) - 2 \sum_{j=1}^{N'} \delta_j y(t) h_j + \sum_{j=1}^{N'} \sum_{i=1}^{N'} \delta_j h_j \delta_i h_i) dt = \quad (2.27a)$$

$$\begin{aligned} & \int_0^T y^2(t) dt - 2 \sum_{j=1}^{N'} \sum_{i=1}^n z_{(j-1)n+i} \delta_{(j-1)n+i} + \\ & + \sum_{j=1}^{N'} \sum_{i=1}^{N'} \sum_{f=1}^n \sum_{l=1}^n \delta_{(j-1)n+f} h(t - ((j-1)n + f)T) \delta_{(i-1)n+l} h(t - ((i-1)n + l)T) \end{aligned} \quad (2.27b)$$

where

$$z_j = \int_0^T y(t) h_j dt \quad (2.28)$$

and

$$z_{(j-1)n+i} = \int_0^T y(t) h(t - ((j-1)n + i)T) dt$$

represents the i^{th} element of the vector z_j . Defining also :

$$\begin{aligned} r_{|(i-1)n+f| - |(j-1)n+l|} &= r_{|(i-j)n+f-l|} = \\ &= \int_0^T h(t - ((j-1)n + f)T) h(t - ((i-1)n + l)T) dt \end{aligned}$$

where $r_{|(i-j)n+f-l|} = 0$ for $|(i-j)n+f-l| > m$ and m : the channel's memory, we get :

$$\int_0^T y^2(t) dt - 2 \sum_{j=1}^{N'} \sum_{i=1}^n z_{(j-1)n+i} \delta_{(j-1)n+i} + \sum_{j=1}^{N'} \sum_{i=1}^{N'} \sum_{f=1}^n \sum_{l=1}^n \delta_{(j-1)n+f} \delta_{(i-1)n+l} r_{|(i-j)n+f-l|}$$

The quantity $\int_0^T y^2(t) dt$, is a common term of the 'distance', of each

competitor- sequence $\{d_i\}$, and the sufficient statistic can be reduced to the computation of the following expression.

$$2 \sum_{j=1}^{N'} \sum_{n=1}^N z_{(j-1)N+i} d_{(j-1)N+i} - \sum_{j=1}^{N'} \sum_{i=1}^{N'} \sum_{n=1}^N \sum_{l=1}^N d_{(j-1)N+i} d_{(i-1)N+l} r_{[(i-j)N+j-l]} \quad (2.29)$$

or equivalently as in eq. (2.27a)

$$2 \sum_{j=1}^{N'} \delta_j y(t) h_j + \sum_{j=1}^{N'} \sum_{i=1}^{N'} \delta_j h_j \delta_i h_i \quad (2.29a)$$

Adopting at this point the previously discussed "whitening filter's" approach (which shortens the intersymbol interference from two-sided and future to one-sided involving only the past) for reasons of simplicity, since no loss of optimality is introduced by such an assumption, the output of such a filter z_j can be written as :

$$z_{(j-1)N+i} = \sum_{l=0}^{m'-1} d_{(j-1)N+i-l} r_l + n_{(j-1)N+i}$$

The process of detection.

Now, in order to avoid unnecessary complexity, we continue regarding a subsequence of n symbols as the vector b_j , and define an auxiliary channel's block-memory m' . This quantity expresses the contribution of the last m' vectors b_{j-i} , for $i \leq m'$, to the received value of the vector b_m . In respect with the definition of the channel memory as :

$$r_{i-j} = \int_0^T h(t-iT)h(t-jT) dt$$

and $r_{i-j} = 0$ for $|i - j| > m$, we define it as

$$r'_{i-j} = \int_0^T h(t - inT)h(t - jnT) dt \quad (2.30a)$$

and $r'_{i-j} = 0$ for $|i - j| > m'$.

In terms of the channel memory m and the length of a codeword n , it can also be defined as :

$$m' = \left\lceil \frac{m}{n} \right\rceil \quad (2.30b)$$

where $\lceil \cdot \rceil$ denote the minimum integer with value greater or equal to the ratio m/n . The necessity of the above definitions is obvious, since we deal with blocks of symbols (codewords) instead of independent symbols, as in the case of the uncoded transmission.

Following the same procedure with that in [1], we define a set of state vectors (see eq.(5), [1]) as :

$$\sigma_k = \{ b_{k-m'+1}, b_{k-m'+2}, \dots, b_k \} \quad k = m', m' + 1, \dots, N' \quad (2.31)$$

and S^k (the sequence of state vectors up to and including the state at time kT).

$$S^k = \{ \sigma_{m'}, \sigma_{m'+1}, \dots, \sigma_k \} = \{ b_1, b_2, \dots, b_k \} \quad k \leq N'$$

Similarly

$$z^k = \{ z_1, z_2, \dots, z_k \} \quad k \leq N'$$

where the elements of z^k were previously defined in eq. (2.28).

Then :

$$U(z^k, S^k) =$$

$$2 \sum_{j=1}^k \sum_{i=1}^n z_{(j-1)n+i} \hat{d}_{(j-1)n+i} - \sum_{j=1}^k \sum_{i=1}^k \sum_{f=1}^n \sum_{l=1}^n \hat{d}_{(j-1)n+f} \hat{d}_{(i-1)n+l} r_{[(i-f)n+f-l]}$$

and representing

$$V(z_k, \sigma_{k-1}, \sigma_k) =$$

$$2 \sum_{i=1}^n z_{(k-1)n+i} \hat{d}_{(k-1)n+i} - 2 \sum_{i=1}^n \hat{d}_{(k-1)n+i} \cdot \sum_{j=1}^m \hat{d}_{(i-1)n+j} r_j - \sum_{i=1}^n \hat{d}_{(k-1)n+i}^2 r_0$$

follows that

$$U(z^k, S^k) = U(z^{k-1}, S^{k-1}) + V(z_k, \sigma_{k-1}, \sigma_k) \quad (2.32)$$

The quantity $V(z_k, \sigma_{k-1}, \sigma_k)$ represents the part of the metric of a sequence due to transition between two successive states. It can be farther analyzed as :

$$V(z_k, \sigma_{k-1}, \sigma_k) =$$

$$+ 2 z_{(k-1)n+1} \hat{d}_{(k-1)n+1} - 2 \hat{d}_{(k-1)n+1} \sum_{j=1}^m \hat{d}_{(k-1)n+1-j} r_j - \hat{d}_{(k-1)n+1}^2 r_0$$

$$+ 2 z_{(k-1)n+2} \hat{d}_{(k-1)n+2} - 2 \hat{d}_{(k-1)n+2} \sum_{j=1}^m \hat{d}_{(k-1)n+2-j} r_j - \hat{d}_{(k-1)n+2}^2 r_0$$

$$\dots$$

$$+ 2 z_{kn} \hat{d}_{kn} - 2 \hat{d}_{kn} \sum_{j=1}^m \hat{d}_{kn-j} r_j - \hat{d}_{kn}^2 r_0$$

or

$$V(z_k, \sigma_{k-1}, \sigma_k) = \sum_{j=1}^n V(z_{kj}, \sigma_{k-1}, \sigma_k) \quad (2.33)$$

Noting that $U(z^{N'}, S^{N'})$ equals the sufficient statistic defined in eq. (2.29), and also the Markovian nature of the process, the closest path can recursively be found from the following expression

$$F(\sigma_k) = \max_{\sigma_{k-1}/\sigma_k} [V(z_k, \sigma_{k-1}, \sigma_k) + F(\sigma_{k-1})]. \quad (2.34)$$

According to this, all that is needed for the ML sequence estimation, is the computation of $V(z_k, \sigma_{k-1}, \sigma_k)$ for $k = m', m' + 1, \dots, N'$.

2.4.3 Implementation of the Decoder.

I. Overall Memory of the System.

It can be clearly seen from eq. (2.33) or its equivalent expression

$$\begin{aligned} V(z_k, \sigma_{k-1}, \sigma_k) = & \\ & 2 z_{(k-1)n+1} d_{(k-1)n+1} - 2 d_{(k-1)n+1} \sum_{j=1}^m d_{(k-1)n+1-j} r_j - d_{(k-1)n+1}^2 r_0 \\ & + 2 z_{(k-1)n+2} d_{(k-1)n+2} - 2 d_{(k-1)n+2} \sum_{j=1}^m d_{(k-1)n+2-j} r_j - d_{(k-1)n+2}^2 r_0 \\ & \dots \\ & + 2 z_{kn} d_{kn} - 2 d_{kn} \sum_{j=1}^m d_{kn-j} r_j - d_{kn}^2 r_0 \end{aligned}$$

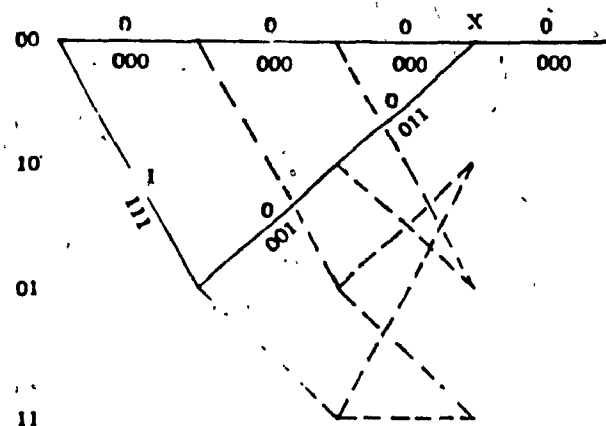
that the values of m symbols of the previous m' vectors b_j are needed for the

computation of the transition metric $V(z_k, \sigma_{k-1}, \sigma_k)$. A decoding technique based on a trellis-description of the formation of each possible path, should require an overall memory of $L - 1 + m'$, where L represents the constraint length of the code and m' the channel's block memory, previously defined in eq. (2.30). Note that the overall system's memory is not $L - 1 + m$, because of the fact that the process evolves with steps of length n , thus the normalized value of m is the appropriate measure of correlation.

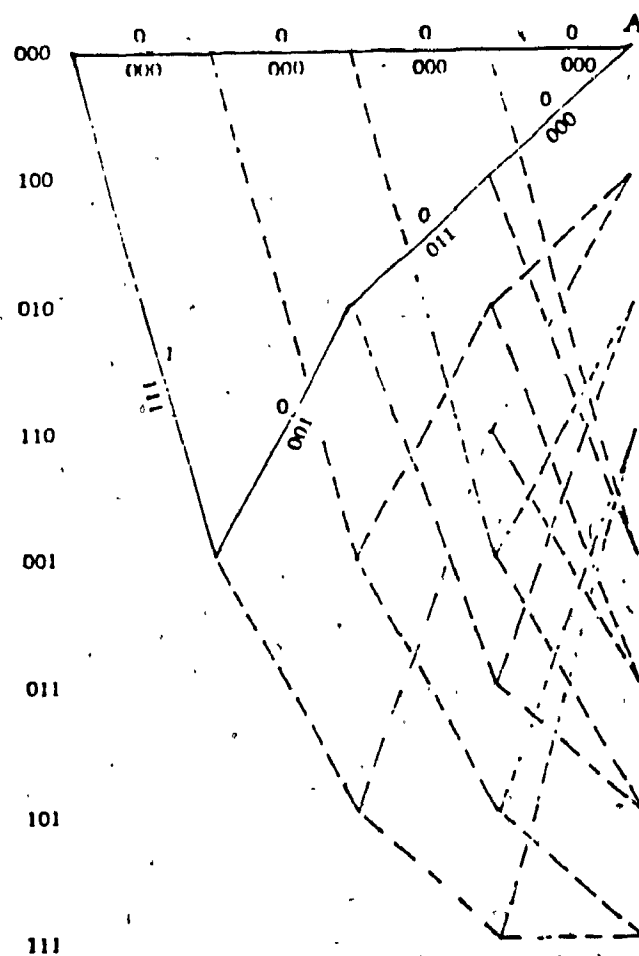
Example : Suppose the use of a convolutional code with rate $1/3$ and constraint length $L = 3$. A trellis-based decoding technique for this code is shown in fig. 2.14a. In the absence of ISI, a decision between the branches (000) and (100) can be taken at point (X) regardless of the rest of the received sequence.

This is not generally the case when ISI is involved. Let's assume again the same code transmitted over a channel with memory, $m = 2$. In this case, in order to find out which of the two paths has the minimum metric, one step is further required than that at point (X), that is, the comparison of the paths (0000) and (1000) should be performed, instead of that between (000) and (100).

The previous observation can also be verified by noting the dependence of the expression of $V(z_k, \sigma_k, \sigma_{k+1})$ on the previous codeword ((000) and (011) in this example, fig. 2.14a). In order to maintain the convenient principle of a trellis-based implementation of the VA, that from a set of branches merging at one trellis knot the selection of the survivor at this point should define the closest branch to the received one, an expansion of the trellis is necessary, as illustrated in fig. 2.14b.



(a)



(b)

Fig. 2.14 (a) Decision between the paths 000 and 100 of the rate-1/3 $L = 3$, convolutional code. (b) Expanded trellis description of the rate-1/3 $L = 3$, convolutional code.

By the increment of the trellis' states, we guarantee that the last m bits of any two competing branches will be the same, and a decision at the merging point, concerning the survivor, is therefore allowable (point A in fig. 2.14b).

In the general case of decoding a convolutional code with rate $\frac{k}{n}$ and constraint length L , transmitted over an ISI channel with memory m , an expanded trellis can be used, with number of states :

$$M^{L-1+m'} \quad (2.35)$$

where $M = 2^k$ and m' is the block memory of the channel.

There is an exception to the previous conclusion which is of interest. Suppose that we use a systematic convolutional code with rate $\frac{k}{n}$ whose the last k bits of each codeword, consist of information bits and thus remain uncoded. If the channel memory m is less or equal to k , then no trellis' expansion is required and the system's memory equals that of the encoder, $(L-1)$. Note that by definition, when two paths merge, they should exhibit $k(L-1)$ alike information bits. If at least, the last m of them are transmitted uncoded, then the last m bits of the codewords of all the competitors are the same, and a decision at the merging point is therefore allowable.

II. The decoding procedure .

The first act of the decoder is the separation of the received sequence into segments of length n , so that the consecutive vectors z_i for $i = 1, 2, \dots, N'$

can be formed. Then starting from the zero state of the trellis, an information symbol u_1 (0 in the previous example) should produce a codeword of length n , which is translated into PAM values and formulates the vector b_1 . These values of b_1 and z_1 , accompanied by the channel characteristics r_j for $j = 0, 1, \dots, m$ are all the necessary components for the computation of the metric $V(z_1, \sigma_0, \sigma_1)$.

Once the metric computation and the overall memory were defined, the VA applies again as a recursive computational technique that performs MLSE. Parallel processing can also be used for the computation of the quantity $V(z_1, \sigma_{k-1}, \sigma_k)$, due to its accumulative form (eq. 2.33).

III. Complexity .

Suppose a convolutional code $\frac{k}{n}$ with constraint length L . The necessary amount of computations for any transition between states can be expressed as

$$M^{L-1}(A + \delta) \quad (2.36)$$

here δ depends on hard or soft-decision decoding mode and encompasses the computations for the transition-metric.

In the case where ISI is also present and the convolutional code is not systematic, the necessary computations are

$$M^{m'} M^{L-1}(A + n f(m)) \quad (2.37)$$

where $f(m)$ represents the amount of computations needed for each

$V(z_k, \sigma_{k-1}, \sigma_k)$ of eq. (2.33) as a function of the channel memory m .

When a rate- k/n systematic convolutional code is applied, the complexity reduces to :

$$M^{L-1}(A + n f(m)) \quad (2.38)$$

IV. Conclusion.

Transmission of nonsystematic convolutional codes over ISI channels implies an overall memory of $L - 1 + m'$. The exponential contribution of the memory of a code to the complexity of a Viterbi decoder is the prohibiting factor in the usage of this algorithm in decoding highly efficient convolutional codes with great values of L . Thus, any linear increment of the overall memory, due to ISI, has significant effects on the decoding complexity.

On the other hand, it was previously shown that rate- k/n systematic convolutional codes require M^m less decoding effort than their nonsystematic competitors, when the condition $k \geq m$ is satisfied. It is also known that they do not perform as well as nonsystematic codes with the same constraint length (Sec. 1.2.1). Selection of $k = m$ seems to optimize their capability and suggests their application in band-limited channels as a practical compromise between error performance and decoding complexity.

CHAPTER 3

3 Simulations and Results [16].

The following sections describe the simulation effort in this project, in order to observe the relative error-performance of the two decoding techniques. As an introduction, the structure of the simulation is briefly discussed (see also fig. 3.1), followed by extensive descriptions of the techniques which have been used here.

The simulation program had the following components :

a. *Generation of the information sequence*

A uniformly distributed random sequence with values in the range (0-1) (produced by a standard random number generator on a VAX-11/780 system) formed by the intervention of a threshold , the binary information sequence.

b. *Encoding and Modulation.*

A rate-1/3, $L = 3$ convolutional code and Binary Phase Shift Keying (BPSK) modulation were used here.

c. *Channel Model*

Memoryless channel was assumed with two variations.

In an unquantized AWGN channel the received signal is the sum of the transmitted signal and noise with a Gaussian distribution . The implementation of the Gaussian noise source was based on the ' composition method ' modified by another technique known as the ' Teichroew ' s method ' (Sec. 3.1.1) in order to

improve the accuracy of the approximation.

Under Binary Symmetric Channel assumption, a hard quantization of the received signal reduces the infinite output alphabet into a binary one. An alternative (and quite efficient) method adopted here is the modulo 2 summation on the encoded stream of a random binary valued one, binomially distributed with p the crossover probability of the BSC.

d. Decoding.

The software implementation of both the decoding algorithms, were based on the trellis description of the codes. The one-to-one representation of each individual path enabled accurate testing procedures.

The 'independent replication' technique (Sec. 3.2.1) was used for the computation of the probability of error and the definition of confidence intervals. The 'antithetic variate' (Sec. 3.3.1) method produced pairs of random sequences with the same mean and negative correlation, aiming at the compression of the observed samples variance. N such independent pairs of different information and noise sequences (produced by N different 'seeds') used to define confidence intervals by means of a student's t distribution with $N-1$ degrees of freedom.

Simulations conducted over a range of 2 - 7 dB, due to computer time limitations and the high precision requirements imposed by the closeness at the performance of the two decoding algorithms.

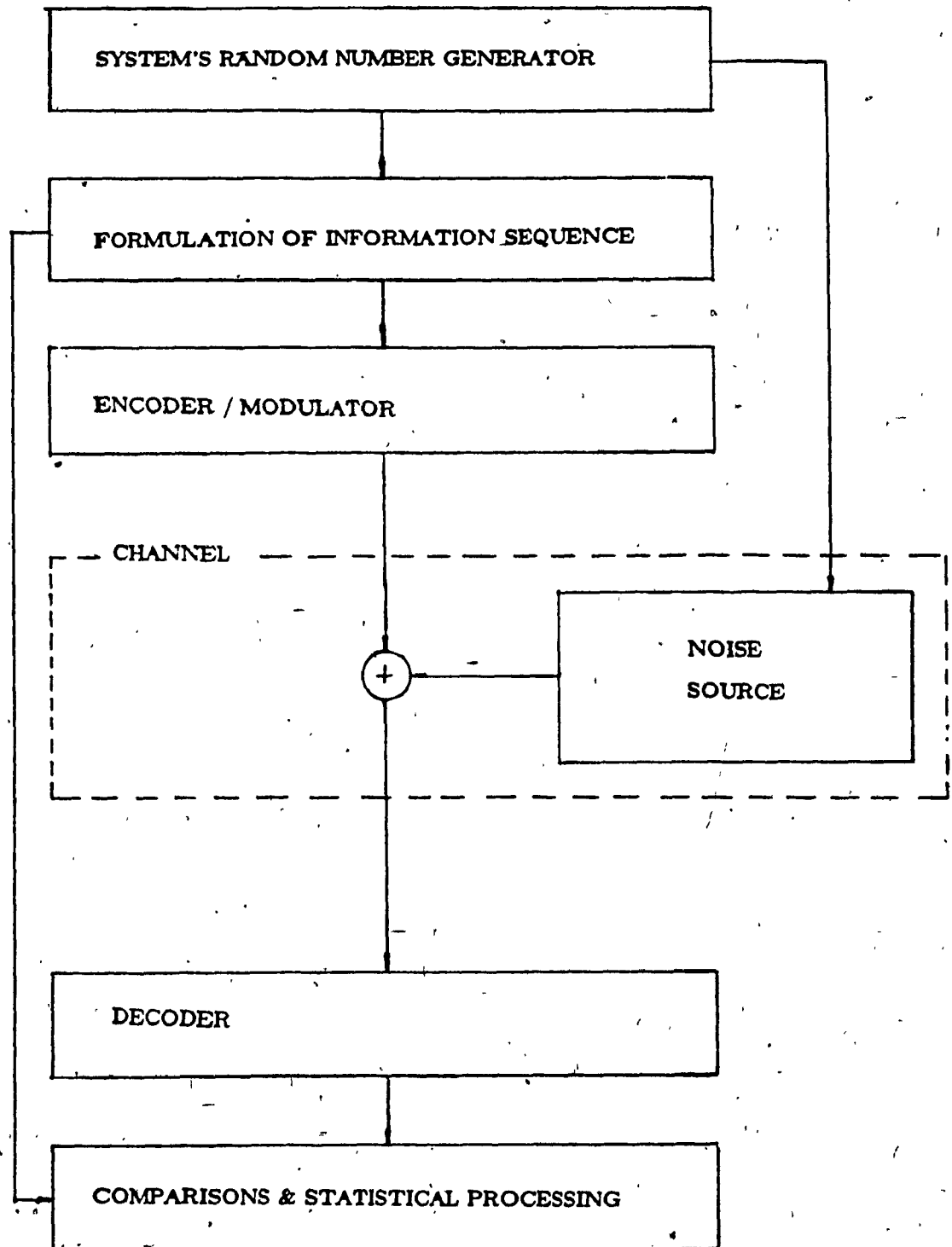


Fig. 3.1 Most general flow chart.

3.1 Simulation of the Noisy Conditions of the Channel

The random error stream assumed two different forms, depending on the channel model under consideration.

A. Binary Symmetric Channel model.

In the case in which hard-decisions was employed, a binomially distributed error sequence was (modulo 2) added on the information-coded sequence. This error sequence was produced by the comparison of a uniformly distributed (0-1) random sequence with a threshold value p . When the i^{th} symbol of the latter sequence found to be less or equal to p , the i^{th} bit of the error sequence was defined as 1 and 0 otherwise. Modulo -2 addition of the coded and the error sequence, simulated the transmission of the convolutional code over a binary symmetric channel, with crossover probability p .

B. AWGN channel model

When transmission of convolutionally encoded signalling waveforms by binary coherent Phase Shift Keying on the AWGN channel is assumed, the received signal can be written as :

$$y_{jm} = 2\alpha E (2c_{jm} - 1) + v_{jm}$$

where α represents the attenuation factor, E is the transmitted signal energy for

each code bit c_{jm} . The quantity v_{jm} denotes the additive Gaussian noise whose values during the conduction of these simulations were produced by the Gaussian random number generator that is described below.

3.1.1 The Gaussian-Random Number Generator .

Its implementation is based on the "composition method" which uses the property of the *asymptotic normality* of the central limit theorem.

Defining a random variable S by

$$S = Y_1 + Y_2 + \dots + Y_n$$

where Y_1, Y_2, \dots, Y_n are independent samples of the uniform variates between 0 and 1. Then for large n , the variable S is approximately normally distributed with mean $n/2$ and variance $n/12$. The variable X defined by

$$X = \frac{(S - \frac{n}{2})\sigma}{\sqrt{n/12}} + \mu$$

approximates the normal variable with mean μ and variance σ^2 . A convenient choice is $n = 12$ since eliminates the square root term from the last expression. This value of n truncates the distribution at $\pm 6\sigma$ limits and is unable to generate values beyond 3σ .

• *The Teichroew's method*.

In order to improve the accuracy of the previous approximation, the following procedure has been proposed and is known by the name of Teichroew's method [17].

Choosing $n = 12$, $\mu = 0$ and $\sigma = 1/4$ and denoting the resulting variable by R , we get from the previous expression :

$$R = \left(\frac{1}{4}\right)[Y_1 + Y_2 + \dots + Y_{12} - 6]$$

Setting

$$X = [((a_9 R^2 + a_7)R^2 + a_6)R^2 + a_5]R^2 + a_1]R$$

where

$$a_1 = 3.949846138$$

$$a_5 = 0.252408784$$

$$a_6 = 0.076542912$$

$$a_7 = 0.008355968$$

$$a_9 = 0.029899776$$

Then X is a fairly good approximation to a variate with the standard normal distribution and its transform

$$Z = X\sigma + \mu$$

is normally distributed with mean μ and variance σ^2 .

3.2 On the statistical analysis of the simulation results.

The following sections describe the conduction of simulation, dealing with questions as sample size determination, reliability and methods for improving the efficiency of statistical simulations.

3.2.1 Choice of sample size, the 'independent replication' method.

In order to perform a reliable simulation and establish 'confidence intervals' for our observations, we need to know the variance σ^2 of the variable that we want to predict (Bit error rate in this case). The exact value of σ^2 is generally unknown, thus an estimation of this quantity must also be performed.

The method followed for this purpose is the 'independent-replication method' [16] — the replication of the simulation experiment using different (and presumably independent) streams of random numbers for different runs. (The independence of these random streams in this statistical experiment is assumed by rerunning the random number generator of the computing system for different starting values ('seeds')).

Denoting by μ the mean value of the probability of error P , we assume that N observations of this variable, P_1, P_2, \dots, P_N , are statistically independent with normal distribution with the unknown mean μ . By definition a "confidence level" denoted by $(1-\alpha)$, is the degree of assurance that a particular statement is correct, under specified conditions.

Such a confidence interval for the sample mean

$$\bar{P} = \frac{\sum_{i=1}^N P_i}{N}$$

can be obtained by means of normalization of the random variable P by the transformation

$$Z = \frac{(\bar{P} - \mu)\sqrt{N}}{\sigma} \quad (3.1)$$

in the form :

$$P \left\{ -z_{\alpha/2} \leq \frac{(\bar{P} - \mu)\sqrt{N}}{\sigma} \leq z_{\alpha/2} \right\} = 1 - \alpha$$

or equivalently

$$P \left\{ \bar{P} - \frac{z_{\alpha/2}\sigma}{\sqrt{N}} \leq \mu \leq \bar{P} + \frac{z_{\alpha/2}\sigma}{\sqrt{N}} \right\} = 1 - \alpha \quad (3.2)$$

where $z_{\alpha/2}$ denote the upper $\alpha/2 \times 100$ percentile of the standard normal distribution (the p.d.f of the previously defined Z variable), and σ is the standard deviation of the Probability of error. The frequency interpretation of the probability statement (2) is as follows. If the procedure of taking random and independent observations and constructing the associated random interval , $\bar{P} \pm z_{\alpha/2}\sigma/\sqrt{N}$, is

repeated many times , approximately $(1-\alpha) \times 100$ percent of these random intervals will contain μ .

Defining the values of the desired confidence coefficient (and $z_{\alpha/2}$ subsequently) and that of the precision requirement $h = \bar{P} - \mu$ and knowing the value of σ , the necessary number of observations N can be calculated from

$$N = \left(\frac{z_{\alpha/2} \sigma}{\bar{P} - \mu} \right)^2 \quad (3.3)$$

The exact value of the variance σ^2 is not known in our case , so an estimation s_P^2 derived by means of a trial test should be used instead, in the form

$$s_P^2 = \frac{\sum_{i=1}^n (P_i - \bar{P})^2}{n-1} \quad (3.4)$$

where n is the sample size of this test (for practical purposes greater than 30 or so to assure the valid use of the central limit theorem).

Following the previous analysis in the software implementation of the BSC and the Viterbi decoder , the probability of error for $p = 0.07$, found to have the value, $\bar{P} \approx 0.006$, by conducting 30 different runs produced by 30 different seeds of the VAX-11/780 random number generator. Because of the great number of trials the observed s_P can be used instead of the true value of σ . The sample standard deviation s_P found to be $4.5\% \bar{P} \approx 0.000313$ thus, for precision

$$h: \bar{P} - \mu \approx 2.8\% \bar{P} \approx 0.0002$$

and confidence level 98%, the number of necessary observations found to be (according to eq. 3.3)

$$N = \left(\frac{2.236 \times 0.000313}{0.0002} \right)^2 \approx 13$$

Note here that in order to compute the mean value of P for various crossover-probabilities p , another sample size should be determined, that of the length of the information sequence, one of the most important factors of the standard deviation of P . (Clearly, this length should be longer for relatively small crossover-probabilities than the respective one for bigger values of p).

The procedure adopted here is based on the exploitation of qualitative knowledge available for this situation.

The general shape of the P 's curve with respect to p (or equivalently to SNR), is known. Proceeding in the calculation of the probability of error for different values of p by steps equal to Δp , we maintain a degree of certainty for the order of the probability of error and consequently for the necessary sample length.

Also, although the exact distribution of the probability of error P is not known, should exhibit greater values of σ for $p > 0$, than those for $p \ll 1$ (for example 3 bit errors imposed on an encoded sequence of 10 symbols of a

code with $d_{\min} = 6$, as the one investigated here, would cause at most 1 symbol-error, whereas a greater number of channel errors should definitely increase the possible symbol-error patterns).

Such observations do not supply the simulation with the necessary precision. In order to maintain the same accuracy and the same confidence in the calculation of each probability of error P for various SNR values, we recalculate the required number of observations for each value of p , using the following expression :

$$N = \left(\frac{t_{\alpha/2, N-1} s_P}{\bar{P} - \mu} \right)^2 \quad (3.5)$$

where $t_{\alpha/2, N-1}$ denote the upper $\alpha/2 \times 100$ percentile of the student's t distribution with $N-1$ degrees of freedom, and s_P the observed standard deviation.

The values of the probability of error that were calculated here, belong in 95% confidence intervals with precision $h : 2.8\%$ of the observed sample mean \bar{P} .

3.3.2 The single run method .

Another method to calculate the probability of error is to make the runs consecutively, using the same random sequence. This long simulation is then divided into N contiguous segments and the P_i 's mean values are treated as

individual observations. The advantage of the 'single-run' method, in comparison with the 'independent replication' one, is the reduction of the transition period to the initial one (for the first segment). Its disadvantage is that the set of sample means $P_i, i=1,2,\dots,N$ is not, strictly speaking statistically independent.

In our case stabilization period is not an important issue primary because of the deterministic structure of both the decoding algorithms. Given that the length of the information sequence is long enough for a few hundreds of errors to be counted at the receiver, each of these N contiguous segments can be substituted by an independently produced one, with the same length.

3.2.3 The regenerative method

Another method that can be used in the estimation of the probability of error for various SNR levels, is the regenerative method [16].

This method is used in *regenerative* systems, i.e. systems exhibiting a particular state called *regenerative state*, such that whenever this state is reached, the history of past states of the system has no influence on the future of the system.

A sequence of epochs at which the system returns to such a *regenerative state* are called 'regeneration' (or renewal) points and the time between the k th and the $(k+1)$ th regeneration points is called the k^{th} cycle. Thus, for a regenerative process $\{X(t); t \geq 0\}$, the continuation of the process beyond a regenerative point, say t_1 , is a probabilistic replica of the whole process commencing at epoch

0, independent of $\{X(t); t \geq 0\}$. The definition is also applicable to regenerative processes in discrete time.

If the time between two successive regeneration points is finite, then observations made in this cycle are statistically independent of the observations made in other cycles. In addition, the proper collection of measurements made during such cycles are identically distributed.

In our case a regenerative state can be defined as the state of the decoder, where a *merge* phenomenon occurs, i.e., the situation in which all the 'survivors' of the $(k+1)$ th pass (or emerge) from a certain 'substate' of the k th state. Such a situation implies that the relative metrics of the substates are of no importance, practically, the decoding process restarts from the certain substate and its future evolution is independent of its past fig. 3.2.

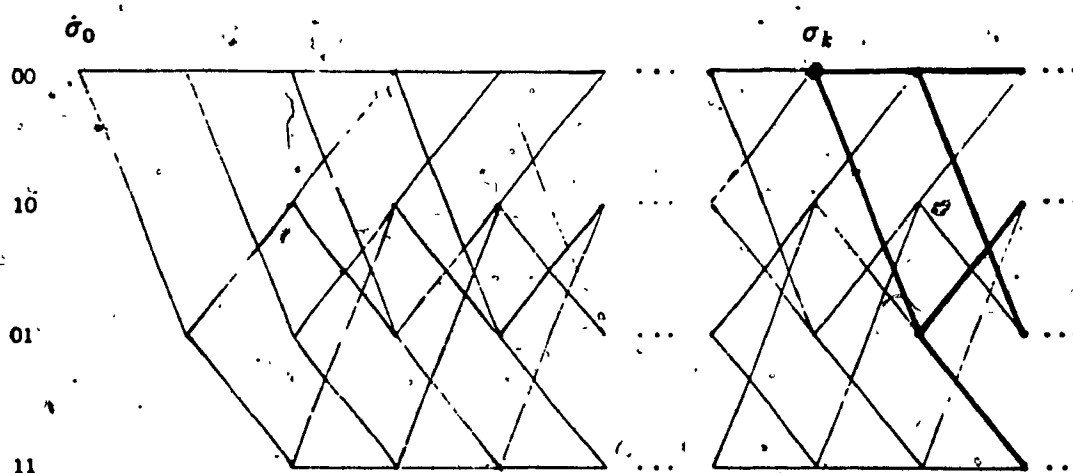


Fig. 3.2 Regenerative states σ_0, σ_k (heavy lines: survivors).

Finally, correct assessment of the error conditions that lead to a merge phenomenon, could save computation time in the conduction of the experiment in high SNR levels.

An obvious way to make the encoding/decoding process regenerative and compress the time consuming simulations for high SNR values is the following technique that effectively uses the error correcting capability of convolutional codes.

The alternative way of decoding convolutional codes is the one of selecting the single most likely path by periodically forcing the encoder into a particular state. This is done by inputting $k(L-1)$ bit fixed information sequence (usually the all zero, one) to the encoder after each set of N information bits. Consecutive independent decisions concerning N bits at each time, lead to the decoding of the complete information sequence.

Now let's assume a rate- $\frac{k}{n}$ convolutional code with constraint length L and minimum distance d_{min} transmitted over a BSC channel with crossover probability p . Since we truncate the convolutional code in blocks of N symbols, we practically deal with a block code whose error correcting capability is [18] :

$$C_e = \begin{cases} \frac{d_{min}}{2} - 1 & d_{min} : \text{even} \\ \frac{d_{min} - 1}{2} & d_{min} : \text{odd} \end{cases} \quad (3.6)$$

In order to implement the previously discussed technique of decoding the information sequence in parts of length $N + k(L - 1)$ bits (or equivalently $\frac{n}{k}[N + k(L - 1)]$ code bits, since the code rate is k/n), we also have to add modulo 2 to the latter sequence a binary error sequence of the same length, binomially distributed with p , the crossover probability of the channel.

Decoding of the noisy sequence is first assumed to be done by means of the Viterbi algorithm. The computer time compression is attained by counting the number of 1s contained in the binary error sequence -- the number of channel errors. Whenever this number is less or equal to the error correcting capability C_e of the code no symbol errors will occur, consequently this part of information is assumed correctly decoded and the decoder refrains from further processing. The previous procedure is again repeated for the next subsequence, until the whole sequence is decoded.

A similar situation can be shown - by means of a worst case argument - to hold in the case of optimal symbol-by-symbol decoding, with some limitations concerning the length $N + k(L - 1)$ of each subsequence (see APPENDIX A).

The computer time compression is quite significant when $p \ll 1$ (high SNR values). The probability of skipping a cycle of length $N + k(L - 1)$ bits, is

$$P(\text{number of channel errors in } N' \text{ code bits} \leq C_e) = \sum_{i=0}^{C_e} \binom{N'}{i} p^i (1-p)^{N'-i}$$

where N' represents the length of the encoded information

$(N' = \frac{n}{k}[N + (L - 1)k])$ and p is the crossover probability of the binary symmetric channel.

3.3 Monte Carlo techniques.

It was previously discussed that the estimation of the probability of error in the performance of a Viterbi or Symbol-by-Symbol decoder for a certain SNR level, requires the repetition of the statistical experiment N times, where N is defined by the values of the observed variance σ_p^2 , the level of confidence and the desired precision requirements. Reduction of the sample variance is always desirable since it minimizes the necessary number N of independent runs. An obvious way to attain such a situation, is the increment of the length of each run.

More sophisticated methods for improving the efficiency of statistical simulations are known as Monte Carlo techniques [16]. The basic idea is the use of any available knowledge of the simulation. Proper exploitation of such information, in the form of definition and use of an auxiliary random variable, can substantially reduce the variance of the sample observations. The reader should be reminded at this point that the randomness in our experiment is due to the simulation of the noisy conditions of the channel in the form of an error sequence imposed on the encoded information. The following two Monte Carlo techniques exploit the information carried by the formulation of these error sequences.

3.3.1 The antithetic-Variate method

Let's suppose that by conducting a simulation we want to estimate the mean value of a random variable $Y = Y(R)$ where the vector R represents a stream of random numbers, uniformly distributed between 0 and 1.

If another random variable $X = X(R)$ is made available to us and that X has the same (unknown) expectation as Y and possesses a high degree of negative correlation with Y , then the variable

$$\alpha Y(R) + (1-\alpha)X(R)$$

has the same unknown mean as Y , but its variance can be made significantly smaller than $\text{Var}[Y]$ by suitable choice of α .

The key in applying this method to any simulation is to find an auxiliary variable X . A very simple system-independent method for obtaining X is to let

$$X(R) = Y(R')$$

where the vector $R' = 1 - R$, that is $R'_i = 1 - R_i$ for $i = 1, 2, \dots$

Since R is uniform (0, 1), R' has the same distribution as R , consequently, the same mean value, 0.5. The optimum value of α in this case is 1/2. A simulation is performed by using a random-number sequence R to compute values of Y , and only then are the values of the auxiliary variable X calculated by re-running the simulation, using its antithetic partner R' . These two steps are completely independent except that in performing the random sampling in calculating Y ,

provisions must be made for obtaining data that will be needed later in calculating X 's. If the above procedure creates negative correlation between the two responses X and Y , then their mean $(X + Y)/2$ will have a smaller variance than that of Y (or X).

3.3.2 Implementation of the 'antithetic-variate' method.

The use of the previously discussed technique in the simulation of the noise process in the present work, took the following form.

An equiprobably valued sequence R of length n produced by the random number generator of the system, formulated a complementary one R^* ; $R^* = 1 - R$. Simple modifications of the main process depending on the decoding mode (soft or hard) defined the following respective procedures.

A. Simulation of noise under BSC assumption .

The comparison of the values of the two sequences with the value of the cross-over probability p , produces two different error streams binomially distributed with mean np , negatively correlated. Then this pair of sequences imposed on the encoded information one, feeds two identical decoders and invokes two responses P_j' and $P_j'^*$ denoting the sample means of the random processes $P(R)$,

$P(R^*)$. A third value $P_j = \frac{P_j' + P_j'^*}{2}$, defined as the representative value of the

j^{th} experiment, is used for the computation of the sample variance. The previous

procedure is repeated N times where N is the necessary number of independent sample observations for the estimation of the mean value of P .

The usage of the *antithetic-variate* method in these simulations did not only reduce the sample variance, but also the necessary computations, as a consequence of the generation of some initial processes only once. These processes, namely, the information sequence's formulation, the encoding procedure and the random number generator calls, are common for both the random sequences, resulting in a computer time compression of 15% (calculated in a hard decision decoding case by means of the Symbol-by-Symbol decoder).

The reduction of the sample variance observed here was 10% of the measured one in an 'Independent replication method' test of the same length.

B. Simulation of noise under AWGN assumption.

When soft-decision decoding is attempted, a complementary stream is produced in a similar fashion to the previously reported one. In the AWGN channel simulation the error sequence is composed of variates normally distributed. Two successive calls of the Gaussian generator with complementary inputs formulate the pairs of the partially antithetic error symbols.

This procedure can be further simplified by taking a closer look at the generator's structure. Its implementation is based on the central limit theorem, subsequently modified by Telchroew's method (see sec. 3.1.1) with

$$R_1 = \left(\frac{1}{4}\right)[Y_1 + Y_2 + \dots + Y_{12} - 6]$$

where $Y_i, i = 1, \dots, 12$ are uniform variates between 0 and 1, and

$$X = [((a_6 R_1^2 + a_7) R_1^2 + a_8) R_1^2 + a_9] R_1$$

where a_i is a set of constants. Then X is a good approximation to a variate with the standard normal distribution.

It was earlier mentioned that the set of Y_i , represents independent uniform variates between 0 and 1. By the definition of the antithetic sequence's variates $Y_i^* = 1 - Y_i$, the variable R_1^* becomes :

$$R_1^* = \left(\frac{1}{4}\right)[1 - Y_1 + 1 - Y_2 + \dots + 1 - Y_{12} - 6] = -R_1$$

and consequently (see the definition of X in terms of R_1)

$$X^* = -X$$

This means that in order to form the antithetic random sequence X^* from its original counter-part X , it is sufficient to invert the sign of the variates of X .

The benefits of the use of the previous method in soft-decision decoding can be summarized as follows : A Gaussian error pattern which randomly maximizes the number of decoding errors, and increases the sample variance, can possibly be neutralized by the parallel use of another error pattern composed by the opposite variates.

3.3.3 The Control-Variate method .

Reduction of the variance of sample observations can be also accomplished by the control-variate method [1]. Consider a random variable Z defined by

$$Z(R; \beta) = Y(R) - \beta(X(R) - E[X(R)]) \quad (3.5)$$

where $X(R)$ called the auxiliary variable, is a random variable whose expectation $E[X(R)]$ is known (R again represents a stream of random numbers statistically independent and uniformly distributed between 0 and 1). Clearly, Z is an unbiased estimate of $E[Y]$ for any β and its variance is given by

$$\text{Var}[Z] = \text{Var}[Y] + \beta^2 \text{Var}[X] - 2\beta \text{Cov}[X, Y].$$

The coefficient β is selected to minimize the variance $\text{Var}[Z]$, which leads to

$$\beta_0 = \text{Cov}[X, Y] / \text{Var}[X]$$

and the minimized variance is therefore given by

$$\text{Var}[Z] = \text{Var}[Y](1 - \rho_{XY}^2) \leq \text{Var}[Y]$$

where ρ_{XY} is the correlation coefficient between X and Y . In practice, the covariance between X and Y will not be known; hence it must be estimated from data. For a given N independent observations (or replications) $X^{(j)}$ and $Y^{(j)}$ ($1 \leq j \leq N$), an estimated optimum β_0 will be of the form

$$\hat{\beta}_0 = \frac{\sum_{j=1}^N (Y^{(j)} - \bar{Y})(X^{(j)} - \bar{X})}{\sum_{j=1}^N (X^{(j)} - \bar{X})^2}$$

The above theory is used in a stochastic simulation as follows. We wish to estimate the expected value of the response Y , where Y is related to the random vector R in unknown functional form $Y(R)$. For a chosen control variable $X(R)$ we then observe simulated outputs $Y^{(j)}$ and $X^{(j)}$ for random number stream R . We approximate $E\{Z\}$ by \bar{Z} :

$$\bar{Z} = \frac{1}{N} \sum_{j=1}^N (Y^{(j)} - \hat{\beta}_0 X^{(j)}) + \hat{\beta}_0 E[X].$$

The expectation of the predictive variable X is known. The control-variate variance-reduction method minimizes the variation of the response variable Y by taking advantage of the information furnished by the auxiliary variable X .

One possible predictive variable that can be used in this simulation is the error sequence $ER(R)$, the sequence which is added to the transmitted sequence and simulates the noise process. Under the BSC assumption, this variable is binomially distributed with mean value equal to np , whereas in the unquantized model is Gaussian distributed with zero mean.

The idea behind the use of the control-variate method in this statistical experiment is the fact that a randomly produced error sequence of a certain length n , usually exhibits a sample mean different from its expected one. A

Viterbi or Symbol-by-Symbol decoder can be forced in error-decisions by two factors : certain error patterns and the additive noise level (or equivalently, the number of channel errors in a BSC). The definition of the error sequence as a predictor variable, could possibly reduce the variance of the sample observations, by weighting the measured P_i by a factor proportional to the sample mean of the error sequence involved in the experiment.

3.4 Results of the Simulation

The rate-1/3 , $L=3$ systematic convolutional encoder of fig. 1.6 , was assumed as an introducing correlation process in a randomly produced binary information sequence. After addition of channel errors to the coded sequence, the decoding process was independently performed by means of the Viterbi and the optimal symbol-by-symbol algorithms (their function was previously analyzed in Chap. 2) employing a decoding delay of 18 symbols ($6L$).

The correctness of the simulation results was tested by :

- i) checking the correct function of the individual parts of the program (encoder, error source, decoder) using sequences of short length.
- ii) removing the source of channel errors and decoding long sequences without error occurrence.
- iii) indirectly comparing the observed bit error rate of the VA decoder with existing simulation results [8]. Fig. 3.3. displays the soft-decoding (8 level) bit error rates of various 1/3-rate convolutional codes of maximum d_{free} , with constraint length L varying from 4 to 8. Fig. 3.4 shows the bit error rate of the 1/2-rate family of codes of maximum free distance, for hard decisions and L varying from 3 to 8. Due to the lack of simulation results for the specific 1/3-rate $L = 3$ systematic convolutional code that was used here, its error performance was indirectly compared with the respective ones of the 1/3-rate , $L=4$, code (inferior to this code) and the 1/2-rate, $L=3$ code (superior to this code).

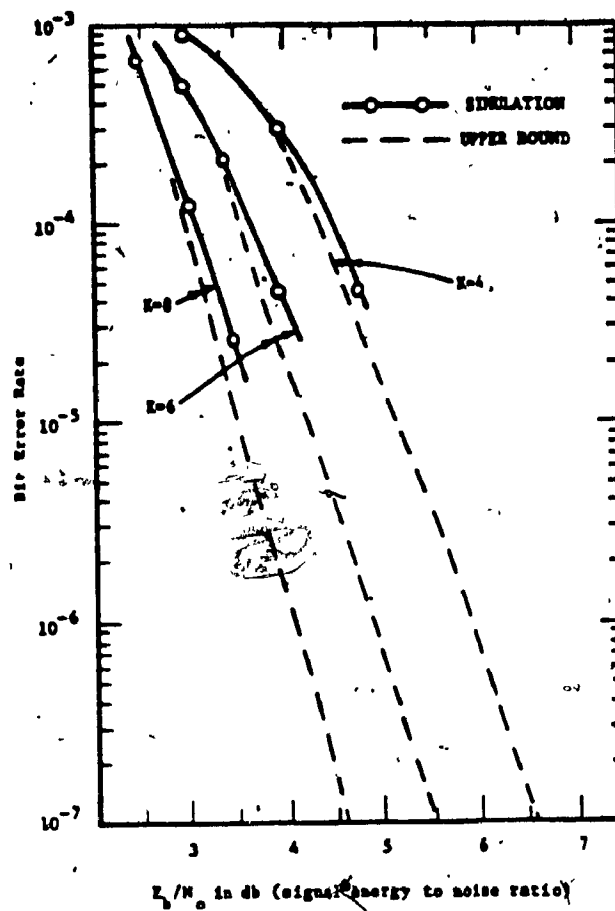


Fig. 3.3 Performance of the rate-1/3, $K=4$, 6 and 8 codes with Viterbi soft-decoding ($Q=8$ and by K is denoted the constraint length).

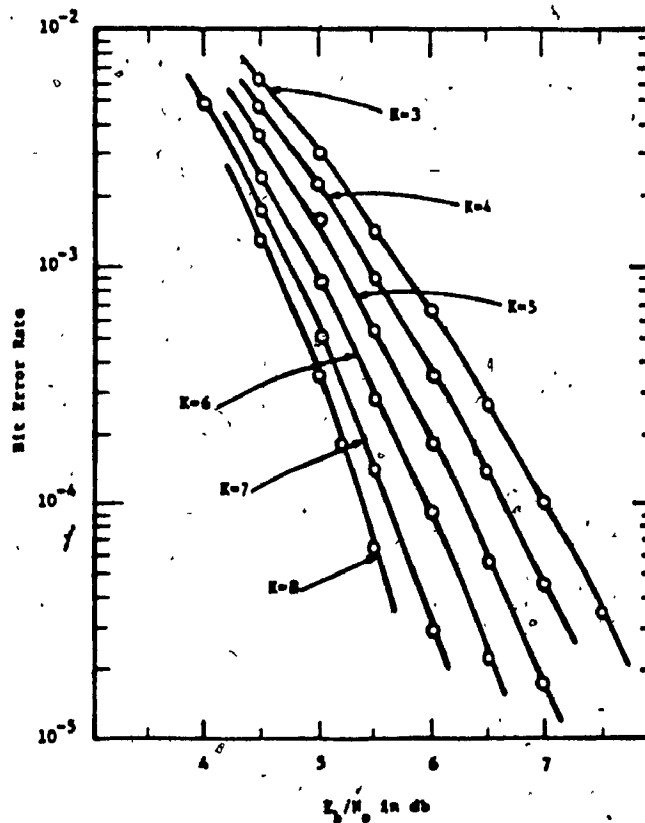


Fig. 3.4 Bit error rate versus E_b/N_0 for 1/2 Viterbi decoding and hard-quantized received data with 32-bit paths, $K=3$ through 8.

Fig. 3.5 shows the bit error rate performance of the $1/3$ -rate, $L=3$ systematic code for both hard and infinitely soft decisions of a Viterbi decoder which employs a decoding delay of 18 bits.

The results concerning the comparative error performance of the VA and the optimal symbol-by-symbol algorithm, when hard-decisions were employed, are presented in fig. 3.6. The symbol-by-symbol algorithm performs slightly better than the Viterbi algorithm (less than 0.1 dB). The simulation study was done for various SNR values in the range 2-7 dB. In table 3.1 , the previous results are displayed with better accuracy.

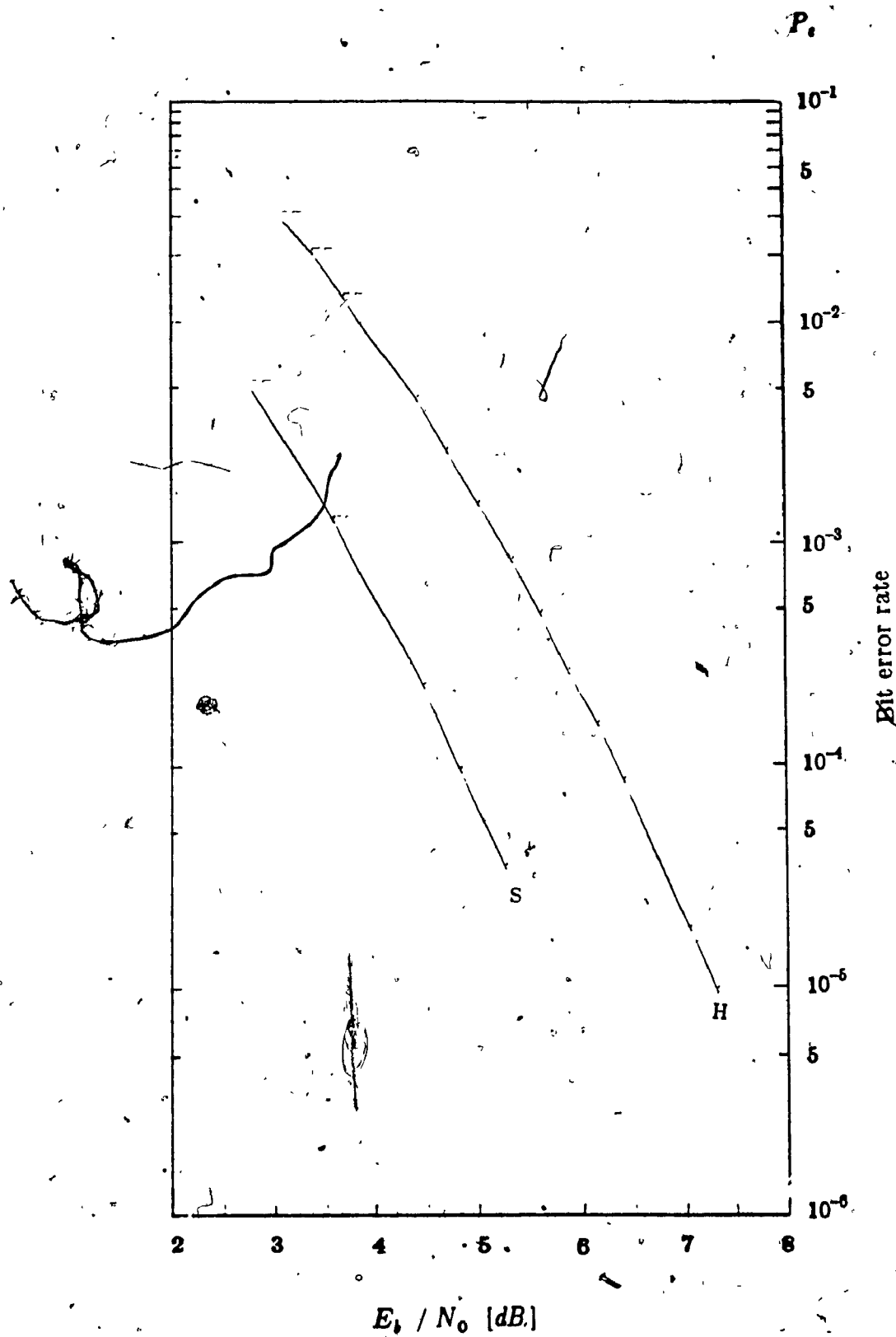


Fig. 3.5 Viterbi decoding for the rate-1/3, L=3 convolutional code, hard and soft decisions.

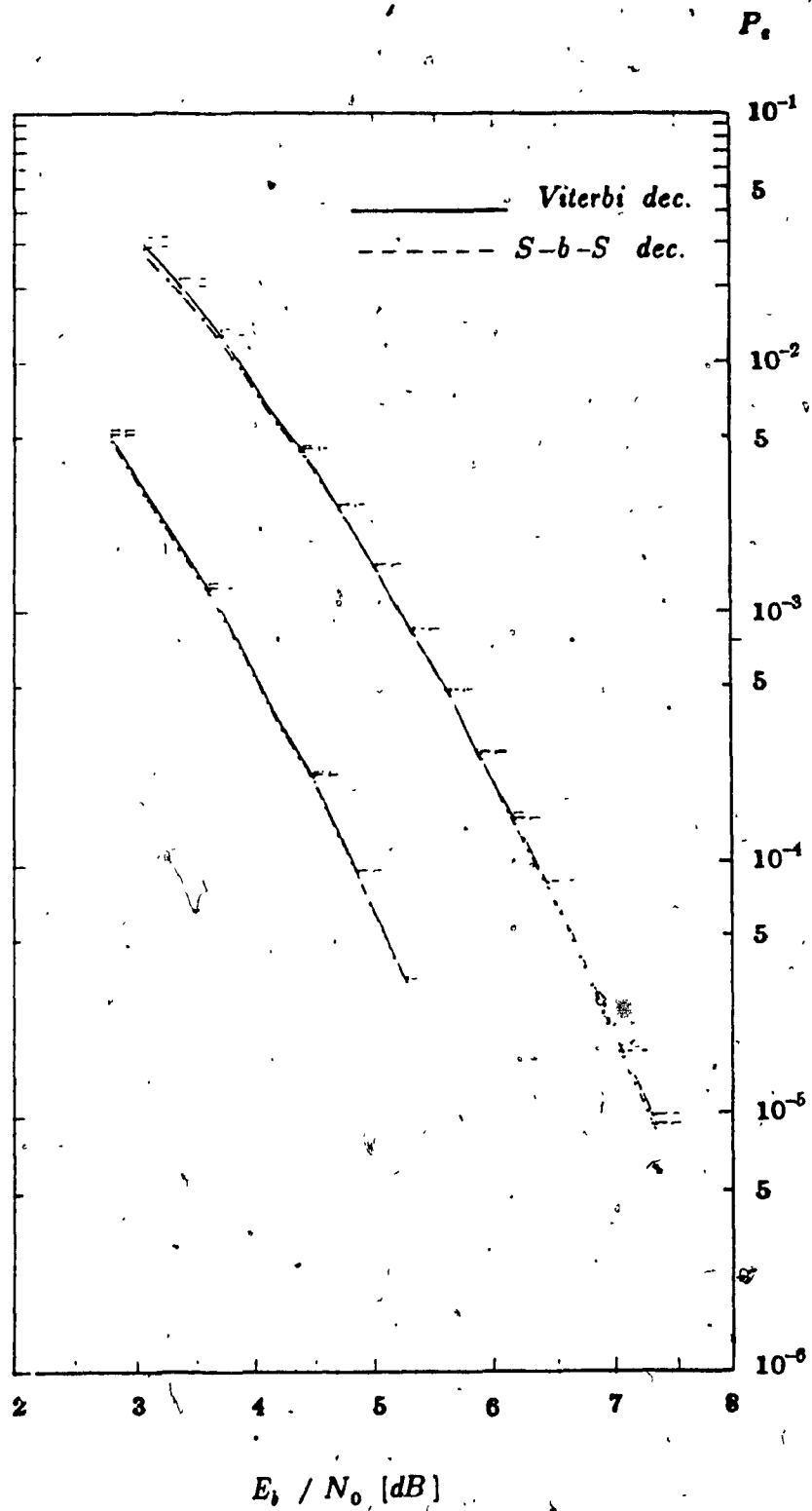


Fig. 3,6 Comparative error performance of the two decoders, hard and soft decisions.

<i>Table 3.1 Comparative Bit Error Rate (hard decoding)</i>		
SNR [dB]	P_e Viterbi dec.	P_e Sym-by-Sym dec.
3.057	0.029651711	0.027111612
3.368	0.020273507	0.018867580
3.698	0.012737179	0.012009114
4.405	0.004400712	0.004259995
4.725	0.002566239	0.002539865
5.033	0.001494017	0.001478820
5.331	0.000829772	0.000817043
5.619	0.000474225	0.000465595
5.898	0.000260838	0.000260081
6.168	0.000151024	0.000143508
6.429	0.000084345	0.000081130
7.052	0.000018177	0.000017053
7.328	0.000009594	0.000008839

The closeness in the error performance of the two decoders was again observed when soft-decoding was employed. Fig. 3.7 shows the combined results for hard and soft decisions of both the decoders under consideration. The performance gain due to soft-decoding is approximately 2 dB and the slight superiority of the symbol-by-symbol decoder relative to the Viterbi decoder is again displayed throughout the examined SNR range (table 3.2).

<i>Table 3.2 Comparative Bit Error Rate (soft decoding)</i>		
SNR _c [dB]	P_e Viterbi dec.	P_e Sym-by-Sym dec.
2.77331	0.005089318	0.0048865781
3.59766	0.001246581	0.0012004428
4.48692	0.000221867	0.0002143607
4.87020	0.000092849	0.0000896741
5.28112	0.000033958	0.0000330639

Although both the techniques are optimal in a different sense and the optimal symbol-by-symbol decoder constantly performs better than the Viterbi decoder, their closeness in error performance (in the range of 2 - 7 dB that was examined here) is in favor of the Viterbi algorithm ; due to the comparably less decoding effort that this technique requires (see eq. 2.19 and 2.20).

Conclusions

The optimal algorithm in [1], originally proposed for symbol-by-symbol detection of pulse amplitude modulated sequences was further considered in this work, as an optimal decoding technique of convolutional codes. This algorithm is similar to the Viterbi algorithm whose optimality criterion is focused on sequence detection.

Due to reasons of complexity (explained in Appendix B) an analytical expression for the symbol-by-symbol algorithm is still not available, thus a comparison between the latter and the Viterbi algorithm was subject to computer simulations for their relative error performance to be observed.

The results concerning the use of a rate-1/3, $L = 3$ systematic convolutional code and decoding performed by means of both the algorithms under consideration with soft and hard decisions, show a slight superiority in the performance of the symbol-by-symbol technique under both the decoding modes. The significance of the observed advantage of the symbol-by-symbol algorithm is balanced by the relatively increased complexity in computations that this technique implies.

A number of statistical techniques were also applied in order to supply confidence intervals to our observations and improve the efficiency of the conducted simulations. Chapter 3 serves also as a general survey of the most popular techniques aiming at the previous purposes.

Finally, a rather straight-forward analysis concerning the optimal sequence detection of convolutionally encoded signals transmitted over an ISI channel was also presented. The topic although not strictly related to the main purposes of this work, offers an interesting suggestion for a possible application of systematic convolutional codes in band-limited channels, as a practical compromise between error performance and complexity requirements.

REFERENCES-

- [1] J.F.Hayes, T.M.Cover, J.B.Riera, "Optimal Sequence Detection and Optimal Symbol-by-Symbol Detection : Similar Algorithms", *IEEE Trans. on Comm.*, vol. Com-30, Jan. 1982
- [2] G.D. Forney, " The Viterbi Algorithm ", *Proc. IEEE* vol. 61, pp.268-278, March 1973.
- [3] J.F.Hayes, "The Viterbi Algorithm Applied to Digital Data Transmission", *Commun. Soc.*, vol. 13, pp.15-20, Mar. 1975.
- [4] G.D. Forney, " Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference ", *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 363-378, May 1972.
- [5] G.D. Forney, " Convolutional codes I: Algebraic structure ", *IEEE Trans. Inform. Theory*, vol. IT-16, pp. 720-738, Nov. 1970.
- [6] Linkabit Corp., "Coding systems study for high data rate telemetry links ", Final Rep. on NASA Ames Res. Cen., Moffett Field, Calif., Contract NAS 2-6024, Rep. CR-114278, 1970.
- [7] A. Viterbi, " Convolutional Codes and Their Performance in Communication Systems", *IEEE Trans.Com. Tech.* vol-19, Oct. 1971.
- [8] J.A. Heller, J.M. Jacobs, " Viterbi Decoding for Satellite and Space Communications ", *IEEE Trans.Com. Tech.* vol-19, Oct. 1971.
- [9] H. Kobayashi, " Correlative Level Coding and Maximum-likelihood Decoding", *IEEE Trans. Inform. Theory*, vol. IT-17, pp. 586-594, Sept. 1971.
- [10] E. R. Berlekamp, *Key Papers in the Development of Coding Theory*, IEEE Press, 1974.
- [11] J. Proakis, *Digital Communications*, McGraw-Hill, New York, 1983.
- [12] R.W. Chang and J.C. Hancock, " On receiver structures for channels having memory ", *IEEE Trans. Inform. Theory*, vol. IT-12, pp. 463-468, Oct. 1966.

- [13] K. Abend and B.D. Fritchman, " Statistical Detection for Communication Channels with Intersymbol Interference ", *Proc. IEEE* vol. 58, pp.779-785, May 1970.
- [14] G. Ungerboeck, " Nonlinear Equalization of Binary Signals in Gaussian Noise ", *IEEE Trans. Commun. Tech.* vol. COM-19, pp. 1128-1137, Dec. 1971.
- [15] J. Omura, " On the Viterbi Decoding Algorithm ", *IEEE Trans. Inform. Theory*, vol. IT-15. pp. 177-179, Jan. 1969.
- [16] H. Kobayashi, *Modeling and Analysis : An Introduction to System Performance Evaluation Methodology* , Addison-Wesley Pub. Co. Inc., 1978.
- [17] D.E. Knuth, *The Art of Computer Programming. : Vol. 2 : Seminumerical Algorithms*, Reading Mass: Addison-Wesley.
- [18] A. Viterbi and J. Omura *Principles of Digital Communications and Coding*, McGraw-Hill, New York, 1979.
- [19] A. Viterbi, " Error Bounds for Convolutional Codes and an Asymptotic Optimum Decoding Algorithm ", *IEEE Trans. Inform. Theory*, vol. IT-13. pp. 260-269, April 1967.
- [20] G. Ungerboeck, " Channel Coding with Multilevel Phase Signals ", *IEEE Trans. Inform. Theory*, vol. IT-28. pp. 177-179, Jan. 1982.
- [21] G.D. Forney, R. G. Gallager, G. R. Longstaff, S. V. Quereshi, " Efficient Modulation for Band-Limited Channels ", *IEEE Journal on Selec. area in Communications* , vol. SAC -2, Sept. 1984
- [22] G.D. Forney, " Coset Codes I: Binary Codes and Lattices ", *IEEE Communications Theory Workshop*, Palm Springs, Calif., April 28, 1986.

APPENDIX A

Error correcting capability of a convolutional code under symbol-by-symbol decoding.

The transfer function of the systematic code of fig. 1.6, with $d_{\min} = 6$ and correcting capability $C_c = 2$, is [11] :

$$T(D) = D^6 + 2D^8 + 4D^{10} + 8D^{12} + \dots$$

or

$$T(D, N) = ND^6 + 2N^2D^8 + 4N^3D^{10} + 8N^4D^{12} + \dots \quad (A-1)$$

where N is the number of information 1s in each path.

Symbol-by-symbol detection of the ω^{th} symbol, for $\omega = 1, \dots, K$, implies that we have to add all the metrics of the possible sequences with a 0 at the ω position, and compare this sum, denoted as $G(a_\omega = 0)$, with the respective one that represents the likelihood of the value 1 ($G(a_\omega = 1)$). If $G(a_\omega = 0) > G(a_\omega = 1)$, we decide that $a_\omega = 0$.

Without loss of generality we can assume that the all zero sequence of length K symbols was transmitted, thus :

Distribution of paths with $a_\omega = 0$

$\binom{K-1}{0}$ paths with 0 ones

$\binom{K-1}{1}$ paths with 1 ones

$\binom{K-1}{K-1}$ paths with $K-1$ ones

Distribution of paths with $a_\omega = 1$

$\binom{K-1}{0}$ paths with 1 ones

$\binom{K-1}{1}$ paths with 2 ones

$\binom{K-1}{K-1}$ paths with K ones

In order to find the distances of these paths, we use the transfer function of the code in eq. (A-1).

$\binom{K-1}{0}$ paths of distance D^0

$\binom{K-1}{0}$ paths of distance D^0

$\binom{K-1}{1}$ paths of distance D^0

$\binom{K-1}{1}$ paths of distance D^8

$\binom{K-1}{K-1}$ paths of distance $D^{4+2(k-1)}$

$\binom{K-1}{K-1}$ paths of distance $D^{0+2(k-1)}$

Assuming 2 channel errors to occur, the worst case should be the one in which all the distances of the paths in the correct subset are increased by 2 and those of the incorrect subset are decreased by 2. Thus,

$\binom{K-1}{0}$ paths of distance D^2

$\binom{K-1}{0}$ paths of distance D^8

$\binom{K-1}{1}$ paths of distance D^8

$\binom{K-1}{1}$ paths of distance D^0

$\binom{K-1}{K-1}$ paths of distance D^{4+2k}

$\binom{K-1}{K-1}$ paths of distance D^{2+2k}

Now in order to decide that $a_w = 0$ (the correct decision in this case, since the all zero information sequence was transmitted), we should have: $G(a_w = 0) > G(a_w = 1)$. The metric value assigned to each possibly transmitted sequence in Hamming distance d from the received one is (see Sec. 1.2.1), α^d with $\alpha = \frac{p}{1-p}$ where p denotes the crossover probability of the BSC. The quantities $G(a_w = 0)$ and $G(a_w = 1)$, are the sums of these metrics that correspond to sequences with $a_w = 0$ and 1 respectively. Using the last distribution of the paths we get:

$$G(a_w = 0) > G(a_w = 1) \rightarrow$$

$$\alpha^2 + \sum_{i=1}^{K-1} \binom{K-1}{i} \alpha^{0+2i} > \alpha^4 + \sum_{i=1}^{K-1} \binom{K-1}{i} \alpha^{4+2i} \rightarrow$$

$$\alpha^0 + \sum_{i=1}^{K-1} \binom{K-1}{i} \alpha^{4+2i} > \alpha^2 + \sum_{i=1}^{K-1} \binom{K-1}{i} \alpha^{2+2i} \rightarrow$$

$$1 - \alpha^2 + \sum_{i=1}^{K-1} \binom{K-1}{i} \alpha^{2+2i} (\alpha^2 - 1) > 0 \rightarrow$$

$$(1 - \alpha^2) \left(1 - \sum_{i=1}^{K-1} \binom{K-1}{i} \alpha^{2+2i} \right) > 0$$

The first term of this product is positive for $p > 0.5$, the second one depends on K and α . Choosing a suitable value for the length K , the last expression guarantees that under symbol-by-symbol decoding, no detection errors will occur when the number of channel errors are less or equal to the error correcting capability of the code.

APPENDIX B

A first approach to the error performance of the symbol-by-symbol algorithm.

In reference [1], the optimal symbol-by-symbol algorithm was presented as a technique that uses symbol error probability as the optimality criterion in demodulating PAM signals corrupted by intersymbol interference. Although this algorithm is similar to the Viterbi algorithm, an analytical derivation of its error performance cannot be similar to the one proposed by Forney in [4], for reasons that are going to be explained here.

We use the same notation with that in [1], representing by L the number of values an information symbol a_i can assume, N the length of the information sequence, and m the channel's memory. The matched filter's output is

$$z_i = \sum_{j=-m}^m a_{i+j} r_j + n_i$$

with $r_{i-j} = 0$ for $|i-j| > m$ and n_i representing the additive Gaussian noise.

We also denote the system's state by

$$\sigma_k = \{ \hat{a}_{k-m+1}, \hat{a}_{k-m+2}, \dots, \hat{a}_k \} \quad k = m, m+1, \dots, N.$$

and a set of states or matched filter's outputs up to time kT as :

$$S^k = \{ \sigma_m, \sigma_{m+1}, \dots, \sigma_k \} = \{ \hat{a}_1, \hat{a}_2, \dots, \hat{a}_k \} \quad k \leq N.$$

and $z^k = \{z_1, z_2, \dots, z_k\}$ for $k \leq N$, respectively.

The metric assigned to a possibly transmitted sequence of length N is :

$$\exp [U(z^N, S^N)] = \exp [U(z^{N-1}, S^{N-1}) + V(z_N, \sigma_{N-1}, \sigma_N)] \quad (B-1)$$

where

$$U(z^k, S^k) = 2 \sum_{i=1}^k \hat{a}_i z_i - \sum_{i=1}^k \sum_{j=1}^k \hat{a}_i \hat{a}_j r_{i-j} \quad (B-2)$$

and

$$V(z_k, \sigma_{k-1}, \sigma_k) = 2 z_k \hat{a}_k - 2 \hat{a}_k \sum_{i=k-m}^{k-1} \hat{a}_i r_{k-i} - \hat{a}_k^2 r_0 \quad (B-3)$$

Maximization of the M.A.P criterion for the estimation of the ω^{th} symbol leads to the following expression :

$$\max_{\hat{a}_\omega} \Pr [a_\omega = \hat{a}_\omega | z(t), 0 < t < \tau] = c \max_{\hat{a}_\omega, \sigma_{m1}, \dots, \sigma_N} \sum \exp [U(z^N, S^N)] \quad (B-4)$$

This summation over all possible σ_{kl} means that in order to compute the likelihood of the event $a_\omega = l$, (l is one of a_ω 's L possible values) we assign a metric to every possible sequence of length N with an l at the ω^{th} position and add all these metrics. The same procedure is followed in order to compute the sum of the metrics of the paths having at the ω^{th} position a different value of a_ω . Finally we compare these sums corresponding to the different symbol values and choose their maximum as the optimal decision of our demodulator.

Alternately we could describe the previous procedure by means of set partitioning. Every possible path represented by its metric is an element of the set. We partition the set inspecting the ω^{th} position of every possible path and according to this value we form L subsets each of them corresponding to one of the L possible values of the symbol a_ω and having magnitude L^{N-1} . Finally we sum the elements -metrics of each subset, chose their maximum, say l , and decide that $a_\omega = i$. The same procedure is repeated for every symbol of the sequence ($\omega : 1, \dots, N$) until the estimation of the symbol a_N .

The metric of a sequence of N symbols was previously defined as :

$$\exp [U(z^N, S^N)] = \exp [U(z^{N-1}, S^{N-1}) + V(z_N, \sigma_{N-1}, \sigma_N)] \quad (B-5)$$

Seeking a clarification of the ω^{th} symbol's contribution to this metric we represent by

$$t_\omega^l = \{ a_{\omega-m}, \dots, a_{\omega-1}, l, a_{\omega+1}, \dots, a_{\omega+m} \} \quad (B-6)$$

and

$$t_\omega = \{ a_{\omega-m}, \dots, a_{\omega-1}, a_{\omega+1}, \dots, a_{\omega+m} \} \quad (B-7)$$

Then eq.(B-5) becomes :

$$\exp [U(z^N, S^N)] = \exp \left[2 \sum_{\substack{i=1 \\ i \neq \omega}}^N a_i z_i - \sum_{\substack{i=1 \\ i \neq \omega}}^N \sum_{\substack{j=1 \\ j \neq \omega}}^N a_i a_j r_{i-j} + 2 z_\omega a_\omega - 2 a_\omega \sum_{\substack{i=\omega-m \\ i \neq \omega}}^{\omega+m} a_\omega r_{\omega-i} - a_\omega^2 r_0 \right].$$

or, using eq. (B-6)

$$\exp [U(z^N, S_l^N)] = \exp [U'(z^N, S_l^N) + V'(z_\omega, t_\omega^l)].$$

where

$$U'(z^N, S_l^N) = 2 \sum_{\substack{i=1 \\ i \neq \omega}}^N \hat{a}_i z_i - \sum_{\substack{i=1 \\ i \neq \omega}}^N \sum_{\substack{j=1 \\ j \neq \omega}}^N \hat{a}_i \hat{a}_j r_{i-j}$$

$$V'(z_\omega, t_\omega^l) = 2 z_\omega \hat{a}_\omega - 2 \hat{a}_\omega \sum_{\substack{i=\omega-m \\ i \neq \omega}}^{\omega+m} \hat{a}_\omega r_{\omega-i} - \hat{a}_\omega^2 r_0$$

Now, in order to compute the likelihood of the symbol value $\hat{a}_\omega = l$ in eq.

(B-4) we follow the next procedure :

(I) we set initial values to the sequence of length $2m + 1$ around the ω^{th} symbol (we previously defined it as t_ω^l). For example if binary PSK is the case and $m = 2$, these values can be

$$\begin{matrix} -1 & -1 & l & -1 & -1 \end{matrix} \quad \text{where } l = +1 \text{ or } -1.$$

(II) we compute the metrics of the paths which include this already set sequence t_ω^l , represented by $S_l^N \in t_\omega^l$ and sum them up. Their sum is :

$$\sum_{S_l^N \in t_\omega^l} \exp [U'(z^N, S_l^N) + V'(z_\omega, t_\omega^l)].$$

Noting that having set the values of t_ω^l , $V'(z_\omega, t_\omega^l)$ is a common factor to all realizations of S_l^N and the last expression can be written as :

$$\exp [V'(z_\omega, t_\omega^l)] \sum_{S_l^N \in t_\omega^l} \exp [U'(z^N, S_l^N)].$$

(iii) set other values to the sequence t_ω^l , and repeat the same procedure until we reach the final realization of the sequence t_ω^l .

(iv) The summation of the sums computed in steps (ii) and (iii), is the metric we assign to the symbol value $a_\omega = l$. That is :

$$M(a_\omega = l) = \sum_{\text{all } t_\omega} \exp [V' (z_\omega, t_\omega^l)] \sum_{\text{all } S_i^N \in t_\omega^l} \exp [U' (z^N, S_i^N)].$$

(v) we set another symbol value to ω^{th} position, say $a_\omega = \mu$, and the previous steps are repeated for each $\mu \neq l$.

Defining

$$S_i = \sum_{S_i^N \in t_\omega^l} \exp [U' (z^N, S_i^N)]. \quad (\text{B-8a})$$

and

$$V_i^l = \exp [V' (z_\omega, t_\omega^l)] \quad (\text{B-8b})$$

the likelihood of $a_\omega = l$, expressed by $M(a_\omega = l)$, takes the following form :

$$M(a_\omega = l) = \sum_{i=1}^{L^{2m-1}} S_i V_i^l \quad (\text{B-9a})$$

Similarly for every $\mu \neq l$

$$M(a_\omega = \mu) = \sum_{i=1}^{L^{2m-1}} S_i V_i^\mu \quad (\text{B-9b})$$

since S_i is independent of a_w . We decide that $a_w = l$ if $M(a_w = l) > M(a_w = \mu)$ for all $\mu \neq l$.

Now suppose that l was actually sent, but we take a wrong decision and choose μ as the correct value of a_w . Then the probability of a single error to occur is :

$$Pr (\text{a single symbol error} / a_w = l) = Pr \{ M(a_w = \mu) > M(a_w = l) \}$$

Using eq. (B-9) the last expression becomes

$$Pr \left\{ \sum_{i=1}^{L^{2m-1}} S_i V_i^{\mu} > \sum_{i=1}^{L^{2m-1}} S_i V_i^l \right\}$$

or

$$Pr \left\{ \frac{\sum_{i=1}^{L^{2m-1}} S_i V_i^{\mu}}{\sum_{i=1}^{L^{2m-1}} S_i V_i^l} > 1 \right\} \quad (B-10)$$

In order to proceed we use the following inequality whose proof is given in appendix C.

$$\frac{\sum_i a_i X_i}{\sum_i a_i Y_i} \leq \sum_i \left(\frac{X_i}{Y_i} \right) \quad \text{for } a_i, X_i, Y_i \geq 0 \quad (B-11)$$

the probability in eq. (B-10) is upper-bounded by

$$Pr \left\{ \frac{\sum_{i=1}^{L^{2m-1}} S_i V_i^{\mu}}{\sum_{i=1}^{L^{2m-1}} S_i V_i^l} > 1 \right\} \leq Pr \left\{ \sum_{i=1}^{L^{2m-1}} \frac{V_i^{\mu}}{V_i^l} > 1 \right\} \quad (B-12)$$

For a certain l , say k ,

$$V_k^l = e^{2z_{\omega}^l - 2l \sum_{i \neq \omega} a_i r_{\omega-1} - l^2 r_0}$$

$$V_k^{\mu} = e^{2z_{\omega}^{\mu} - 2\mu \sum_{i \neq \omega} a_i r_{\omega-1} - \mu^2 r_0}$$

The values a_j are defined by the sequence t_{ω} thus are the same in both these expressions. Then, each ratio in eq.(B-12) has the following form :

$$e^{2(l-\mu)z_{\omega} - 2(l-\mu) \sum_{i \neq \omega} a_i r_{\omega-1} - (l^2 - \mu^2)r_0} \quad (B-13)$$

Using eq. (B-12) and (B-13)

$$\begin{aligned} & Pr \left\{ \sum_{t_{\omega}} e^{2(l-\mu)z_{\omega} - 2(l-\mu) \sum_{i \neq \omega} a_i r_{\omega-1} - (l^2 - \mu^2)r_0} > 1 \right\} \\ &= Pr \left\{ e^{2(l-\mu)z_{\omega} - (l^2 - \mu^2)r_0} \cdot \sum_{t_{\omega}} e^{-2(l-\mu) \sum_{i \neq \omega} a_i r_{\omega-1}} > 1 \right\} \end{aligned}$$

taking the logarithms of both sides :

$$= Pr \left\{ 2(l-\mu)z_{\omega} - (l^2 - \mu^2)r_0 + \ln \left(\sum_{t_{\omega}} e^{-2(l-\mu) \sum_{i \neq \omega} a_i r_{\omega-1}} \right) > 0 \right\}$$

or equivalently :

$$= Pr \left\{ 2(l-\mu)z_{\omega} > (l^2 - \mu^2)r_0 - \ln \left(\sum_{t_{\omega}} e^{-2(l-\mu) \sum_{i \neq \omega} a_i r_{\omega-1}} \right) \right\} \quad (B-14)$$

Denoting by k the second part of eq. (B-13), this expression becomes the proba-

bility of a Gaussian random variable to be greater than k where k is a constant defined by the difference $(l - \mu)$ and a combination the quantiles a_i, r_i .

Substituting the value of z_w

$$z_w = \sum_{i \neq w} a_i r_{i-w} + l r_0 + n_w$$

Into eq. (B-14) we have :

$$Pr \{ 2(l - \mu) \left[\sum_{j \neq w} a_j r_{j-w} + l r_0 + n_w \right] > (l^2 - \mu^2) r_0 - \ln \left(\sum_{l_w} e^{-2(l - \mu) \sum_{i \neq w} a_i r_{i-w}} \right) \}$$

Defining

$$\ln \left(\sum_{l_w} e^{-2(l - \mu) \sum_{i \neq w} a_i r_{i-w}} \right) = C_1$$

the last expression becomes :

$$\begin{aligned} & Pr \{ 2(l - \mu)n_w + 2(l - \mu) \sum_{j \neq w} a_j r_{j-w} + 2l^2 r_0 - 2\mu l r_0 - l^2 r_0 + \mu^2 r_0 > -C_1 \} \\ &= Pr \{ 2(l - \mu)n_w + 2(l - \mu) \sum_{j \neq w} a_j r_{j-w} + (l - \mu)^2 r_0 > -C_1 \} \\ &= Pr \{ 2(l - \mu)n_w > -[2(l - \mu) \sum_{j \neq w} a_j r_{j-w} + (l - \mu)^2 r_0 + C_1] \} \quad (B-15) \end{aligned}$$

Representing by C_2 the quantity :

$$C_2 = -[2(l - \mu) \sum_{j \neq w} a_j r_{j-w} + (l - \mu)^2 r_0 + C_1]$$

and noting that $2(l - \mu)n_w$ is a Gaussian random variable with zero mean and

variance σ^2 , eq. (B-15) becomes :

$$Pr \{ 2 (1 - \mu) n_w > C_2 \} = \frac{1}{2} \operatorname{erfc} \left(\frac{C_2}{\sqrt{2}\sigma} \right) \quad (\text{B-16})$$

where

$$\operatorname{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^{\infty} e^{-t^2} dt$$

The previous bound can be further generalized to the case where the error vector has more than one non-zero coefficients. The main limitation of the previous analysis is the tightness of the established upper bound. Simulation results have shown that the value computed in eq.(B-16) is close to one. Since it constitutes the most important factor in bounding the probability of a symbol error in the performance analysis presented in [4] †, a different approach that could lead to a tighter upper bound should be further considered.

APPENDIX C

Proof of Equation (B-11).

We prove here by induction that :

$$\frac{\sum_i a_i x_i}{\sum_i a_i y_i} \leq \sum_i \left(\frac{x_i}{y_i} \right) \quad \text{for } a_i, x_i, y_i \geq 0$$

1). For $k = 2$

$$\frac{ax_1 + bx_2}{ay_1 + by_2} \leq \frac{ax_1}{ay_1} + \frac{bx_2}{by_2} = \frac{x_1 y_2 + x_2 y_1}{y_1 y_2}$$

clearing fractions :

$$ax_1 y_1 y_2 + bx_2 y_1 y_2 \leq ax_1 y_1 y_2 + ax_2 y_1^2 + bx_1 y_2^2 + bx_2 y_1 y_2$$

or

$$0 \leq ax_2 y_1^2 + bx_1 y_2^2$$

which is valid since $a, b, x_i, y_i \geq 0$ for $i = 1, 2$.

11). Assuming that the inequality holds for k , we have :

$$\frac{ax_1 + bx_2 + \dots + kx_k}{ax_1 + bx_2 + \dots + ky_k} \leq \frac{ax_1}{ay_1} + \frac{bx_2}{by_2} + \dots + \frac{kx_k}{ky_k}$$

III). We prove that it also holds for $k = k + 1$, i.e.,

$$\frac{ax_1 + bx_2 + \dots + kx_k + jx_{k+1}}{ay_1 + by_2 + \dots + ky_k + jy_{k+1}} \leq \frac{ax_1}{ay_1} + \frac{bx_2}{by_2} + \dots + \frac{kx_k}{ky_k} + \frac{jx_{k+1}}{jy_{k+1}}$$

The left part of the previous expression can be written as :

$$\frac{[ax_1 + bx_2 + \dots + kx_k] + jx_{k+1}}{[ay_1 + by_2 + \dots + ky_k] + jy_{k+1}} \leq \frac{ax_1}{ay_1} + \frac{bx_2}{by_2} + \dots + \frac{kx_k}{ky_k} + \frac{jx_{k+1}}{jy_{k+1}}$$

because of the combination of the steps (I). and (II).