TREE CODING OF ANALOG DATA SOURCES

TREE CODING OF ANALOG DATA SOURCES

By

JOHN BRUCE BODIE, B.Sc.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Master of Engineering

McMaster University

April 1974

MASTER OF ENGINEERING (1974) McMASTER UNIVERSITY
(Electrical) Hamilton, Ontario

TITLE: Tree Encoding of Analog Data Sources

AUTHOR: John Bruce Bodie, B.Sc. (University of Manitoba)

SUPERVISORS: Dr. S. S. Haykin

Dr. J. B. Anderson

NUMBER OF PAGES: viii, 90

ABSTRACT

Concepts of tree coding and of rate-distortion theory are applied to the problem of the transmission of analog signals over digital channels.

Coding schemes are developed which yield improvements of up to six dB in signal-to-noise ratio over conventional techniques for the reproduction of speech waveforms.

The author wishes to thank Dr. S. S. Haykin, Director of the Communications Research Laboratory, for providing the necessary computing facilities; and Dr. J. B. Anderson, who suggested this area of research and whose continued advice and support have been invaluable.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

This thesis addresses the problem of reproducing analog signals as faithfully as possible on the basis of information transmitted over a digital channel of given capacity. We show that the systems conventionally employed for this purpose fall short of the performance theoretically attainable, and present a method by which performance considerably closer to the optimum can be obtained.

Summary Of Chapters

In chapter 2, we define more precisely the problem we consider, introducing the concept of a fidelity criterion to permit the use of rate-distortion theory to establish the ultimate performance limit for a given source and channel. Tree source coding is developed as an instrumentable method of approaching the rate-distortion limit, and the design of a good tree coding system is shown to comprise the selection of an appropriate tree code and an effective search algorithm.

Chapter 3 discusses several search algorithms including the M-algorithm, which is of central importance to this work. In chapter 4, the performance of these search algorithms in a somewhat artificial, but easily analyzed, situation is compared.

The treatment of actual data sources begins in chapter 5, where the search algorithms are applied to several tree codes, both idealized and practical, in an attempt to reproduce an uncorrelated Gaussian source.

Chapter 6 develops appropriate tree codes for use with correlated sources; in chapter 7, these codes are applied to the reproduction of sampled speech waveforms, and the improvement which the M-algorithm yields over conventional systems employing these codes is evaluated.

CHAPTER II

RATE-DISTORTION THEORY AND TREE SOURCE CODING

We are concerned here with the problem of transmitting a signal, generated by some source, over a communication channel whose capacity is less than the informational entropy of the source. We know from standard information theory that the signal cannot be reproduced perfectly; some distortion must result. If we define a fidelity criterion which measures the quality of reproduction, rate-distortion theory [1] specifies how well the best possible communication system employing the given channel performs.

To approach the rate-distortion performance limit, it is usually necessary to employ source encoding to match the source to the channel. The process of source encoding essentially selects for transmission those features of the signal which are most important (in the fidelity criterion sense) for its reproduction, and prevents the use of channel capacity for the transmission of redundancies or of unimportant information. In a communication system employing source encoding (see Fig. 2.1), the source encoder accepts the input signal X, with entropy $H_X > C$ (the channel capacity), and produces the transmitted signal Y, with entropy $H_Y < C$, in a form acceptable to the channel. If we assume an error-free channel, as should be the case given suitable channel coding, the source decoder receives Y unchanged, and generates the re-

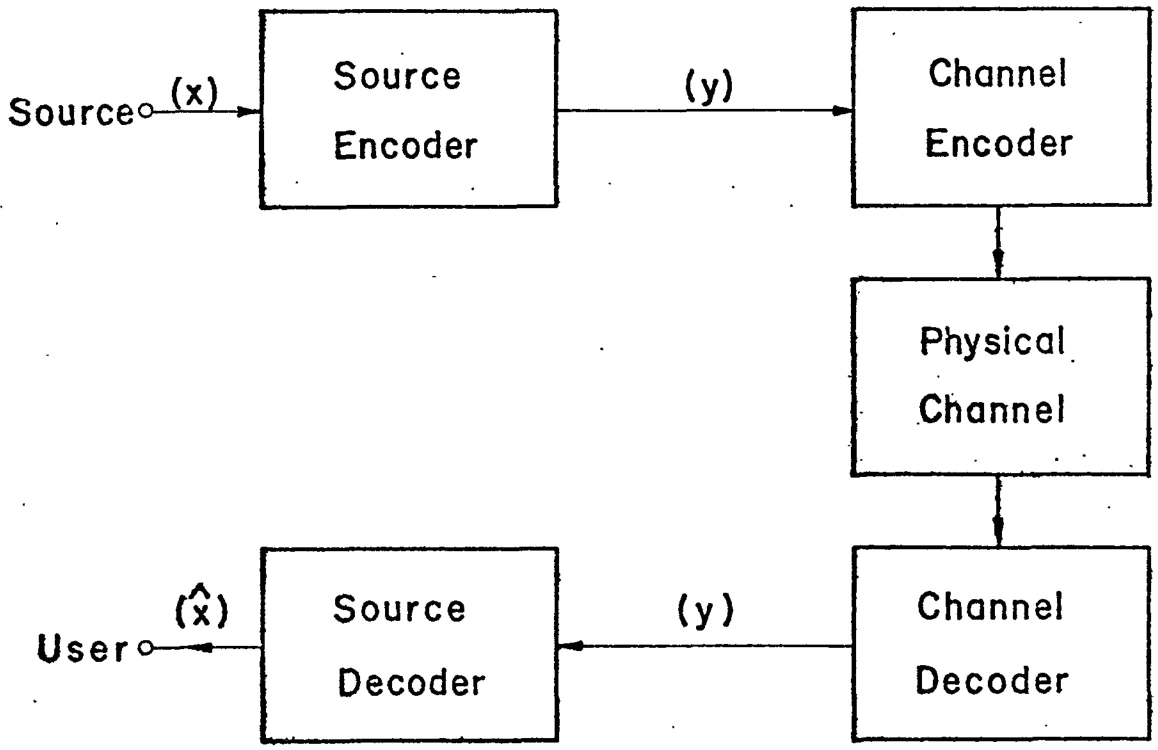


Fig. 2.1 Communication System Block Diagram

produced signal $\hat{X}$, an approximation to X, with entropy $H_{\hat{X}} < C$.

The sources with which we are concerned generate sequences of letters chosen from some <u>source alphabet</u>, which may be a set of discrete symbols, a segment of the real line, or some other alphabet. We will denote a single source letter as x_t and a sequence of these as $\{x\}$. We assume a digital channel capable of the error-free transmission from encoder to decoder of one channel digit y_t, chosen from an alphabet of d digits, for each source letter accepted by the encoder. The capacity of the channel is then $R=\log_2(d)$ bits/letter. R is also known as the <u>rate</u> of the source code used. The decoder generates one letter $\hat{x}$, chosen from a <u>reproducing alphabet</u>, to correspond to each source letter.

In a practical situation, the source letters would typically be produced by sampling a continuous time function at uniform intervals $T_s=1/f_s$. For the purpose of theoretical analysis, however, it is assumed that the source letters are chosen probabilistically according to a specified distribution function.

We have introduced the concept of a fidelity criterion which somehow measures the resemblence between $\{x\}$ and $\{\hat{x}\}$. To employ rate-distortion theory, the fidelity criterion must be a well-defined function $F_N(\{x\},\{\hat{x}\})$, where the subscript indicates the length of the sequences compared. The most easily analyzed fidelity criteria are the <u>single-letter</u> criteria, for which $F_N(\{x\},\{\hat{x}\})=\frac{1}{N}\ F(x_t,\hat{x}_t)$. Here $F(x_t,\hat{x}_t)$ measures the distortion inherent in the reproduction of the single source letter x_t as $\hat{x}_t$, and the fidelity criterion for a sequence of length N is just the average distortion in the reproduction of the N letters of the sequence.

In the analog domain, $F(x_t,\hat{x}_t)$ is often a <u>difference-distortion</u> measure; that is, $F(x_t,\hat{x}_t) = F(x_t-\hat{x}_t)$. The distortion then depends only on the difference between the source and reproducing letters, and not on their actual values. A popular difference-distortion measure is $F(x_t,\hat{x}_t) = |x_t-\hat{x}_t|^k$. With $k=2$, this is the squared-error distortion measure, leading to the mean-squared error or MSE single-letter fidelity criterion.

The fidelity criterion should, as far as possible, be chosen to reflect the quality of reproduction experienced by the ultimate user of the transmitted information. Unfortunately, in many cases (and particularly when the source is a sampled speech waveform) the user's actual quality measurement criterion has not been expressed in a form suitable for the application of rate-distortion theory or for use by the source coding techniques we will present. In such cases, we resort to a criterion which, while not identical to that of the user, bears some relation to the subjective quality of reproduction. The MSE criterion is often chosen for the simplicity of analysis it permits.

As an example of a rate-distortion performance limit, consider the following: we wish to reproduce letters chosen independently from the normal distribution (μ,σ^2) by means of the communication system of Fig. 2.1 . The quality of reproduction is judged by the MSE fidelity criterion. Rate-distortion theory states [1] that $\Delta(R)$, the minimum distortion attainable when R bits/letter of information about the source are transmitted, is given by

$$\Delta(R) = \frac{\sigma^2}{2^{2R}} \tag{2.1}$$

Unfortunately, the theory does not specify how to construct a practical source coder which will acheive this level of performance. We next consider some source coding techniques which can perform this well if the encoder is allowed an infinite amount of storage and an infinite length of time to perform its task, and show how these techniques can be used to approach the rate-distortion limit in a practical system.

Block Codes

Block codes are a very general type of source code; properly applied, they can achieve the rate-distortion performance limit. A block code of blocklength N and rate R consists of a table or code book of 2^{NR} unique code words, each an N-tuple of letters from the reproducing alphabet. The encoder accepts a block of N input samples, finds the code word best matching this block, and transmits to the decoder in the form of N channel digits (or NR bits) the identity of the selected code word. The decoder looks up the code word in its code book and releases to the user the specified N letters of the output sequence. A suitably-chosen block code will approach $\Delta(R)$ as $N \to \infty$.

The code used by the standard pulse-code modulation (PCM) source coding system is a block code with N=1: each input sample is quantized independently to one of the d letters of the reproducing alphabet.

The number of code words in a block code increases exponentially with blocklength, implying that the storage (of code words) and computation (comparisons of the input block with the various code words) required to implement the code do likewise. It is therefore impossible for an encoder of finite speed and storage capacity to employ a block code of arbitrarily large N, as is necessary if the system is to approach $\Delta(R)$ arbitrarily closely. Such a code is termed <u>non-instrumentable</u>.

<u>Tree Codes</u>

A type of block code which will eventually lead to instrumentable coding schemes is the tree code. In a tree code, the code words cannot be chosen independently, but possess a particular structure, best described with reference to the <u>code tree</u> of Fig. 2.2 . A code tree consists of a number of nodes arranged in levels. From each node at a given level, α branches extend to nodes at the next level. No two branches terminate on the same node, and there is thus a unique path consisting of a connected series of branches from the root of the tree (the one node at level 0) to a given node. Associated with each branch of the tree are β branch letters chosen from the reproducing alphabet.

The sequence of branch letters encountered in following a

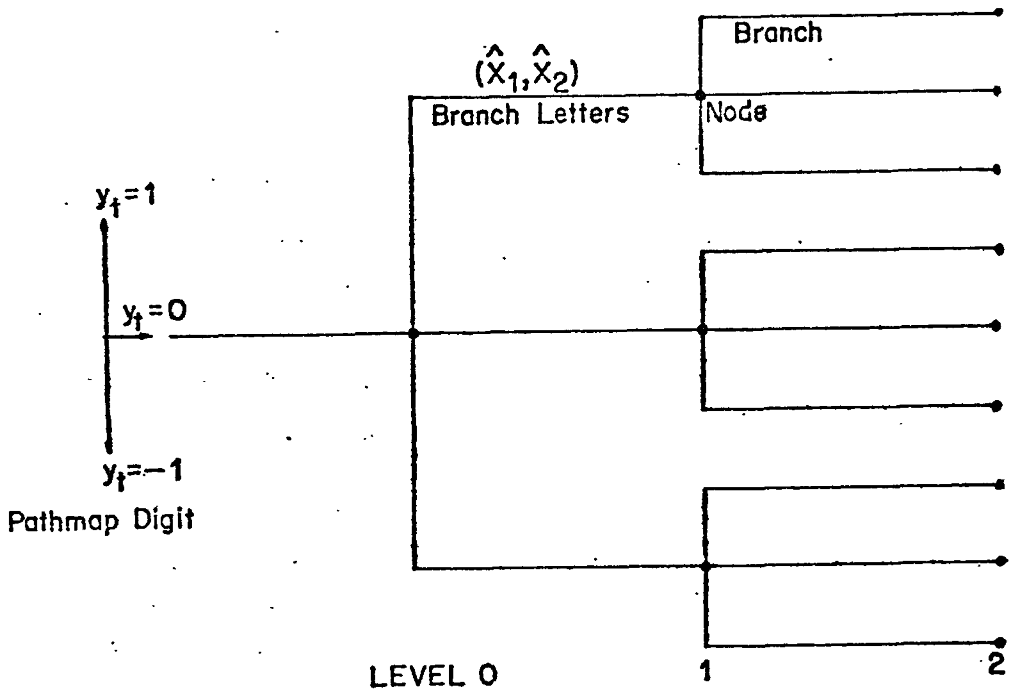


Fig. 2.2 Code Tree: $\alpha=3$, $\beta=2$

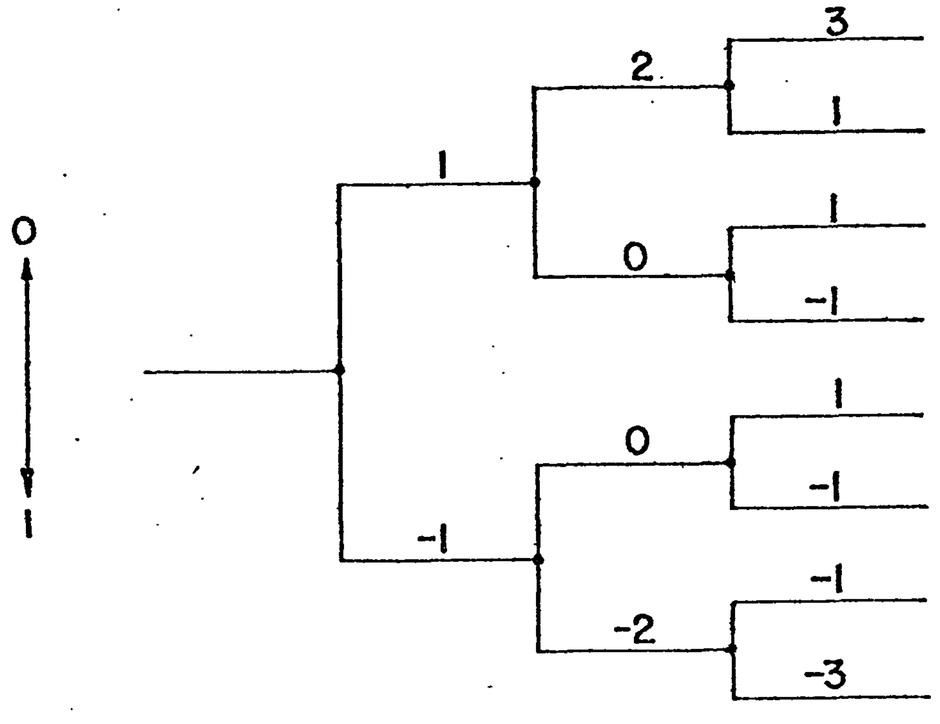


Fig. 2.3 Code Tree of Linear Delta Modulation

path through the entire tree forms a code word. If the tree is L levels deep, the corresponding tree code has α^L code words of blocklength $N=\beta L$. We will assume that each branch contains only one letter and that $\alpha=2^R=d$. It has been shown [1] that a suitable tree code, in the limit $L\to\infty$, achieves $\Delta(R)$.

As with the block codes, the channel digits received by the decoder specify the code word to be generated. While block decoding is just a table look-up procedure, tree decoding can be descibed geometrically: the decoder traces a path through the code tree, with each channel digit specifying which of the d branches extending from the node last reached is to be followed to a node at the next level. As they define, or map, a path through the tree, the channel digits are also referred to as path map digits. The encoder decides on the path map digits to be transmitted to the decoder by evaluating the fidelity criterion for the code words produced by various paths. This process is referred to as searching the code tree.

Tree codes as presented are still non-instrumentable; while the amount of storage required to implement a tree code is somewhat less that that for a block code of equal blocklength, due to the fact that the code words of a tree code share certain letters, it nonetheless increases exponentially with N. If, however, the decoder can generate branch letters as it requires them from the path map digits received, the code book can be entirely eliminated. As an example, consider a tree (with d=2) in which the value of a particular branch letter is just the sum of the path map digits defining the path leading to the corresponding branch. Fig. 2.3 illustrates such a tree for L=3, with the two possible path map digit values assumed to be +1 and -1. A decoder employing a code of this type need not store any branch letters, regardless of blocklength. If the coding system is to perform well, of course, it must be capable of generating code words appropriate for the reproduction of any input block the source may produce. (It is easily seen that the code just described is that used by linear delta modulation, which is well suited to sources whose successive letters are highly correlated and therefore do not differ greatly from one letter to the next.)

The elimination of the code book is not in itself a guarantee of instrumentability, as the problem of exponentially increasing computation remains. An encoder employing the code tree of Fig. 2.3 would still have to compare a given input block with every code word to ensure finding the best match. To reduce the required computation to the level of instrumentability, it is necessary to abandon this exhaustive search of the code tree in favour of a selective search algorithm which considers only the most promising portions of the code tree.

The design of an effective instrumentable tree coding system thus involves two problems; the choice of a good code which can be generated from path map digits, and the choice of an efficient search algorithm.

CHAPTER III

SEARCH ALGORITHMS

In discussing search algorithm operation and performance it is convenient to introduce a second tree structure, called the metric tree. The metric tree differs from the code tree in that the letter (called the path metric) associated with a given branch is not a member of the reproducing alphabet, but is the value of the fidelity criterion evaluated for a given input sequence over the output sequence generated by the path leading to that branch. Choosing the best code word for the reproduction of an input block is equivalent to finding the path through the metric tree which leads to the branch at the final level of the tree with the best path metric.

An exhaustive search will always find this path; as we have seen, however, such a search is feasible only for tree codes of short blocklength. To allow longer blocklengths, we are forced to adopt a non-exhaustive search algorithm. Such algorithms basically rely on the assumption that good paths tend to remain good, and poor to remain poor. For example, if a particular path has a very poor path metric partway through the tree, the search algorithm might well decide that none of the deeper extensions of this path is likely to be very good, and so choose not to explore the portion of the tree stemming from it.

The Random Search

This most extreme of the non-exhaustive search algorithms involves no searching whatever; the encoder simply chooses a path at random, or equivalently transmits randomly-chosen channel digits. Useless in practice, the random search merely serves as a limiting case against which to compare the performance of a more intelligent algorithm.

The Single-Path Search

The simplest search algorithm of any practical value is the single-path search. Having reached a node at some level, the algorithm examines the path metrics of the d branches extending from that node and follows the best branch to a node at the next level. This process is repeated until the end of the tree is reached; the path map digits defining the one path which the algorithm has traced through the tree are then transmitted to the decoder.

We define W, the computational work performed by a search algorithm, as the average number of branches whose path metrics must be examined to encode a source letter. For the single-path algorithm, $W = d$, regardless of the blocklength of the code. As described, however, the single-path search is non-instrumentable; N path map digits must be stored for transmission when the end of the tree is reached. A simple modification of the algorithm removes this difficulty. Once the algorithm has selected a branch at a given level, it is certain that the finally chosen path will include this branch, and the channel digit defining the selected branch can be released to the decoder immediately without compromising the action of the algorithm at future levels. No channel digit storage is then required, and the single-path algorithm, coupled with a code whose branch letters can be calculated from received path map digits, is fully instrumentable. In fact, this search algorithm forms the basis of the PCM, differential PCM (DPCM), and delta modulation systems in common use.

The major failing of the single-path search is that it completely disregards the possibility that the best path to level L may not extend from the best path to an earlier level. The multi-path search algorithms try to take this possibility into account.

Before leaving the single-path algorithm, it should be mentioned that, with certain code trees, the best path to level L must indeed extend from the best to any previous level, and a single-path search will therefore perform just as well as an exhaustive search. Such a tree is that used by the simple quantization (PCM) type of source coder. We have said that this system employs a block code of blocklength 1; it is also equivalent to a tree source coder employing the single-path algorithm to search a tree of infinite blocklength which populates the d branches extending from each node with the same set of branch letters. Because the tree possesses this repetitive structure, the same set of output sequences is available from every node at a particular level, and the best path through the entire tree must extend from the best to that level.

Multi-Path Searches

A simple multi-path search algorithm is the (M,1)-algorithm, or simply the M-algorithm [2]. This algorithm proceeds through the metric tree along M paths of equal length. At any level in the tree, there will be M nodes, called saved nodes, to which these paths have led. The algorithm examines the path metrics of the dM branches extending from the saved nodes, and follows the M best of these branches to nodes at the next level. The process is repeated until the final level is reached, and the path map digits defining the best of the M paths through the entire tree are then transmitted to the decoder.

From the standpoint of computation, the M-algorithm is instrumentable; $W = dM$ for any blocklength. However, the problem of path map digit storage remains. Since in general none of the path map digits is decided upon until the end of the tree is reached, they cannot be transmitted synchronously with the input (that is, one digit transmitted as each source letter is accepted) as was done with the single-path algorithm. To allow a finite amount of path map digit storage to suffice for any blocklength, we can make use of a characteristic of the paths generated by the M-algorithm in searching many types of metric tree:

the M paths retained at a given level usually stem from a single node not too many levels back in the tree. This implies that the path map digits defining the path leading to this single node have already been decided upon, and could be transmitted to the decoder without affecting the future operation of the algorithm. We can therefore modify the algorithm as follows: the best branch of the dM examined at each level is found, and the path map digit l levels back along the path leading to this branch is transmitted. This limits the path map digit storage to a finite value $\leq lM$, for any blocklength. Of course, the M paths at some level will not necessarily always have a common predecessor node no more than l levels back. To allow for this, the algorithm must choose the M branches to be followed to the next level from only those of the dM examined which do stem from the node specified by the transmitted path map digit. (This requirement may in fact occasionally limit the number of branches available to less than M.) The modified algorithm is fully instrumentable, and operates synchronously with the input. Unlike the single-path algorithm, however, the channel digits transmitted (and therefore the decoder's reconstruction of the input sequence) lag behind the input by a fixed encoding delay of l samples.

Multi-path algorithms other than the M-algorithm have been proposed [3], [4], [5] . The attractiveness of the M-algorithm stems from the simplicity of its operation, the synchronous nature of its output (some algorithms require buffering to transmit a uniform stream of channel digits) and the relatively small amount of computation it requires to produce a useful improvement over the single-path search. It appears that the main area of interest for the M-algorithm is the region of low computation (i.e., small M), for other algorithms approach $\Delta(R)$ more closely if a high computational load is permitted. For a comparison of several algorithms in terms of the amount of computation required to approach $\Delta(R)$ to various tolerances, see Anderson[3].

CHAPTER IV

SEARCH ALGORITHM PERFORMANCE ON THE EXPONENTIAL METRIC TREE

The study of search algorithm performance is complicated by the fact that even the metric tree generated by a simple code tree and an elementary source letter probability distribution has a complex structure which makes analysis difficult. To permit the comparison of experimental results with theoretical predictions, we will first apply the search algorithms to a metric tree generated probabilistically from a distribution (of path metric values) chosen to facilitate analysis.

Specifically, we choose a binary (d = 2) tree employing a path metric which is the sum of metric increments μ, one for each branch in the path, chosen independently from the probability density

$$f_{\mu}(x) = e^{-x}u(x),$$

where u(x) is the unit step function. The corresponding distribution is

$$F_{\mu}(x) = (1-e^{-x})u(x).$$

An independent-increment metric tree, which makes analysis relatively simple, also corresponds to the practical situation of a symmetric source and metric [1].

In a less artificial situation, where the metric tree is generated from the interaction of an input sequence, a code tree, and a fidelity criterion, the average per-letter path metric can be identified with the average distortion in the reproduction of the input. We will make this identification here, denoting the average per-letter path metric as D, despite the fact that neither $\{x\}$ nor $\{\hat{x}\}$ appears explicitly.

The Random Search

Choosing branches at random yields an average distortion equal to the expected value of μ. That is,

$$D = E(\mu) = 1$$

The Single-Path Search

At each level, the algorithm chooses the branch which has received the smaller value of μ. The average increase in path metric from one level to the next is then the expected value of the minimum of two independent r.v.'s, each with density $f_\mu(x)$. The density function of this minimum is

$$\begin{aligned} f_m(x) &= 2f_\mu(x)(1-F_\mu(x)) \\ &= 2e^{-2x}u(x), \end{aligned}$$

the expected value of which is

$$D = 1/2$$

The Exhaustive Search

We have calculated (see Appendix A) the performance of an exhaustive search of this metric tree for blocklengths from one to seven, and have computer-simulated the same cases. As shown in Fig. 4.1, the theoretical and experimental values of average distortion agree very closely. The best performance is achieved with the longest blocklength, yielding

$$D_7 = .37 \ .$$

Plotted against log N (see Fig. 4.2), the distortion appears to decrease approximately linearly for $1 \leq N \leq 7$. Unfortunately, it was not practical to carry either the theoretical calculation or the simulation to values of $N > 7$. This prevents our examining the manner in which the distortion approaches a limiting value analogous to $\Delta(1)$ as $N \to \infty$.

The M-Algorithm

We have derived (see Appendix B) a theoretical prediction of the average distortion achieved by the M-algorithm with $M=2$. We have disregarded the problem of path map digit storage, effectively making l infinite. It should be possible to perform similar calculations for larger M, though even in the present somewhat artificial situation they would likely be arduous. We have simulated the algorithm for $1 \leq M \leq 50$; since we are not required here to produce a reconstruction of an input sequence, there is no need to store any path map digits, and l is again effectively infinite.

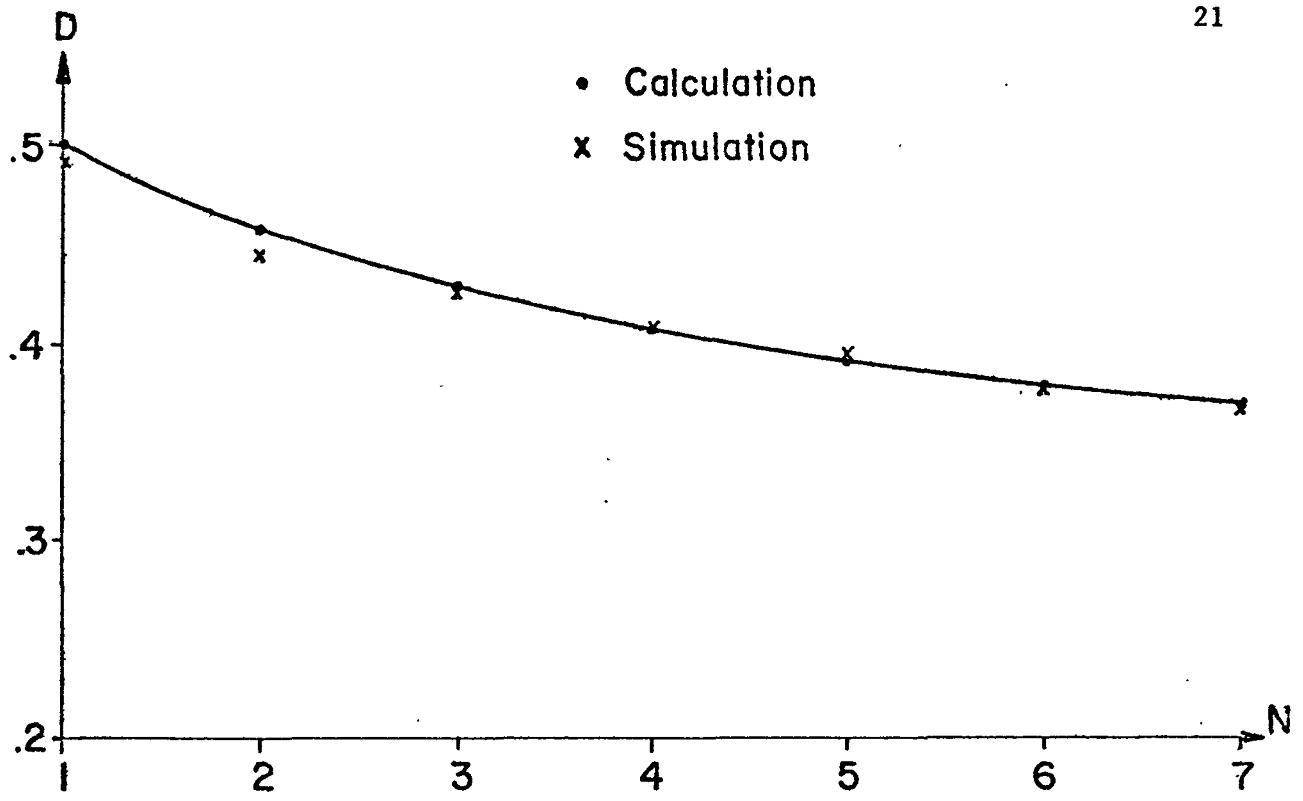


Fig. 4.1 Average Distortion vs. Blocklength: Exponential Metric Tree, Exhuastively Searched

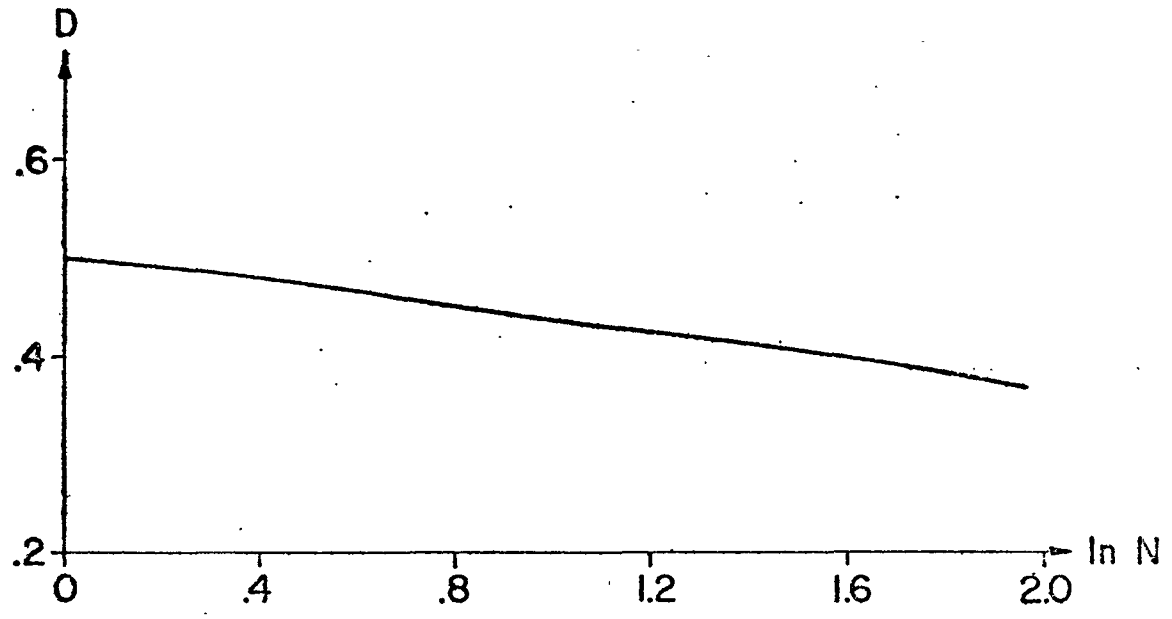


Fig. 4.2 Average Distortion vs. ln(N): Exponential Metric Tree, Exhaustively Searched

Fig. 4.3 shows the theoretical prediction and the results of the simulation. At $M = 1$, where the M-algorithm is equivalent to the single-path search, we find an experimental value of .49, close to the theoretical value of .5 . At $M \doteq 2$, the agreement between theory and experiment is again good: $D = .371$ (experimental) as compared to $D = .3675$ (theoretical). In the simulation, the distortion decreased monotonically with increasing M, reaching $D = .252$ at $M = 50$.

We have been unable to calculate $\Delta(1)$, the distortion attainable for this metric tree with infinite blocklength and the exhaustive search. We can, however, obtain an approximate value by extrapolating our M-algorithm results to infinite M, where the M-algorithm is equivalent to an exhaustive search. It has been found [6] that some search algorithms approach optimal performance in the manner

$$D - \Delta(R) \propto \frac{1}{(\ln C_1 W)^{C_2}} \quad \text{, as } W \to \infty \text{, and that}$$

for at least one algorithm $C_2 = 2$ [3]. Conjecturing that the M-algorithm also behaves in this way, we have plotted D vs. $(\ln M)^{-1}$ and D vs. $(\ln M)^{-2}$ (see Fig. 4.4); the first, equivalent to $C_1 = \frac{1}{d}$, $C_2 = 1$, does indeed appear to yield the desired linear relation. Extrapolating to infinite M, we obtain $\Delta(1) = .22$. As a further check on the assumed linear relation, we have plotted $\ln(D-.22)$ vs. $\ln(\ln(M))$ (see Fig. 4.5). The fact that the resulting curve is very nearly a straight line for the

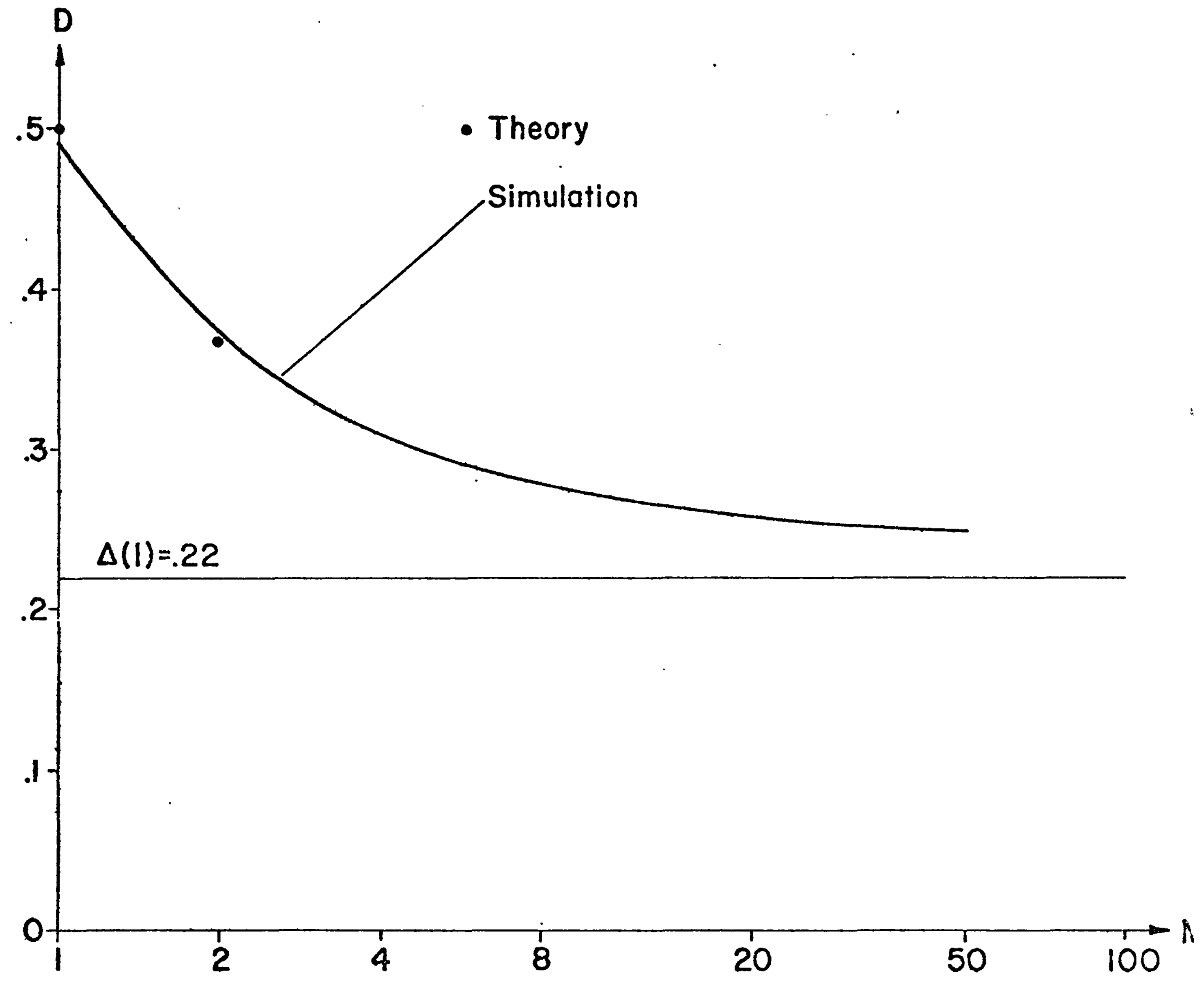


Fig. 4.3 Average Distortion vs. M: Exponential Metric Tree, Searched by the M-Algorithm

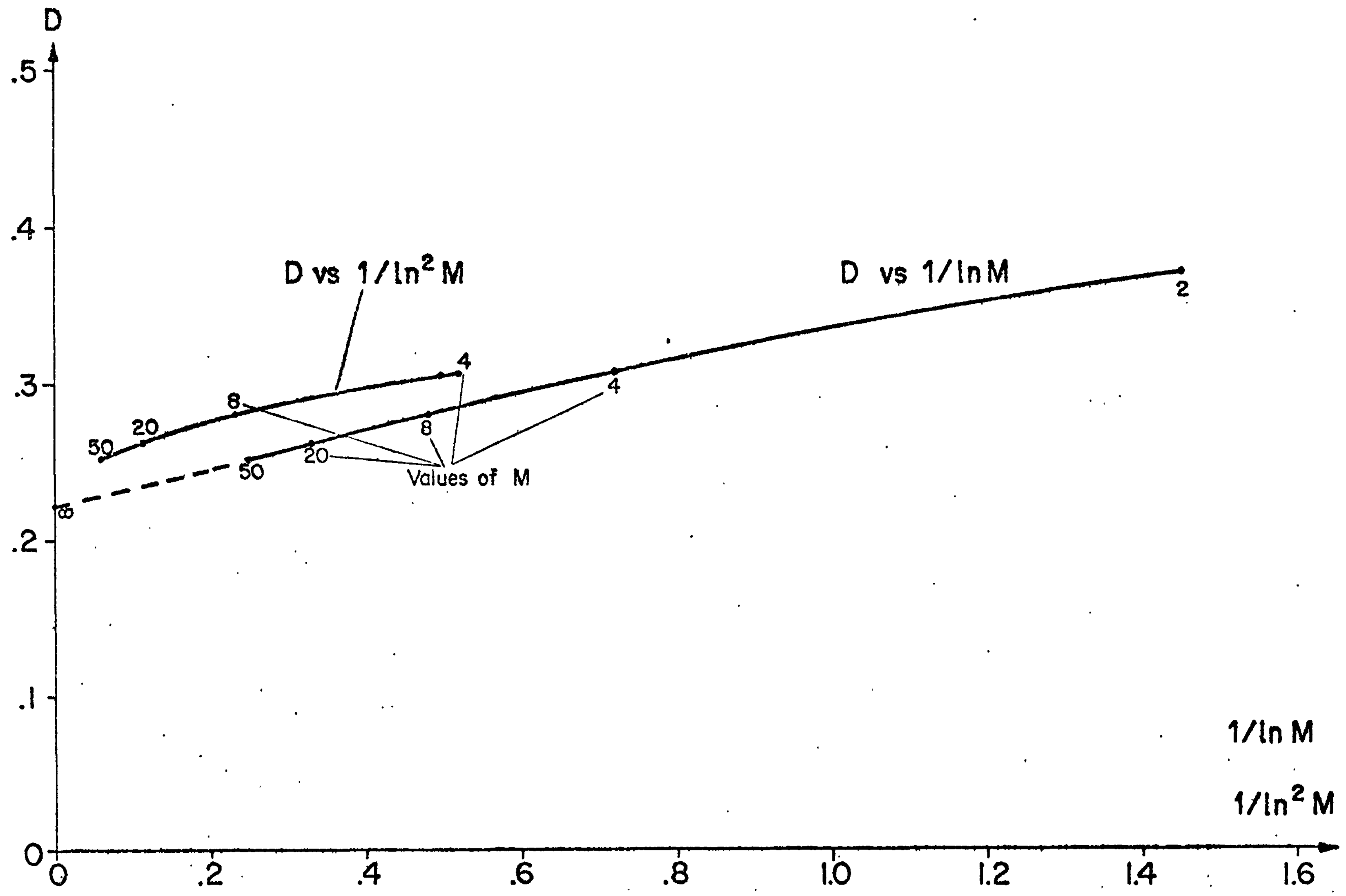


Fig. 4.4 Average Distortion vs. $1/\ln(M)$, $1/\ln^2(M)$: Exponential Metric Tree, Searched by the M-Algorithm

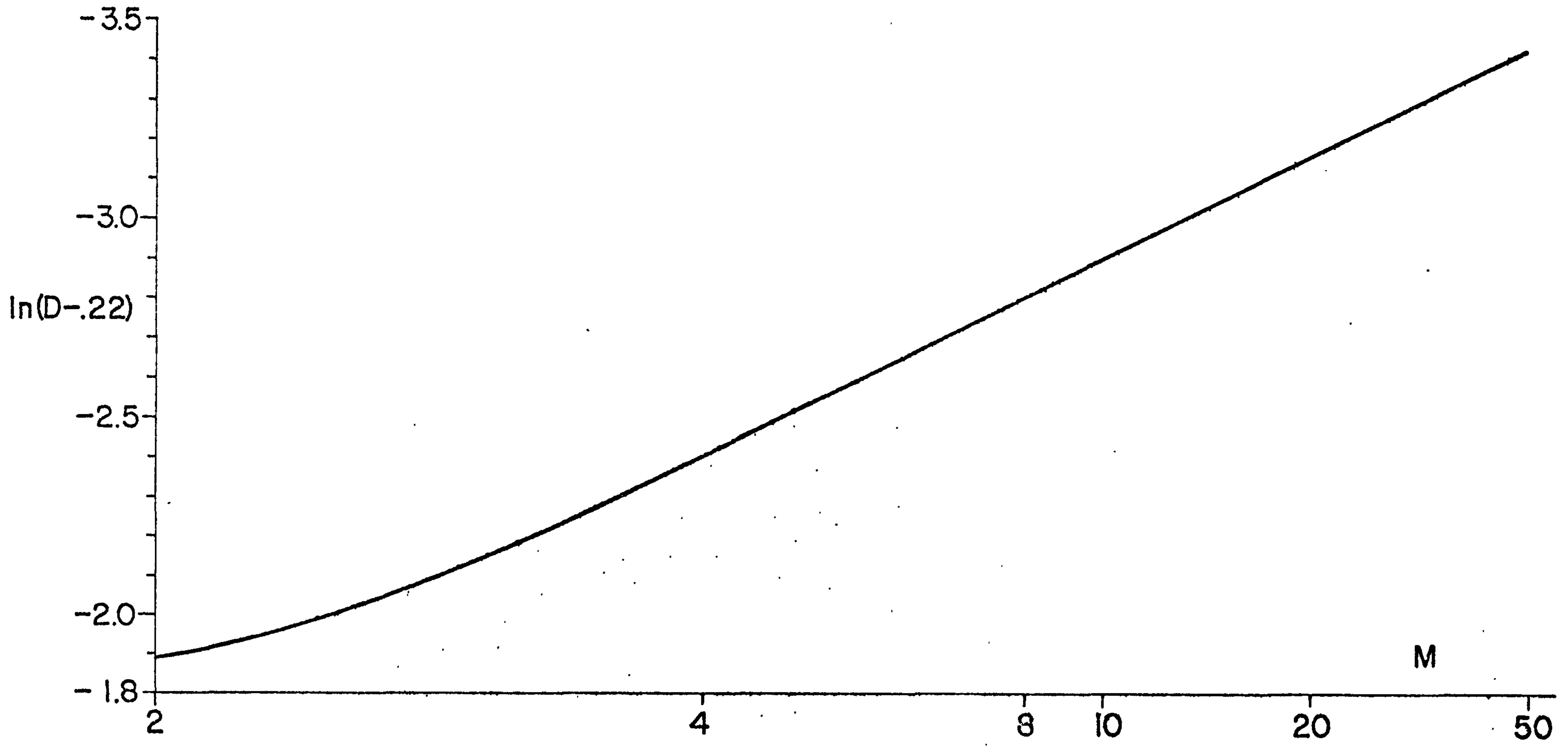


Fig. 4.5 Ln(D-.22) vs. ln(ln(M)): Exponential Metric Tree, Searched by the M-Algorithm

larger values of M verifies the basic form assumed for the asymptotic behaviour of the algorithm; the slope of this line, which is equal to the value of C_2, was measured as .98, very close to the value $C_2 = 1$ found above. Assuming that the value found for $\Delta(1)$, .22, is close to the actual value, the maximum reduction in distortion over that produced by the single-path search which can be obtained through multi-path searching is a factor of .22/.5 , equivalent to 3.6 dB. With the M-algorithm, we achieved about 1.3 dB at M = 2, 2.1 dB at M = 4, 2.5 dB at M = 8, and 3.0 dB at M = 50.

Fig. 4.6 compares the distortion produced by the M-algorithm and the exhaustive search as functions of computational work. It is evident that the M-algorithm is by far the more efficient method for this metric tree.

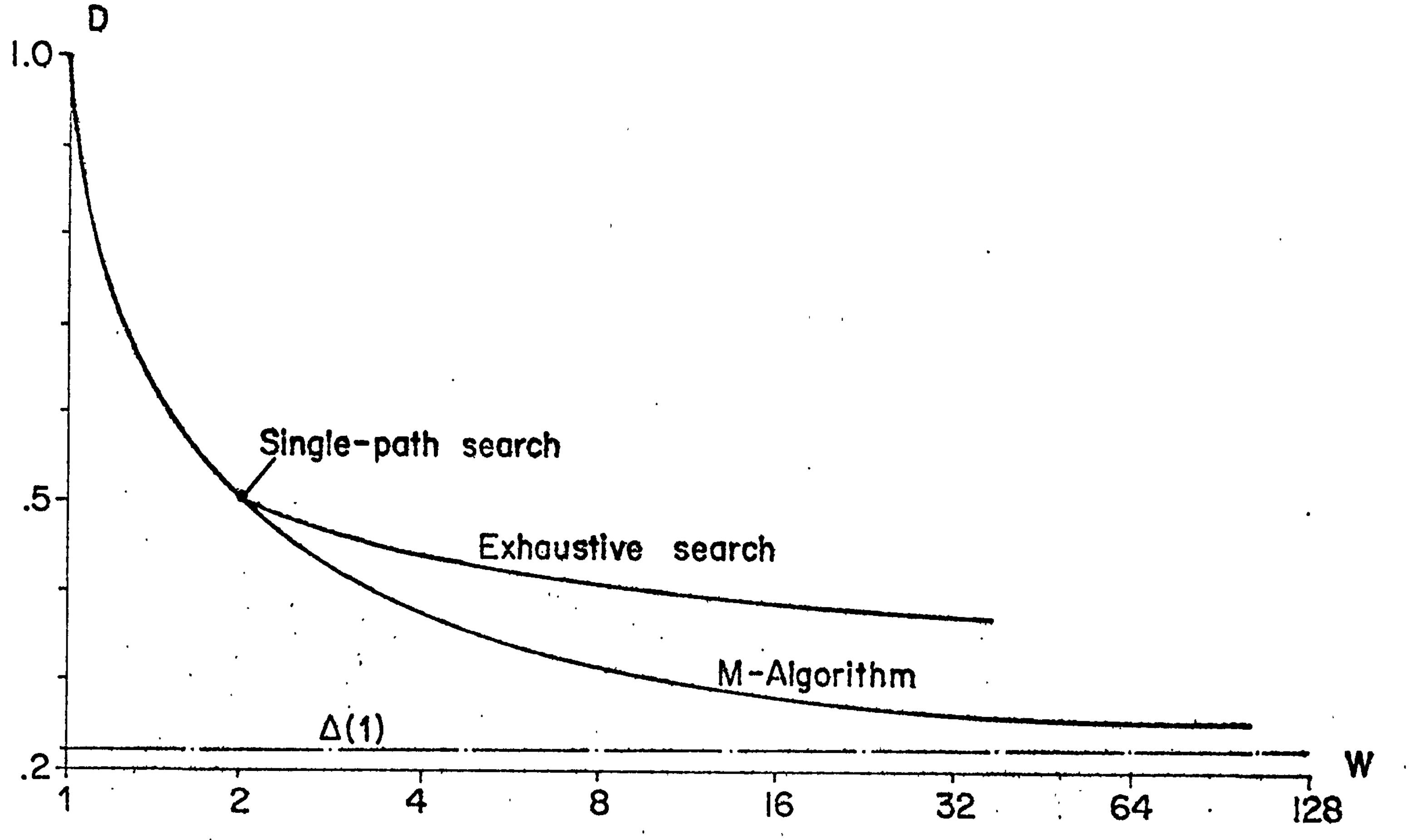


Fig. 4.6 Average Distortion vs. Computation: Exponential Metric Tree, Searched Exhaustively and by the M-Algorithm

CHAPTER V

SEARCH ALGORITHM PERFORMANCE WITH THE UNCORRELATED GAUSSIAN SOURCE

As a first example of the use of tree source coding with an actual data source, we consider the problem of reproducing a sequence of source letters chosen independently from a continuous probability distribution; specifically, the zero-mean, unit-variance normal distribution $\mathcal{N}(0,1)$. We employ the MSE fidelity criterion, and restrict the channel capacity to one bit/letter.

The first choice to be made is that of an appropriate code tree. The conventional choice in this situation would be the tree of Fig. 5.1, in which the two branches extending from each node bear the letters +Q and -Q. An encoder employing this tree is just a two-level quantizer, with quantization levels $\pm Q$. As was mentioned in chapter 3, the single-path algorithm will search this tree optimally; if multi-path searching is not considered, the combination of the quantizer tree, with suitably optimized quantization levels, and the single-path search is the best coding scheme available for this application. It is well-known that the value of Q which optimizes the quantizer for the $\mathcal{N}(0,1)$

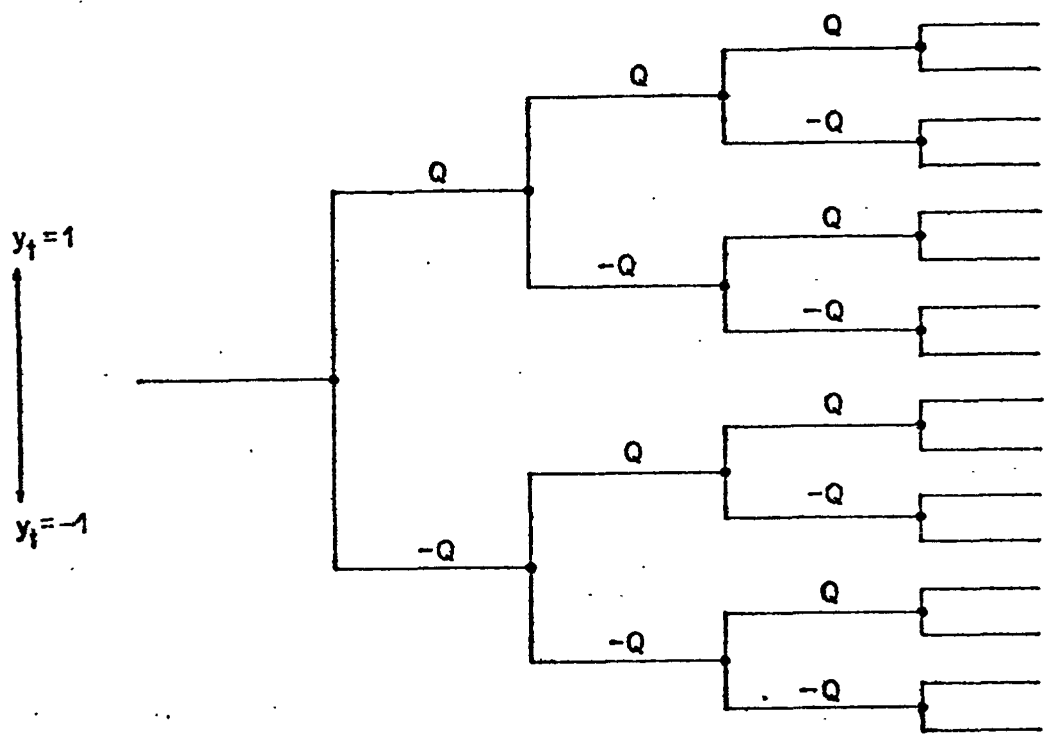


Fig. 5.1 Two-Level Quantizer Code Tree

source distribution is $Q = \sqrt{\frac{2}{\pi}}$, yielding an average distortion of $D = 1-\frac{2}{\pi} = .363$. From (2.1), we find that the performance theoretically attainable is $\Delta(1) = .25$; thus code trees exist which perform (when properly searched) significantly better than the simple quantizer.

One intuitively appealing tree has its branch letters chosen from the same $\mathcal{N}(0,1)$ distribution as the source letters. This tree is in fact very nearly optimal, for Berger shows [1] that a code tree whose branch letters are chosen independently from the $\mathcal{N}(0,.75)$ distribution yields an average distortion approaching $\Delta(1)$ arbitrarily closely for this source as $N\rightarrow\infty$, if exhaustively searched.

Fig. 5.2 plots the results of computer simulation of ensembles of $\mathcal{N}(0,1)$ and $\mathcal{N}(0,.75)$ trees, exhaustively searched, for small values of N; in this region, such trees are in fact much worse than the quantizer. Replacing the exhaustive search with the M-algorithm allows the use of arbitrarily large N, and produces the results of Fig. 5.3 . It is evident that the performance of these probabilistically-generated code trees is worse than that of the quantizer if the single-path search (M=1) is used. With larger M the distortion falls, until at M = 10 they perform about as well as the quantizer. The theoretically-optimal code tree ($\mathcal{N}(0,.75)$) performs somewhat better than the intuitively chosen ($\mathcal{N}(0,1)$) tree for small M, but by M = 10 the two perform about equally well. Still larger values of M would

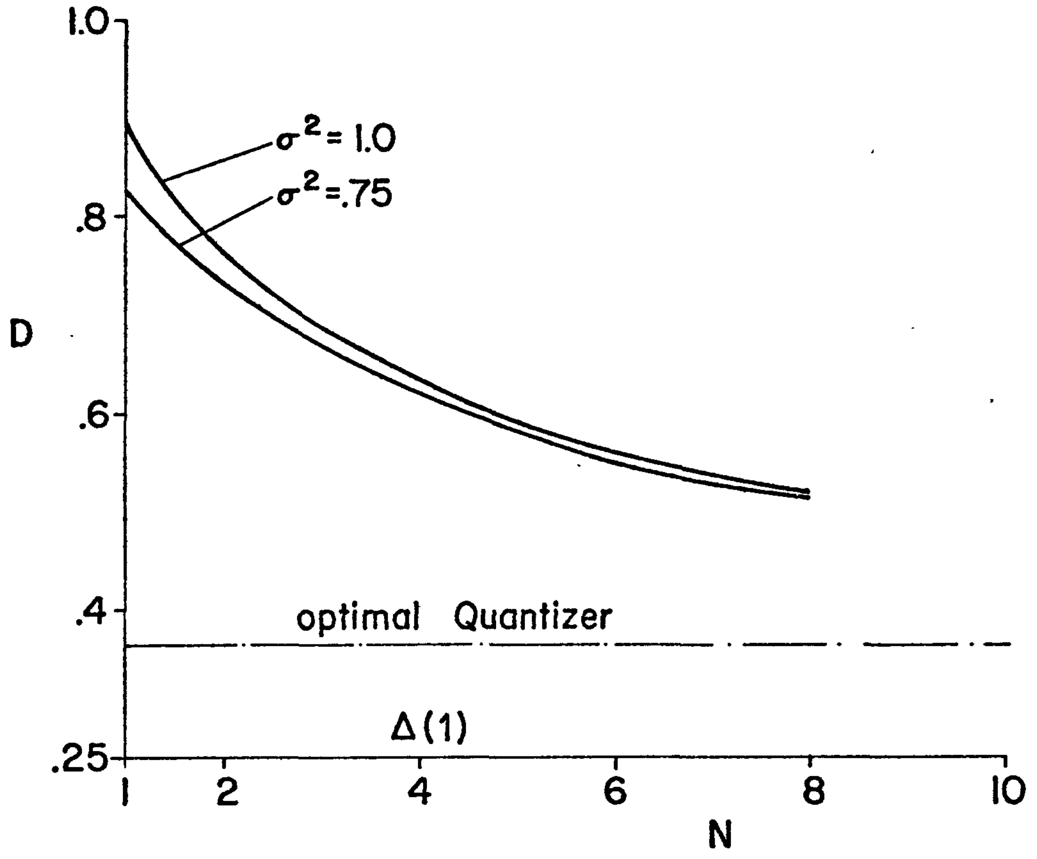


Fig. 5.2 Average Distortion vs. Blocklength: $\mathcal{N}(0,1)$ Source, $\mathcal{N}(0,\sigma^2)$ Code Tree, Exhaustive Search

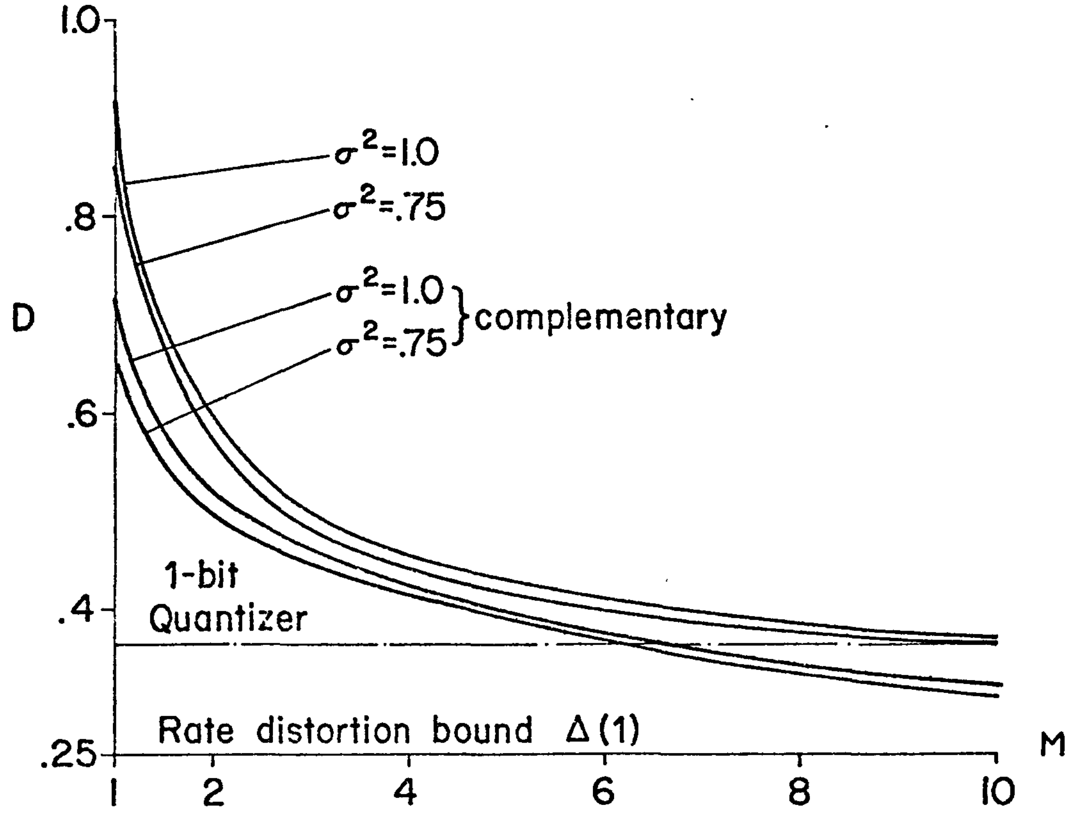


Fig. 5.3 Average Distortion vs. M: $\mathcal{N}(0,1)$ Source, $\mathcal{N}(0,\sigma^2)$ Code Tree

yield lower distortion, but computer time limitations made it impractical to carry the simulation further.

It has usually been found in other tree coding applications [7] that code trees in which the branch letters populating the branches extending from a given node are in some sense complementary to one another tend to perform better than non-complementary trees. We can form complementary trees here by choosing the letter on one of the two branches extending from each node from the desired distribution, and setting the opposing branch letter to the negative of the first. A simulation of trees generated in this way from the distributions used above produced the results labelled "Complementary" in Fig. 5.3. These trees perform significantly better than the original trees, with $\mathcal{N}(0,.75)$ again slightly superior to $\mathcal{N}(0,1)$, reaching the quantizer distortion level near $M = 6$ and an MSE of about .31 at $M = 10$. This lowest value of distortion is about midway between the quantizer distortion level and the rate-distortion limit.

The average distortion to be expected at $M = 1$ with a complementary code tree can be calculated easily. (A similar calculation is possible for the non-complementary trees, but is much more difficult to perform.) At each node, the algorithm chooses from the two available branch letters the one which is closest in value to the source letter. Because of the symmetry of the two branch letters about zero, the one which agrees in sign with the source letter will be chosen. The average distortion is then just the mean-square value of the difference between two r.v.'s whose density functions are those of the absolute values of the source and branch letters. For the $\mathcal{N}(0,1)$ source letter and $\mathcal{N}(0,\sigma^2)$ branch letter distributions, this calculation gives

$$D = 1 + \sigma^2 - \frac{4}{\pi}\sigma \tag{5.1}$$

This relation yields:

$$D = .73,\ \sigma^2 = 1$$
$$D = .65,\ \sigma^2 = .75$$

The experimental values in Fig. 5.3 agree well with this calculation. The value of σ^2 which minimizes (5.1) is $\sigma^2 = \frac{4}{\pi^2}$, for a value of $D = .595$. Using this value of σ^2 for the variance of the branch letters produces a code tree which performs better than the $\mathcal{N}(0,1)$ and $\mathcal{N}(0,.75)$ trees at $M = 1$, but which becomes worse than either of these for $M > 4$.

As in chapter 4, we conjecture that $D - \Delta(1) \propto (\ln M)^{-C_2}$. Fig. 5.4 suggests that such a relationship exists here, with $C_2 = 1$ for the original trees and $C_2 = 2$ for the complementary trees. However, the lack of experimental points for values of M above 10 reduces the reliability of these estimates of C_2.

From these results, it appears that trees whose branch letters are chosen from a continuous distribution require a large amount of computation before their performance even matches that of the low-computation quantizer tree. To improve on the quantizer, however, there must be a <u>range</u> of branch letter

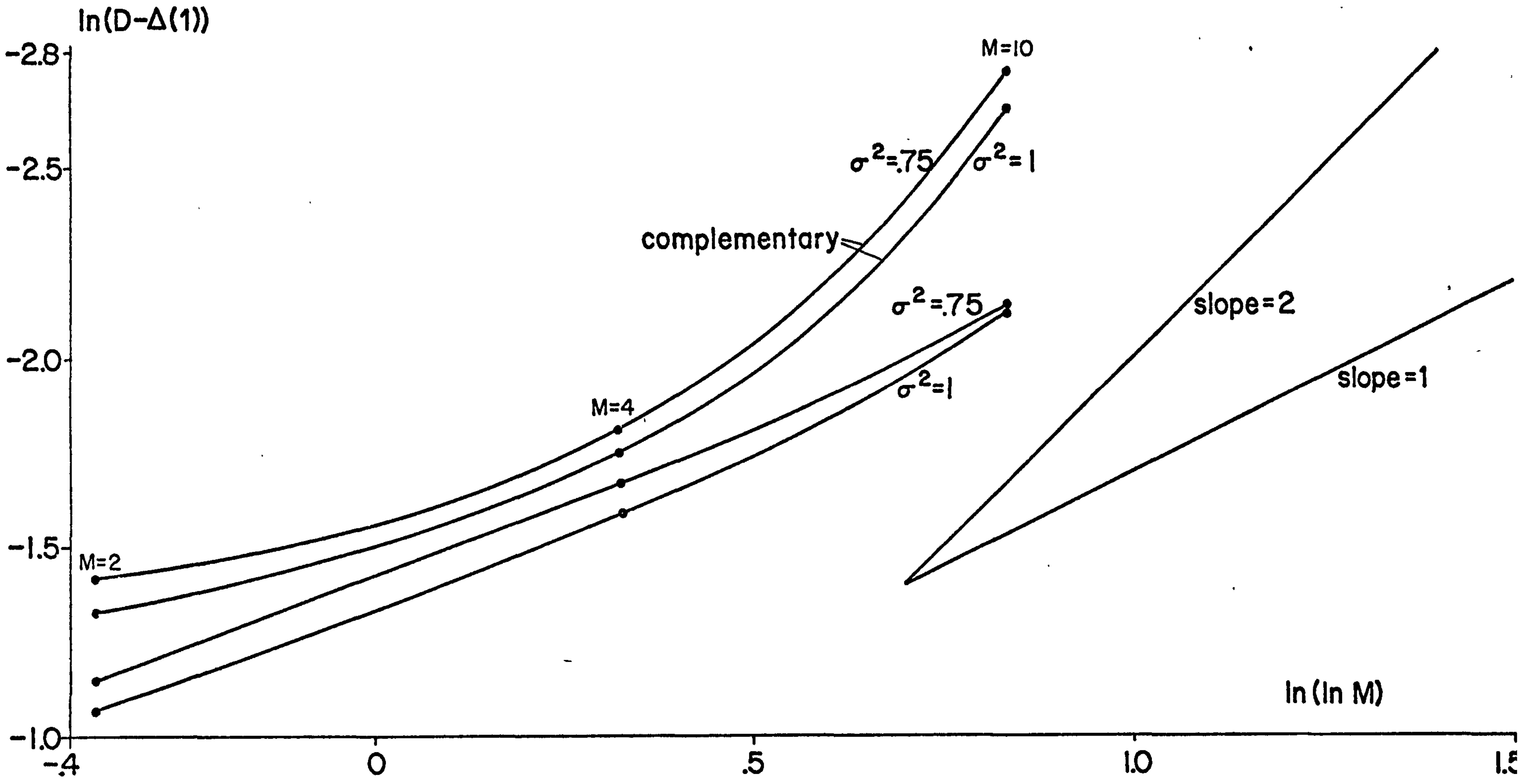


Fig. 5.4 Ln (D-Δ(1)) vs. ln(ln M)): $\mathcal{N}(0,1)$ Source, $\mathcal{N}(0,\sigma^2)$ Code Tree

values available. In the hope of finding a tree which will outperform the quantizer at a moderate level of computation, we investigate trees whose branch letters are chosen from a discrete alphabet. For example, we can consider an extension of the quantizer tree in which the quantization levels are chosen at each node as either $\pm A$ or $\pm B$. For simplicity, we initially assume that the choice is made independently at each node, and that the the two possible choices are equally likely. With the single-path search, we will then effectively be quantizing the source to $\pm A$ half the time, and to $\pm B$ the other half. The resulting distortion will be the average of the distortions produced by $\pm A$ and $\pm B$ quantizers, or

$$D = D_Q + \frac{(Q-A)^2 + (Q-B)^2}{2} \quad ,$$

where Q is the optimal quantization level (very nearly .8 in this case) and D_Q the distortion of the optimal quantizer (.363).

We have experimentally evaluated the distortion produced by this type of tree when searched by the M-algorithm, with M ranging from 1 to 10. At $M = 1$, $A = B = Q$ performs best,as expected (see Fig. 5.5). $M = 2$ yields significant improvement in the performance obtained with widely-spaced quantization levels; it allows $(A,B) = (.6,1.0)$ to perform as well as $(.8,.8)$ (that is, the standard quantizer). It seems likely that a combination of levels not investigated (perhaps $(.75,.85)$) would actually improve upon the optimized quantizer at $M = 2$. At $M=4$,

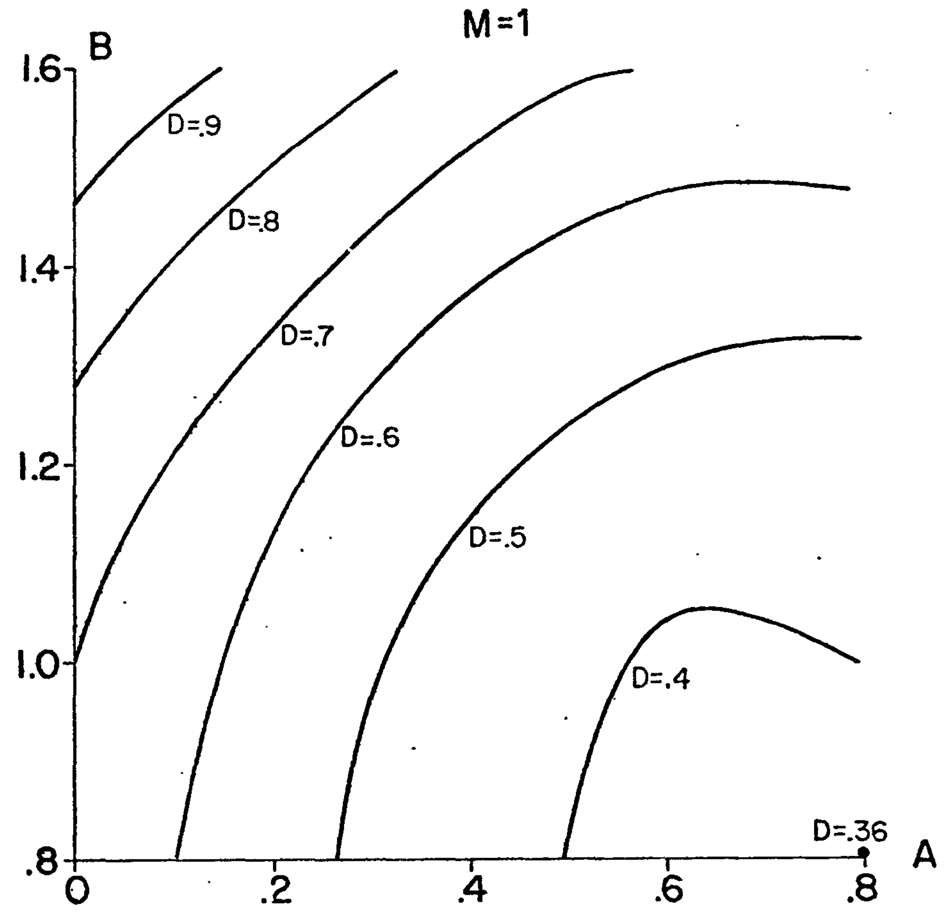


Fig. 5.5 (a)

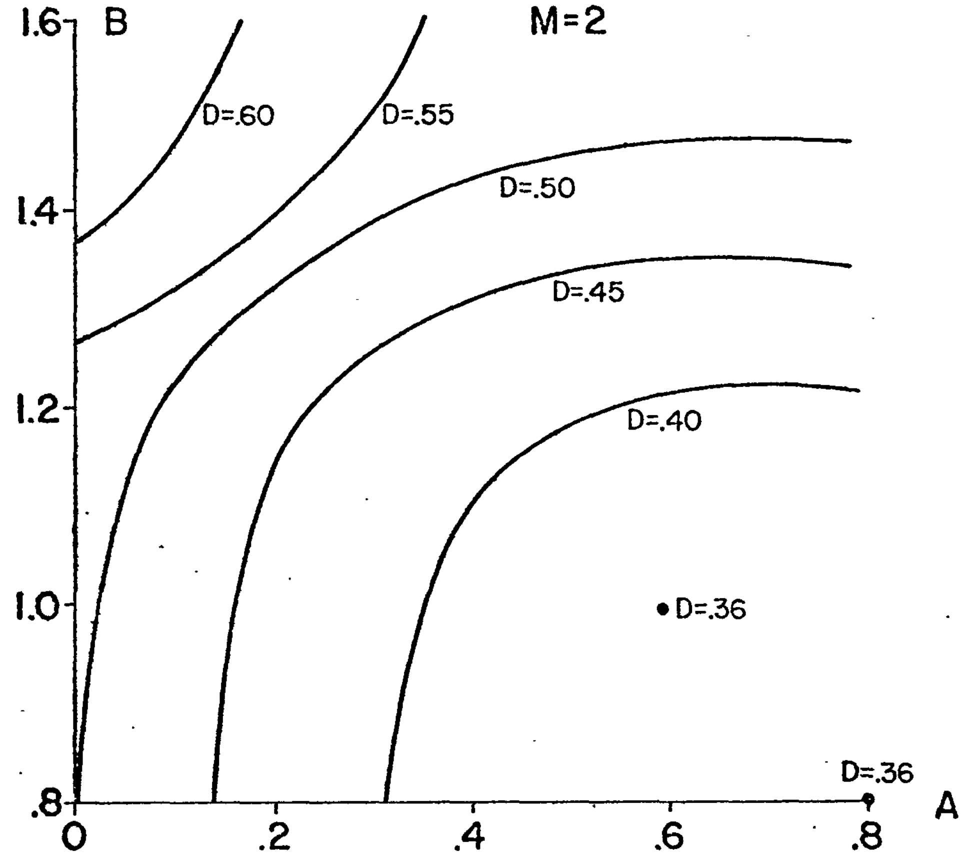


Fig. 5.5 (b)

Average Distortion vs. Quantization Levels:

$\mathcal{N}(0,1)$ Source, Random Four-Level Code Tree

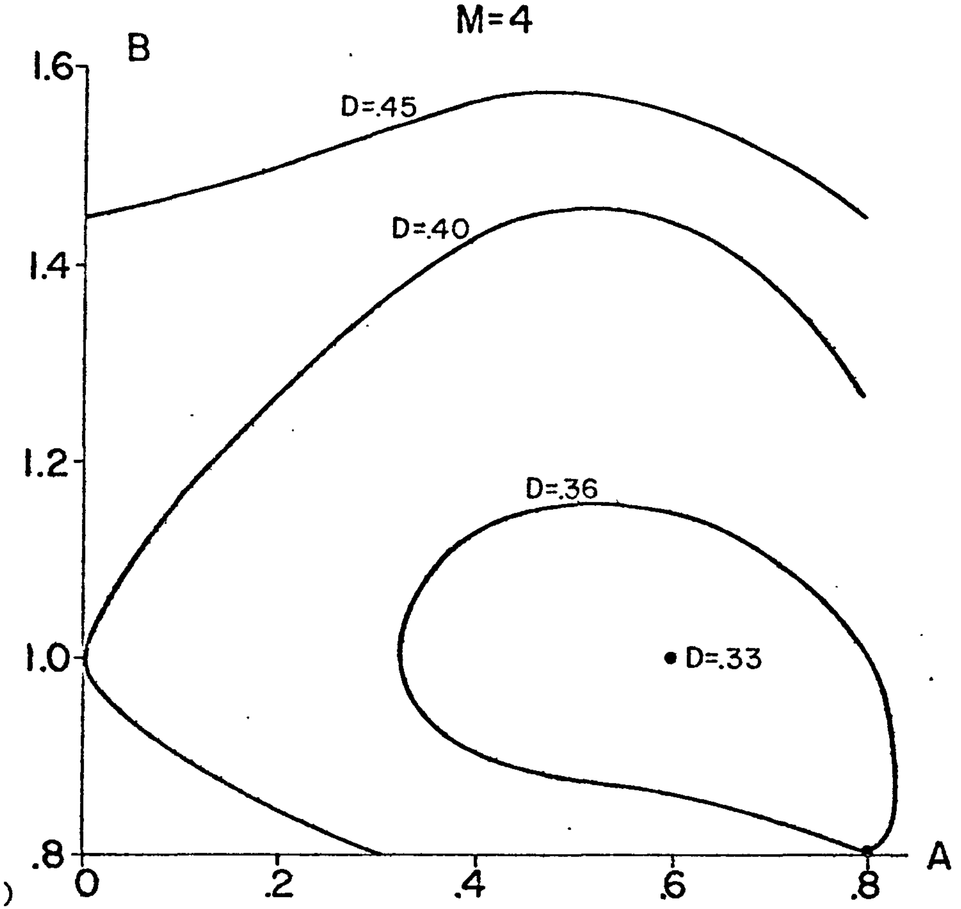


Fig. 5.5 (c)

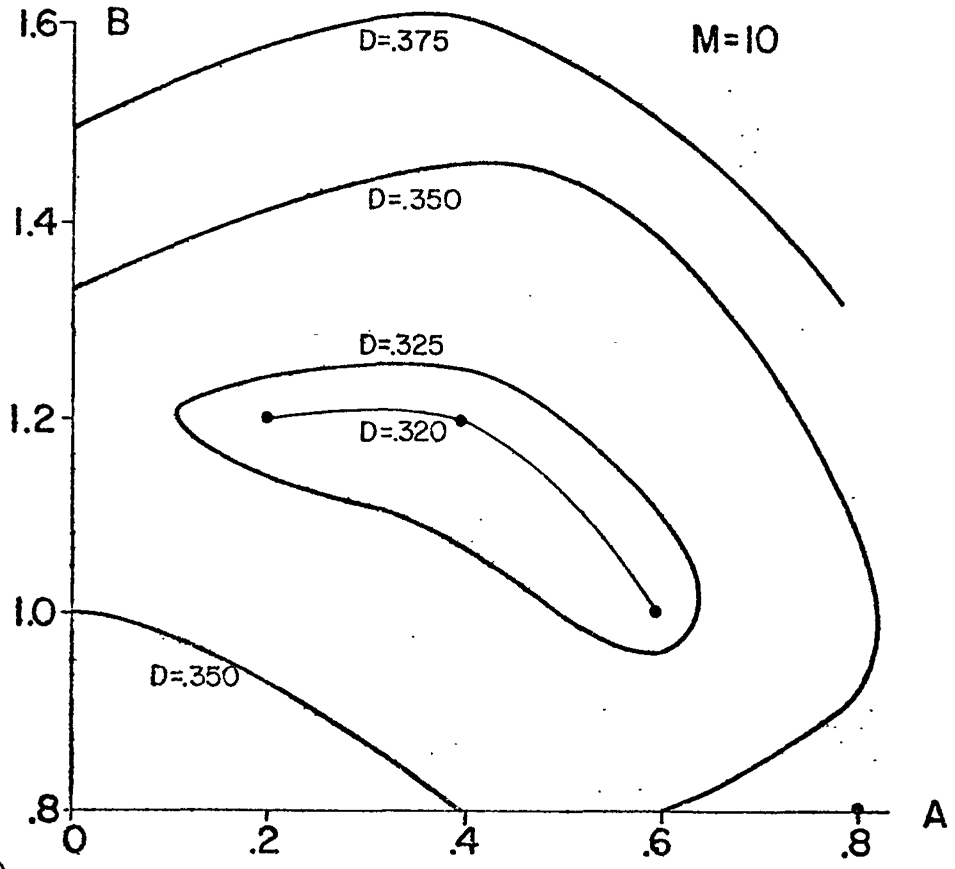


Fig. 5.5 (d)

(A,B) = (.6,1.0) gives D = .33, while at M = 10, (.6,1.0), (.4,1.2), and (.2,1.2) all achieve D = .32; both of these values of distortion are somewhat lower than that of the optimized quantizer.

The effect of these four-level trees and the M-algorithm can also be considered to be an improvement in the dynamic range of the coding system. In many practical situations, the amplitude of the input signal can vary over a considerable range, and while a particular system may be capable of good performance when optimized for an assumed input amplitude, a deviation from the assumed value may cause a significant degradation in the quality of reproduction.

For example, assume that the source we have been considering has a variance in the range (0.5, 2.0) rather than it being fixed at 1. It can be shown (see Appendix C) that choosing the quantization levels of a two-level quantizer to minimize the maximum value of the normalized distortion $\frac{D}{\sigma^2}$ over this range of variance limits $\frac{D}{\sigma^2}$ to a maximum value of .434 . This is about 20% worse than the performance of quantizers individually optimized for each value of variance. It can be seen from Fig. 5.5, however, that at M = 10 the quantization levels of the four-level tree can vary from (.2,.8) to (.4,1.6) (equivalent to a 4:1 change in variance) without the distortion exceeding .37 . This value is only 3% above the optimal two-level quantizer distortion level for a fixed value of variance.

While the trees whose branch letters are chosen from the discrete, four-letter alphabet perform much better at small M than those whose branch letter distribution is continuous, it is still necessary to employ a value of M of at least 10 in order to close the gap between the rate-distortion performance limit and the performance actually attained to half the value reached by the two-level quantizer.

Convolutionally-Generated Trees

The trees so far considered are non-instrumentable; since we have not introduced any deterministic structure into the choice of branch letters, they cannot be calculated from path map digits, and it would be necessary in any implementation to store the entire code tree. One method of generating branch letters from path maps which has been found to be very valuable in the digital-data case [2] is the convolutional approach. Here the path map digits are inserted sequentially into a shift register of length ν (ν is called the constraint length of the code). Each of the β branch letters associated with the branch defined by the path map digits is generated as a weighted sum of the digits in the shift register. The circuit of Fig. 5.6, which performs the convolution described, is used by the encoder to generate branch letters as required, and forms the entire source decoder. In

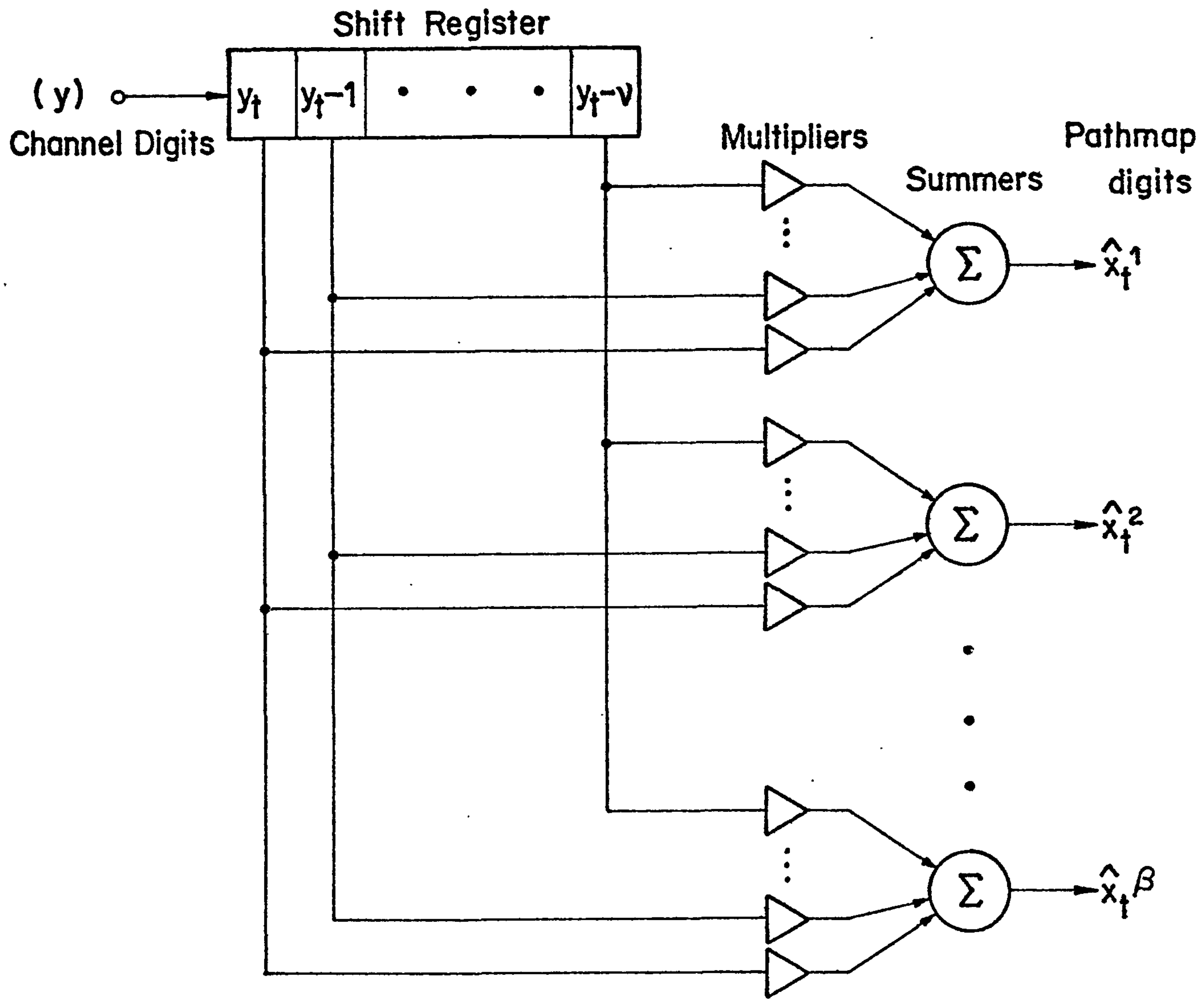


Fig. 5.6 Convolutional Tree Code Generator

the digital case, the multiplications and additions are done modulo the reproducing alphabet size; where the reproducing alphabet is the real line, the circuit becomes a finite-impulse-response (or transversal) digital filter. In an efficient coding system, the channel digits must be essentially uncorrelated; otherwise, there would be redundancy in the channel digit sequence which could be removed by further coding. If the original source is correlated, the digital filter source decoder can be used to generate output sequences having approximately the same correlation function as the source. In the case of an uncorrelated source,however, the correlations introduced by such a filter are undesirable.

The convolutional decoder can be modified to suit uncorrelated continuous-alphabet sources by relacing the linear relation between path map digits and branch letters with an arbitrary non-linear relation. Consider the generation of a binary, four-level tree like those discussed above. We insert ν path map digits (in this case, bits) into a shift register, associated with which are two mod-2 summers fed from different combinations of shift register stages. The four unique combinations of summer outputs ((0,0), (0,1), (1,0), (1,1)) are paired with the four branch letter values (+A, -A, +B, -B) . This pairing can be implemented as a table look-up procedure in which the summer outputs form an address to a table containing the members of the branch letter alphabet. The connections between the summers and the shift register can be described in terms of masks, ν-bit words in which a "1" indicates that the corresponding path map digit participates in the summation with which the mask is associated. With large ν, the code tree approaches the purely probabilistic tree, in which it is equally probable for any branch letter to follow any others. Trees generated in the way described above permit a range of branch letter values, but do not impose a specific correlation function on the output sequences.

We have applied the M-algorithm to convolutionally-generated code trees with values of ν ranging from 1 to 16. Fig. 5.7 is a contour plot representative of the performance of a good convolutional tree at M = 2. Here the masks used are 100...0 and 1110 1101 0100 1000. Since two masks are used, the reproducing alphabet again has four members. The point (.8,.8) corresponds as before to the optimal two-level quantizer and achieves D=.363. The optimal set of levels has become (.5,1.1), with D=.345 . It appears that, as the number of paths searched increases, the optimal quantization levels spread out symmetrically from the value (.8) which is optimal for the single-path search. The performance of trees generated by different masks of a given constraint length varies widely; however, the best code found for $\nu = 5$ reaches D = .34 at M = 4 (performance only slightly poorer than the best probabilistic tree), and the best found for $\nu \leq 16$ gives D = .30 at M = 4 . Not only are the convolutional trees instrumentable,

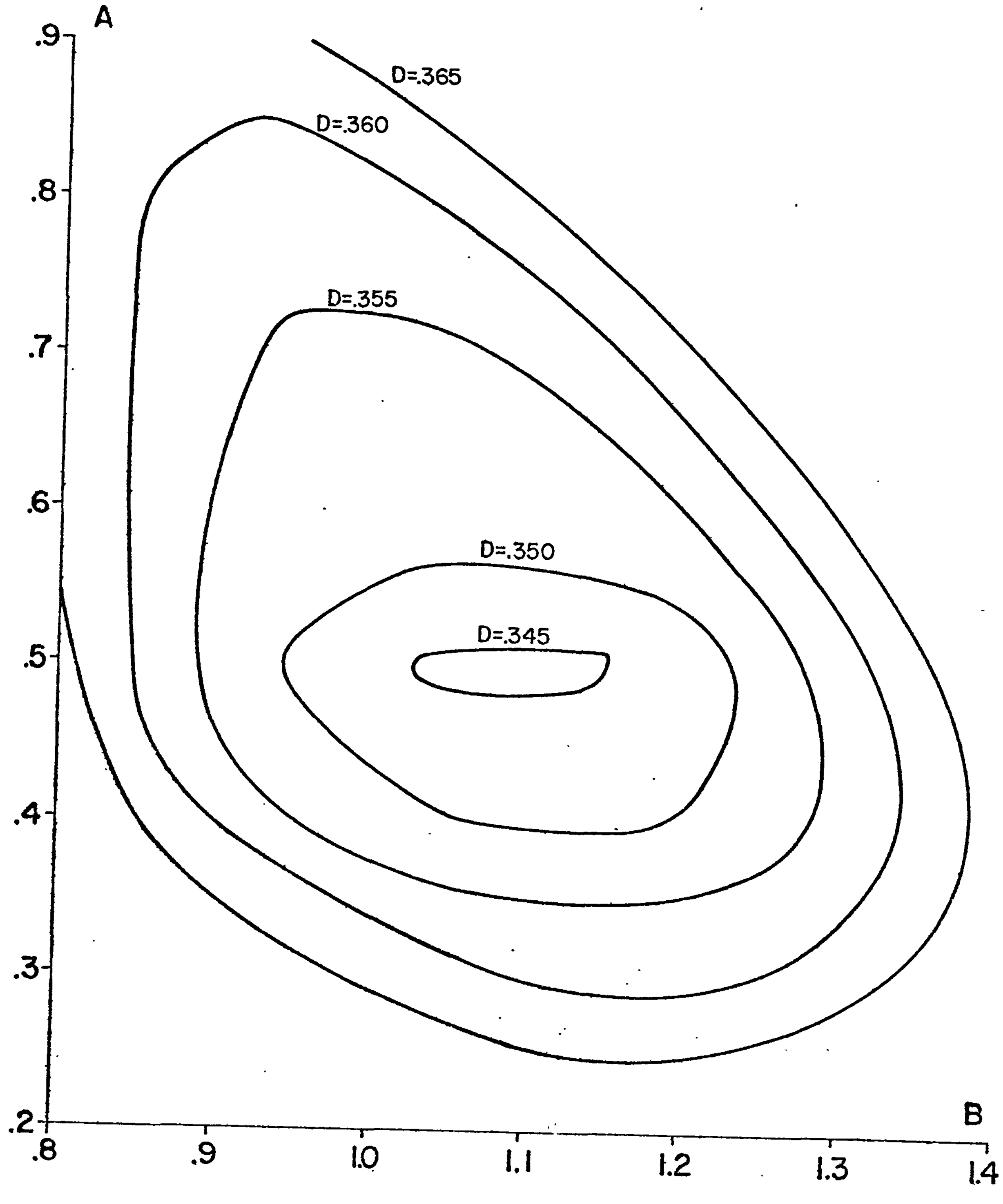


Fig. 5.7 Average Distortion vs. Quantization levels: $\mathcal{N}(0,1)$ Source, Convolutional Four-Level Code Tree

then, but the best members of the group perform significantly better than the average randomly chosen tree. These results are analogous to those reported by Anderson for the digital-data case[2].

With the convolutional trees, we have found an instrumentable coding scheme which performs about midway between the optimal quantizer and the rate-distortion limit and which requires only a quite moderate amount of computation (eight branches per letter examined for the binary tree at $M = 4$).

CHAPTER VI

SOURCE CODES FOR CORRELATED CONTINUOUS-ALPHABET SOURCES

As was mentioned in chapter 5, convolutional source decoders are well-suited to the reproduction of correlated sources. Fig.6.1 shows such a decoder for a continuous-alphabet source. Incoming channel digits $\{y\}$ are first mapped into corresponding quantization levels $\{q\}$ and then applied to a transversal digital reconstruction filter which generates the output sequence $\{\hat{x}\}$ according to the rule

$$\hat{x}_t = \sum_{i=0}^{N_t} c_i q_{t-i} \tag{6.1}$$

The z-transfer function of this filter is

$$C(z) = \sum_{i=0}^{N_t} c_i z^{-i}$$

We can derive the structure of an encoder which employs the single-path algorithm to search the code tree produced by this decoder. With the single-path search, the encoder will have released the channel digits up to y_{t-1} before having accepted x_t from the source. This implies that $\hat{x}_t = c_0 q_t + \tilde{x}_t$, where $\tilde{x}_t = \sum_{i=1}^{N_t} c_i q_{t-i}$ is independent of the choice of q_t.

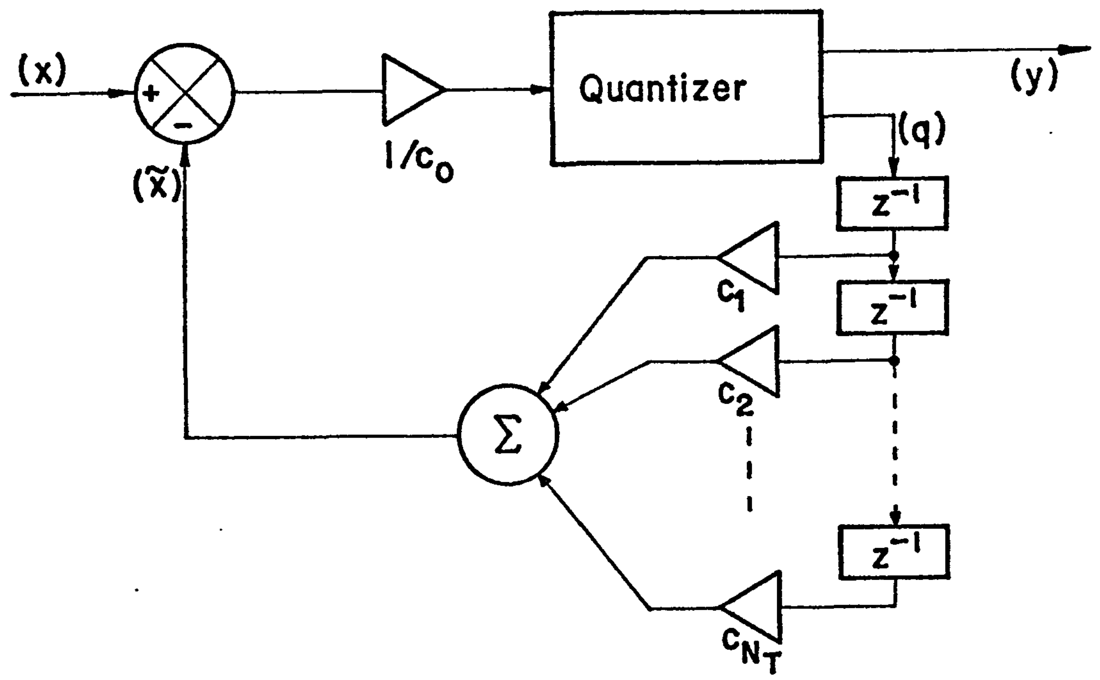


Fig. 6.2 Source Encoder for Transversal Source Decoder

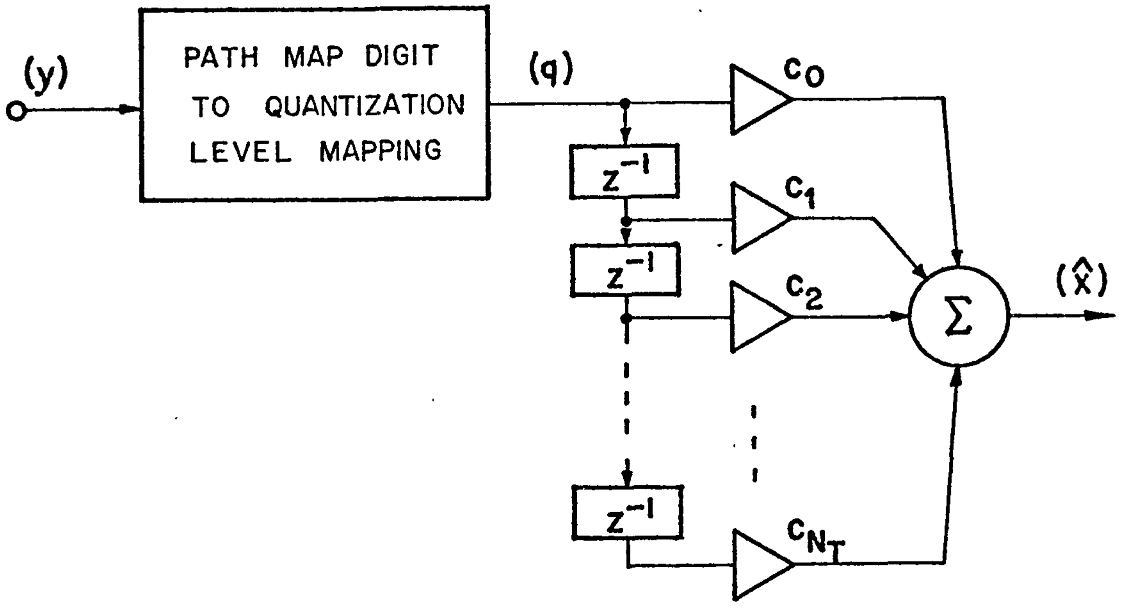


Fig. 6.1 Continuous-Alphabet Source Decoder

To minimize the magnitude of the reproduction error $e_t = x_t - \hat{x}_t$, the encoder must choose q_t to minimize $|x_t - \hat{x}_t| = |x_t - \tilde{x}_t - c_0 q_t|$: a quantizer with input $\frac{x_t - \tilde{x}_t}{c_0}$ and the possible values of q_t as quantization levels does just that. In addition to the quantizer, the encoder must contain a circuit which will generate $\tilde{x}_t$; this is just a transversal filter

$$\tilde{C}(z) = \sum_{i=1}^{N_T} c_i z^{-i} \tag{6.3}$$

The resulting encoder is shown in Fig. 6.2 .

In practice, it is often convenient to employ a recursive reconstruction filter, which generates $\{\hat{x}\}$ according to

$$\hat{x}_t = q_t + \sum_{i=1}^{N_R} a_i \hat{x}_{t-i} \tag{6.4}$$

If we define

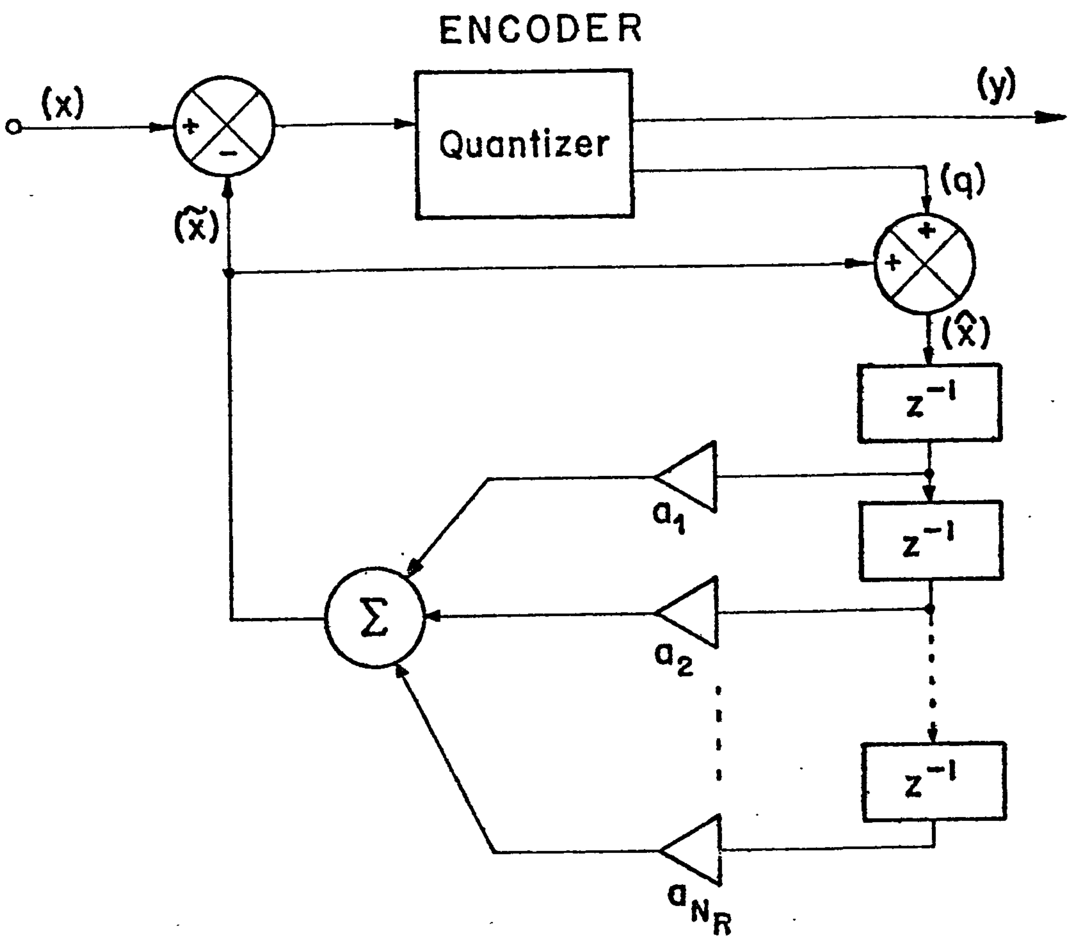
$$A(z) = \sum_{i=1}^{N_R} a_i z^{-i} \tag{6.5}$$

then the transfer function of the recursive filter is $\frac{1}{1-A(z)}$.

Such a filter is equivalent to a transversal filter of the form of (6.2), where c_i has the value of the i^{th} sample of the impulse response of the recursive filter. In general, this transversal equivalent is of infinite length. If the recursive filter is stable, however, the c_i eventually approach zero as $i \to \infty$. In a practical system, the transversal filter can be truncated at the point where the further c_i are below the precision level of the arithmetic used.

Proceeding as for the transversal filter, we define $\tilde{x}_t = \sum_{i=1}^{N_R} a_i\hat{x}_{t-i}$, and take q_t to be the quantized value of $(x_t-\hat{x}_t)$. Fig. 6.3 shows an encoder and decoder employing the recursive reconstruction filter; conventional DPCM systems employ this structure [8].

The use of the recursive filter permits a theoretical optimization of the predictor $A(z)$. The mean-square value of the reproduction error is $\sigma_e^2 = E(e_t^2)$, which is also the mean-square error in the quantization of $p_t = x_t-\hat{x}_t$. For a given form of the probability density of p_t, the minimum value of σ_e^2 attainable through proper choice of the quantization levels is given by $\sigma_e^2 = K_q\sigma_p^2$, where K_q is a parameter determined by the density function of p_t and by the number of quantization levels employed. Max[9] gives a method of finding the optimal choice of levels for a given probability density, and derives the value of K_q for the normal distribution for various numbers of levels. To minimize σ_e^2, we must minimize σ_p^2. This implies that x_t should be as close as possible to x_t: for this reason, $\tilde{x}_t$ is considered to be a prediction of x_t, and p_t is known as the prediction error. If we assume good reproduction, (i.e., $\hat{x}_t \approx x_t$) we can approximate $\tilde{x}_t$ by $\sum_{i=1}^{N_R} a_i x_{t-i}$. (O'Neal states[8] that this assumption is valid for quantizers with at least 8 levels.) Under this assumption, we can calculate the a_i which minimize σ_p^2 for a given N_R by means of LMS (least mean square) prediction theory [7], if the



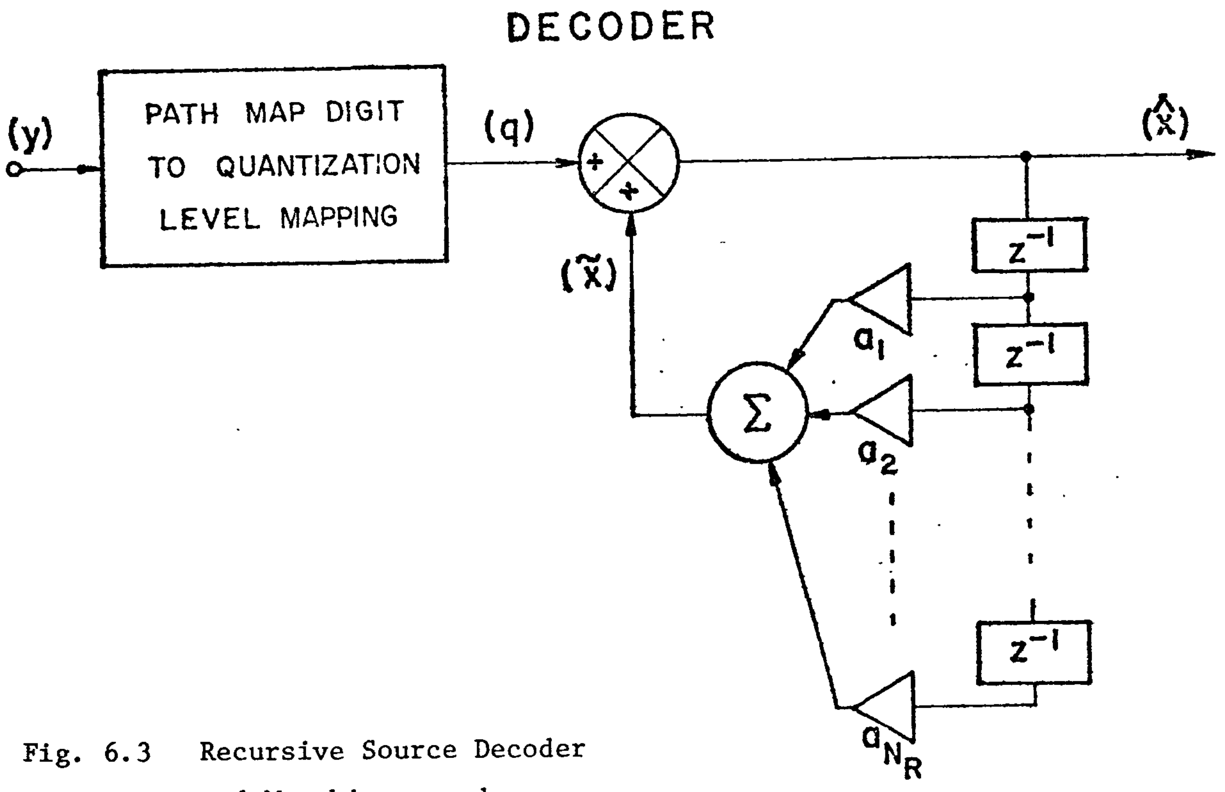


Fig. 6.3 Recursive Source Decoder and Matching Encoder

autocorrelation function $R_{xx}(kT_s)$ is known for $k = 0,1,2,\ldots,N_R$.

Assuming that $\sigma_x^2 = 1$, the signal-to-noise ratio, or SNR, of the DPCM system is $\frac{1}{K_q\sigma_p^2}$. If we let $a_i = 0$ for all i, the system reduces to PCM, with an SNR of $\frac{1}{K_q}$. (Since $p_t = x_t$, $\sigma_p^2 = 1$.) DPCM thus performs at an SNR a factor of $\frac{1}{\sigma_p^2}$ greater than PCM. This factor, usually expressed in dB, is called the signal-to-noise improvement or SNI of the DPCM system. It has been found [10] that for speech sampled at 8KHz, an SNI of about 10 dB can be achieved with a three-tap predictor ($N_R = 3$), and that more taps are of little additional benefit.

The code tree produced by LMS prediction is optimal for the single-path search in the limit of perfect quantization (i.e., an infinite number of levels). With coarse quantization, or a multi-path search, the LMS code need not be (and indeed is not) optimal. The following two approaches suggest modifications of the LMS code for use in such situations:

Rate-Distortion Optimal Codes

Berger shows [1] that the optimal code for use with a certain type of correlated continuous-alphabet source is generated by filtering i.i.d. Gaussian variates with a recursive filter identical to that used in conventional DPCM, in cascade with a low-pass transversal filter. Substituting the chosen sequence of quantization levels for the Gaussian variates produces an instrumentable, DPCM-like coding scheme.

Smoothed LMS Codes

Subjectively, one of the major effects of coarse quantization is to introduce a prominent tone at half the sampling frequency into the reconstruction. In addition to this tone, other noise falling mainly in the high-frequency range of the reconstruction bandwidth is also introduced. To remove the tone, and to reduce the high-frequency noise, it has been suggested that a low-pass transversal filter $B(z) = \sum_{i=0}^{N_S} b_i z^{-i}$, called a smoothing filter, be placed in cascade with the conventioanl reconstruction filter.

Anderson [7] discusses in detail these three code classes (LMS, Rate-Distortion Optimal, Smoothed LMS). Fig. 6.4 shows a typical path map and the corresponding output sequences generated by:

A) A typical recursive reconstruction filter

B) The filter of A) followed by the smoother $B(z) = \frac{1+z^{-1}}{2}$.

Because the DPCM and DPCM-like code trees do not repeat the same set of branch letters at each node, the single-path search is not optimal; we can expect an improvement in SNR if a multi-path search algorithm is applied to such a tree.

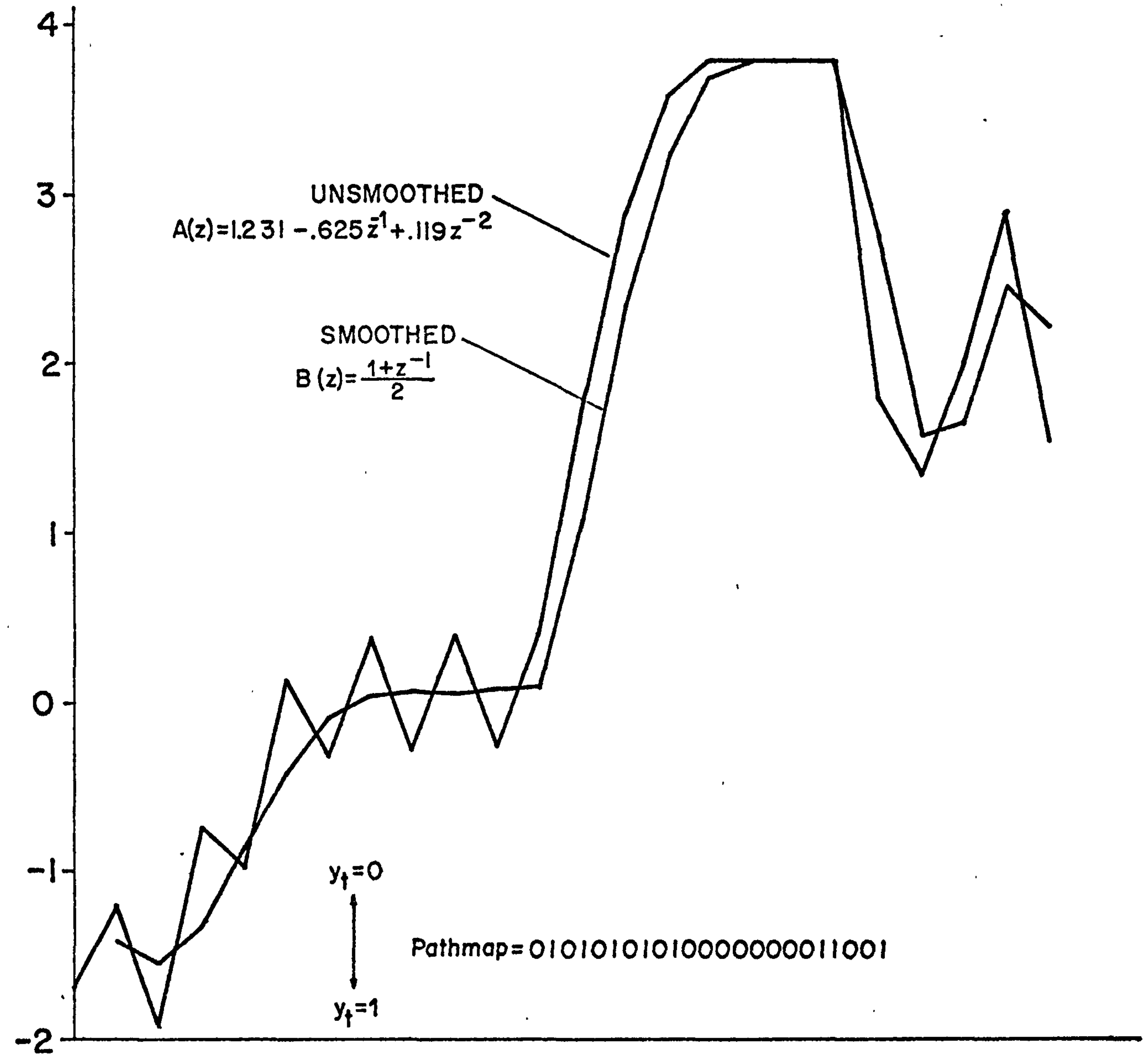


Fig. 6.4 Typical Output Sequences of Smoothed and Unsmoothed Source Decoders

The fact that a multi-path search algorithm is used has no effect on the structure of the decoder; the encoder, however, becomes more complex. In conventional DPCM, the quantizer embodies the logic of the single-path search, while multi-path DPCM requires an encoder structure like that shown in Fig. 6.5 . The "Search Algorithm Unit" selects a number of path maps from those available to the encoder and applies each in turn to the "Decoder Model", which generates the same reproducing letters that would be produced if the corresponding path maps were to be transmitted to the actual decoder. The present source letter and these reproducing letters are presented to the "Metric Calculation Unit", whose output is a measure of the distortion involved in the various proposed reconstructions of the input. An examination of this metric information allows the "Search Algorithm Unit" to decide on the best path map digit to transmit to the decoder.

The discussion in this chapter applies to systems concerned with the transmission of any correlated continuous-alphabet source, and using any search algorithm. In the following chapter, we specialize to the case of a coding system employing the M-algorithm in the transmission of sampled speech waveforms.

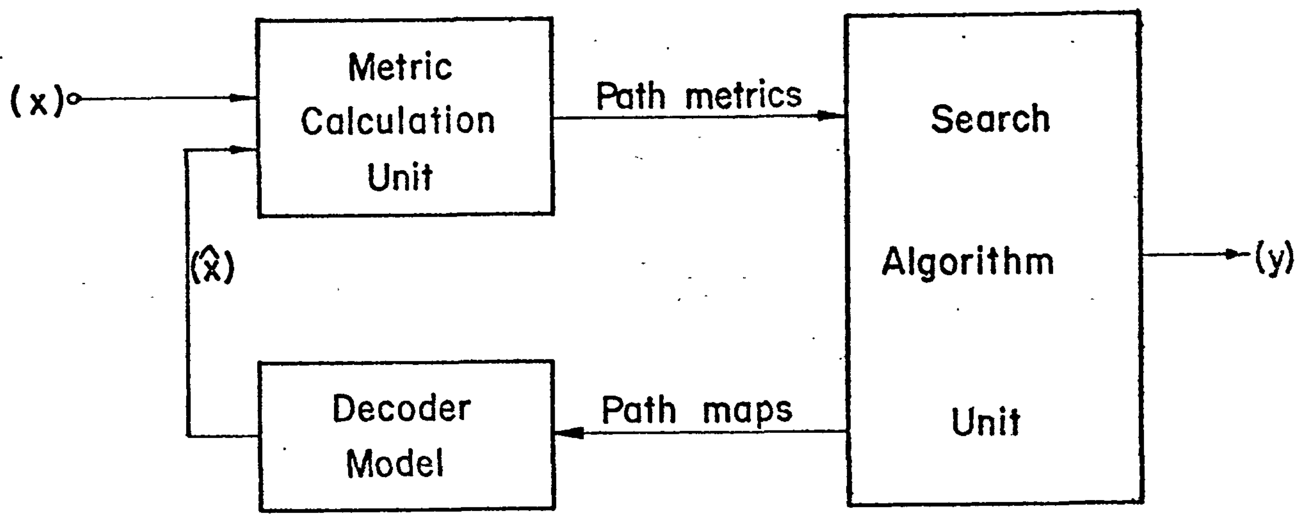


Fig. 6.5 Multi-Path Source Encoder

CHAPTER VII

SIMULATION OF MULTI-PATH SPEECH SOURCE CODING

A system capable of simulating and evaluating single- and multi-path DPCM speech source coders has been implemented on a CDC 1700 computer. The input to the simulation consists of digitized speech samples stored on the computer's magnetic disk. Very fine digitization (14 bits) is employed, allowing the source alphabet to be considered to be continuous. Both encoder and decoder are simulated; the decoder output is stored on disk for later real-time playback. This permits subjective comparisons to be made between the original speech and the coded version, and between reconstructions generated by systems with differring parameters. Objective evaluation is done on the basis of the signal-to-noise ratio of the decoder output, as calculated by the simulation program.

The speech used as input to the simulation was taken from a standard speech processing test tape[11]; to permit meaningful comparisons between different coders, each processed the same spoken sentence, of approximately two seconds duration. A sampling rate of 8Khz was used for the majority of the tests, with the speech bandlimited prior to sampling by Butterworth high- and low-pass filters with cutoff frequencies of 30Hz and 3KHz respectively, and with asymptotic attenuation rates of 24 dB/octave. Code rates of one and two bits/sample were employed, for channel data rates of 8K and 16K bits/second. Further tests were performed with a sampling rate of 5KHz; here, the upper band limit was set by a Butterworth filter with cutoff frequency 2.5 KHz and an asymptotic attenuation rate of 96 dB/octave. Acode rate of 2 bits/sample was used, for a data rate of 10K bits/second.

The reconstruction filters employed were of the smoothed and unsmoothed "LMS" type described in chapter 6. The LMS predictors, with N_R fixed at three, were matched to one of two autocorrelation functions: an average speech autocorrelation reported by McDonald [10] and the autocorrelation corresponding to a two-pole Butterworth low-pass power spectrum with cutoff frequency 1KHz. The Butterworth spectrum was chosen to approximate the long-term average speech autocorrelation while yielding a reconstruction filter of short impulse response. The two autocorrelation functions used are shown in Fig. 7.1 .

The smoothers, with N_S 2 or 3, were generally chosen heuristically; the main considerations in choosing these were the presence of a zero at $f_s/2$ (to null out the tone produced by coarse quantization) and a minimal amount of delay in the passband. A low-pass filter specified by rate-distortion theory, called the R(D) smoother, was also employed. Further details of the choice of reconstruction filters are found in Anderson [7].

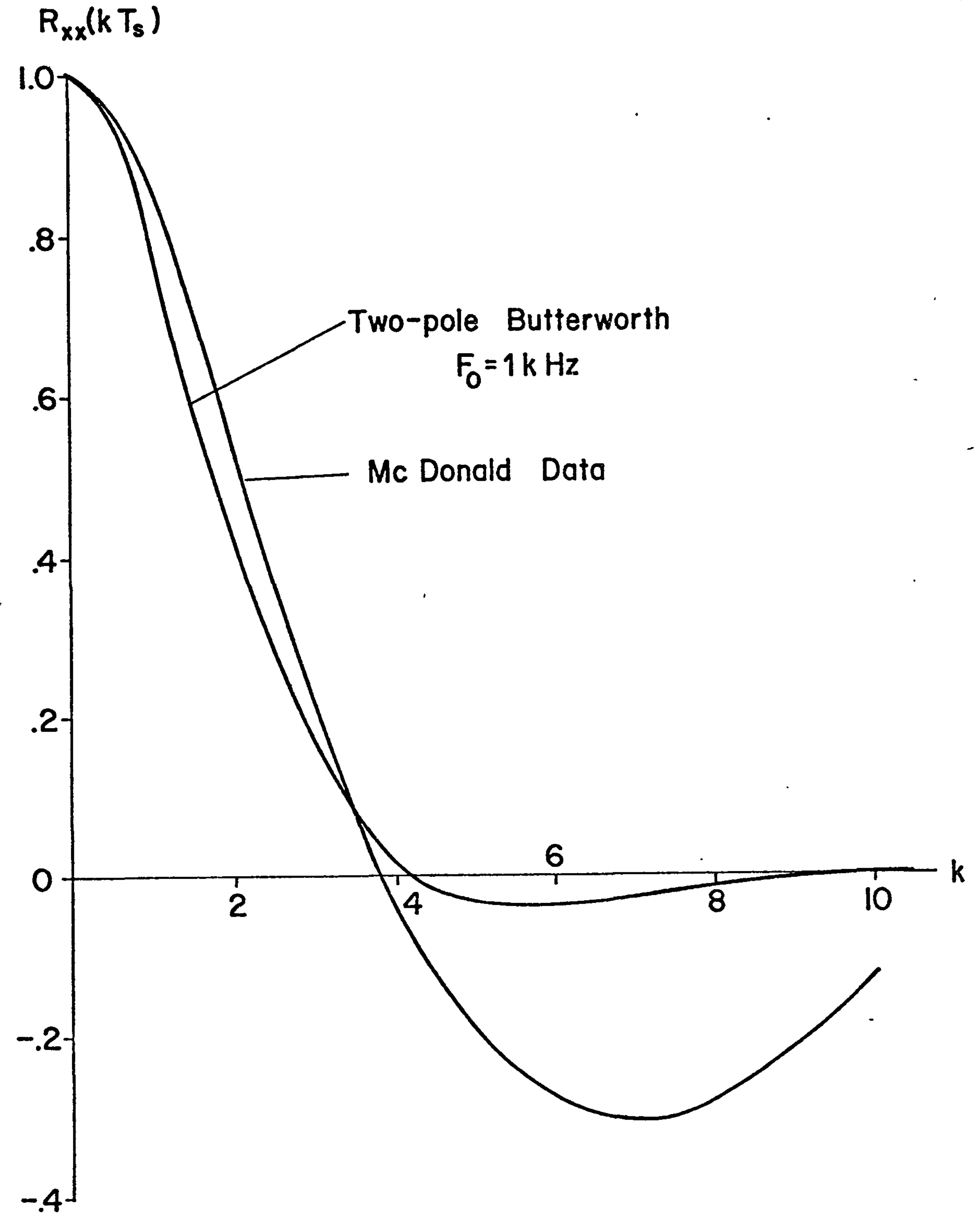


Fig. 7.1 McDonald and Two-Pole Butterworth Autocorrelation Functions

The tree-searching function was performed by the M-algorithm; the fidelity criterion chosen was mean-square error. To avoid excessively long simulation runs, M was limited to values no greater than 10. This is also the region of practical interest for this search algorithm, since other algorithms can be more efficient if a large amount of computation is permitted. It is desirable for l to be large if the M-algorithm is to perform as well as possible; however, to allow the manipulation of path maps as single computer words (16 bits in the CDC 1700) it was necessary to limit $R \cdot l$ to 16. This limitation was not considered likely to have an important effect on the performance of the algorithm.

Very efficient filter operation was achieved by pre-calculating the filter output for each possible path map and storing these in a table. Then, in the course of the simulation, the branch letters corresponding to a particular path map were found by a simple look-up not requiring any arithmetic operations. With the filter output table limited to 1024 words, a maximum of 10 path map bits can affect the branch letter value. This requires that the reconstruction filters have finite-length impulse responses lasting no more than 10 samples at $R = 1$ bit/sample, or 5 samples at $R = 2$. To achieve this, the composite reconstruction filters (i.e., recursive filter-smoother cascades) were expressed in transversal form and truncated to the required length. Fig. 7.2 shows the transversal equivalents (that is,

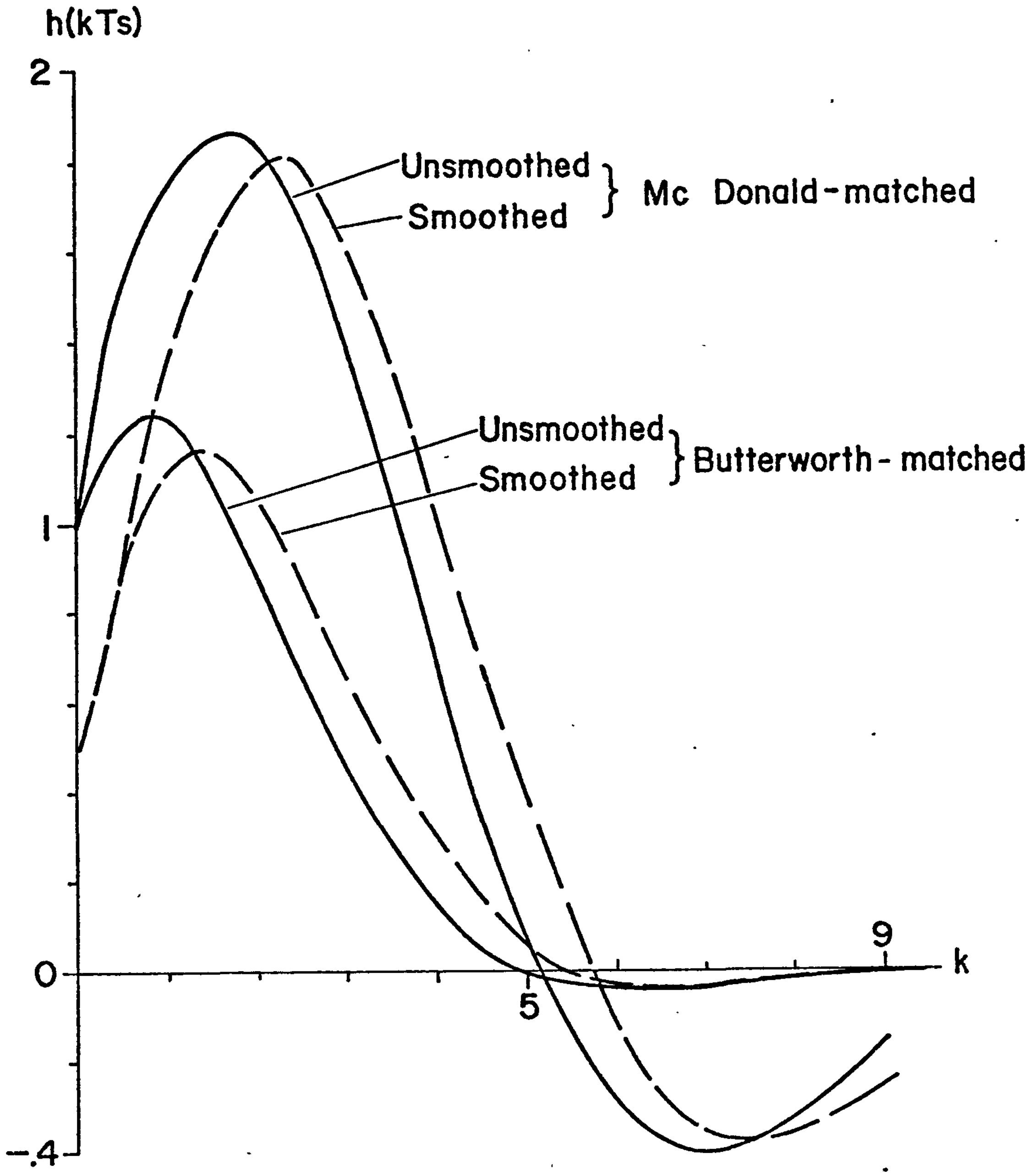


Fig. 7.2 Impulse responses of Smoothed and Unsmoothed Source Decoders

the impulse responses) of several composite filters out to 10 samples at f_s = 8KHz. It is evident that little is sacrificed in truncating the filter matched to the Butterworth autocorrelation,while the truncation of the McDonald-matched filter has a more severe effect on the filter impulse response.

Simulation Results

1) $R = 1$, f_s = 8KHz.

With a code rate of one bit/sample, q_t can assume one of two values. We chose these two possible values to be $\pm s$, where s is the step size of the system. This choice produces a type of complementary code tree, as the two branch letters stemming from a node are symmetrical about the prediction $\tilde{x}_t$. Varying s for best performance is equivalent to adjusting the amplitude of the input signal to best match the encoder. Fig. 7.3 plots SNR as a function of s, at $M = 1$, for several choices of reconstruction filter. (At $M = 1$ the system reduces to single-bit DPCM, or predictive delta modulation.) With the McDonald-matched filter, the maximum value of SNR achieved was about 8 dB, some 1.5 dB better than that attained with the Butterworth-matched filter. The use of a smoother with coefficients (1.,0.8, -0.2) in cascade with the latter filter raised the maximum value of SNR to about 8.5 dB; various smoothers tried in conjunction with the former filter only degraded the performance of the system, by up to 2 dB.

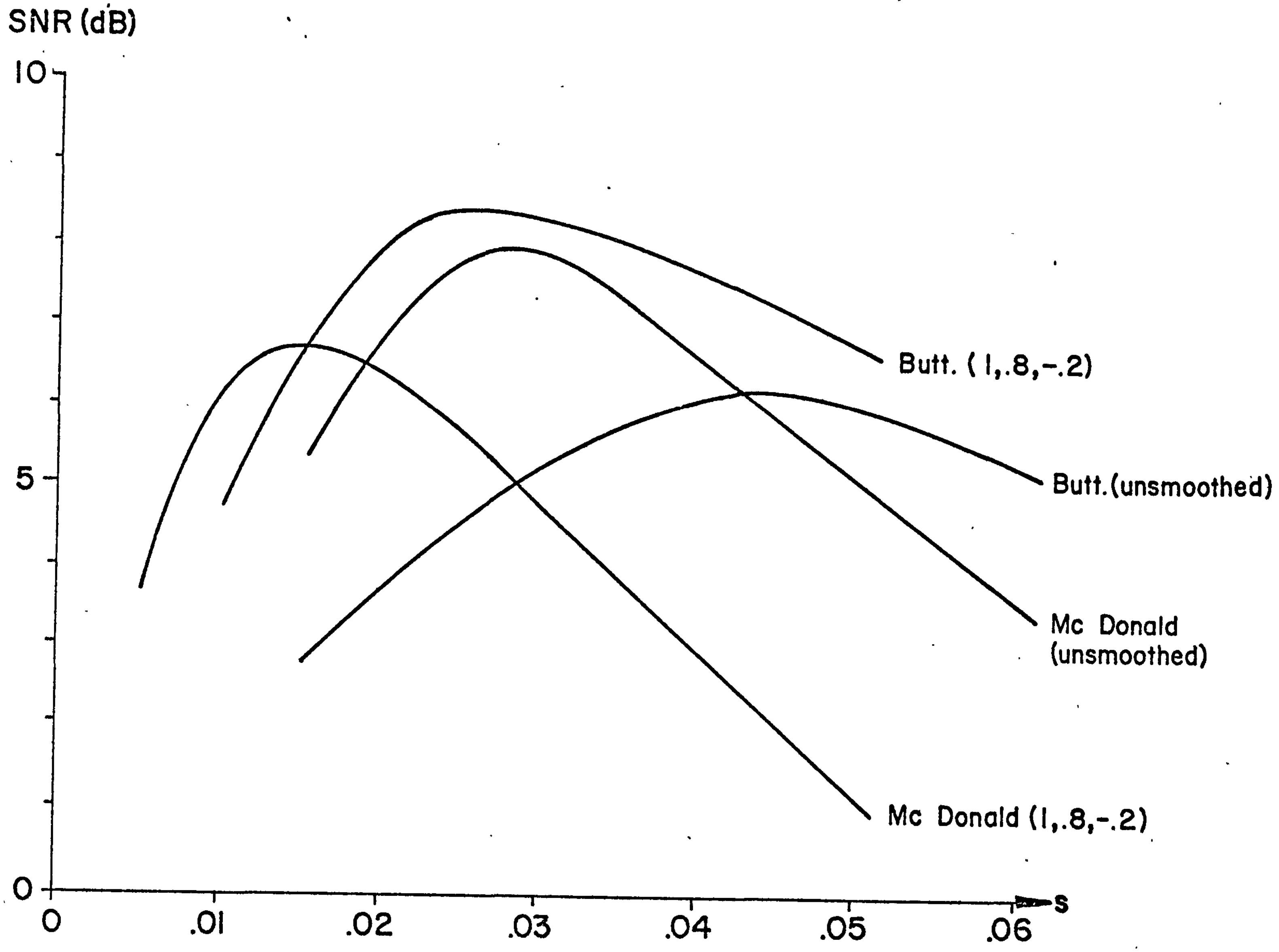


Fig. 7.3 SNR vs. s: M=1, f_s= 8 KHz, R=1

In chapter 6 the SNR of a DPCM system was expressed as $\frac{1}{K_q} \cdot \frac{1}{\sigma_p^2}$, or $\frac{1}{K_q}$.SNI , where the SNI produced by a given reconstruction filter can be calculated if the encoder is assumed to employ infinitely-fine quantization. However, this prediction of SNR breaks down with the coarse quantization used here. If we assume that the probability density of the prediction error is a two-sided exponential (that is, $f_p(x) = \frac{a}{2} e^{-a|x|}$; for experimental results supporting this assumption, see Stroh and Paez [12]), $\frac{1}{K_q}$ = 3.01 dB at R=1. From the theoretical values of SNI for the unsmoothed filters above, we would expect to reach a maximum SNR value of 13dB with the Butterworth-matched filter and about 15 dB with the McDonald-matched filter, rather than the observed values of 6.5 dB and 8 dB. However, the difference between the two values of SNR does seem to reflect a corresponding difference in the SNI values.

Increasing M (from M = 1, used above) produced larger signal-to-noise ratios for every reconstruction filter and at every value of s, with the amount of SNR improvement varying widely with different filters. This is illustrated by Fig. 7.4, in which SNR is plotted as a function of s for the unsmoothed Butterworth-matched filter and the McDonald-matched filter followed by a smoother with coefficients (.5,.5), for M = 1,2,4. The increase in maximum SNR between M = 1 and M = 4 for the former filter is only .2 dB, while for the latter it is 5.5 dB.

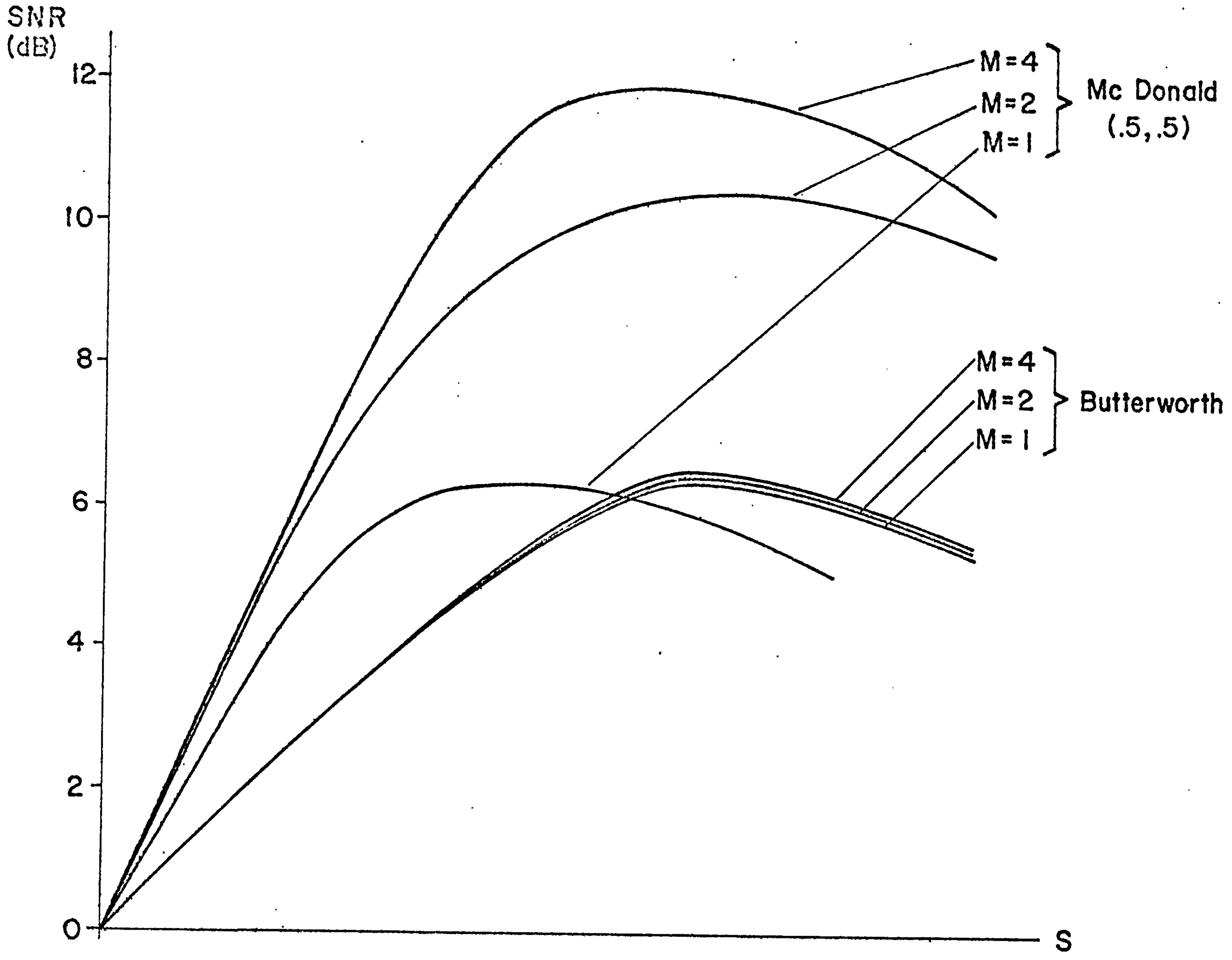


Fig. 7.4 SNR vs. s: M=(1,2;4), f_s= 8 KHz, R=1

The unsmoothed McDonald-matched filter also displays only a small SNR increase (.6 dB) to $M = 4$, while smoothed filters gain from 1.75 dB to the 5.5 dB improvement mentioned above.

These results indicate that the code trees produced by the smoothed filters include paths which match the speech source well but which are somehow hidden, and thus unavailable to the single-path search algorithm. The unsmoothed filters, however, are well-matched to the single-path search; it appears that the single-path algorithm chooses the correct path through the code trees produced by these filters nearly as often as does a more complete search.

2) $R = 2$, $f_s = 8$KHz.

A code rate of two bits/sample allows q_t to assume one of four quantization levels, which we chose to be ($\pm s$, $\pm rs$) where $0 \leq r \leq 1$. For the single-path search, where $\{p\}$ is simply quantized, the value of r (denoted r_{opt}) which minimizes the mean-square quantization error can be calculated for an assumed probability density of p_t. If, as above, we assume this density to be a two-sided exponential, $r_{opt} = .229$. For comparison, assuming a uniform distribution gives $r_{opt} = .333$, while a Gaussian distribution gives $r_{opt} = .30$. To check the assumption of an exponential density function for p_t, and to determine the optimal value of r for $M > 1$, where the analytical determination of r_{opt} does not apply, encoders employing two different reconstruction filters were evaluated at $M = 1$ and $M = 4$. The value of s was varied from .01 to .07, and that of r from .1 to .4 . The two filters used were:

1) Butterworth-matched, with rate-distortion derived smoother; coefficients (1.89, 1.20, .90, .46, .15)

2) A "triangular" filter, coefficients (1.0, .8, .6, .4, .2)

The contour plots of Fig. 7.5 (a) and (b) show the SNR obtained at $M = 1$ for varying s and r. Filter 1) reaches a maximum SNR of 16.6 dB at $(s,r) = (.04, .25)$; filter 2) gives 14.15 dB at (.045, .25) and 14.13 dB at (.045, .20). Fig. 7.5 (c) and (d) are for $M = 4$; here filter 1) reaches 18.1 dB at (.04,.25), and filter 2) 15.1 dB at (.04,.2). These results agree well with the calculated value of r_{opt} for the exponential density function, and suggest that the value of M does not significantly affect r_{opt}. On this basis, we chose to hold r constant at .23 for the tests performed at $R = 2$.

For a two-bit quantizer optimized for the exponential distribution, $\frac{1}{K_q} = 7.54$ dB. With the Butterworth-matched filter, then, we would expect an SNR of 17.2 dB at $M = 1$. With the McDonald-matched filter, we would expect 19.5 dB. Fig. 7.6 shows the $M = 1$ performance of these two filters and of some smoothed versions of them. The unsmoothed Butterworth-matched achieves an actual maximum SNR of 16.1 dB, only 1.1 dB short of the predicted value. The unsmoothed McDonald-matched reaches only 14.0 dB, some 5.5 dB below the prediction. In both cases, however, the discrepancy between the predicted and observed values of maximum SNR is less

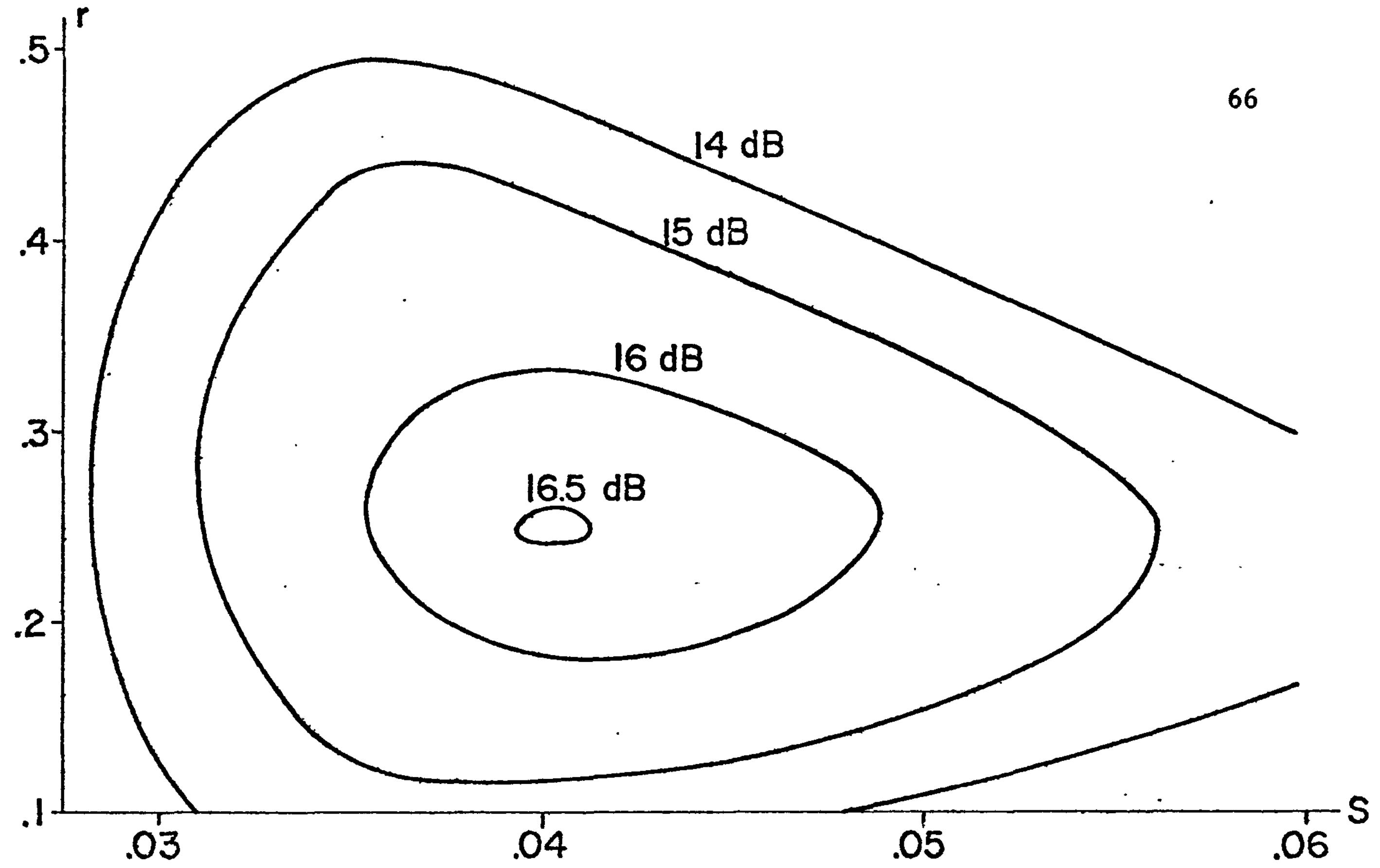


Fig. 7.5 (a)

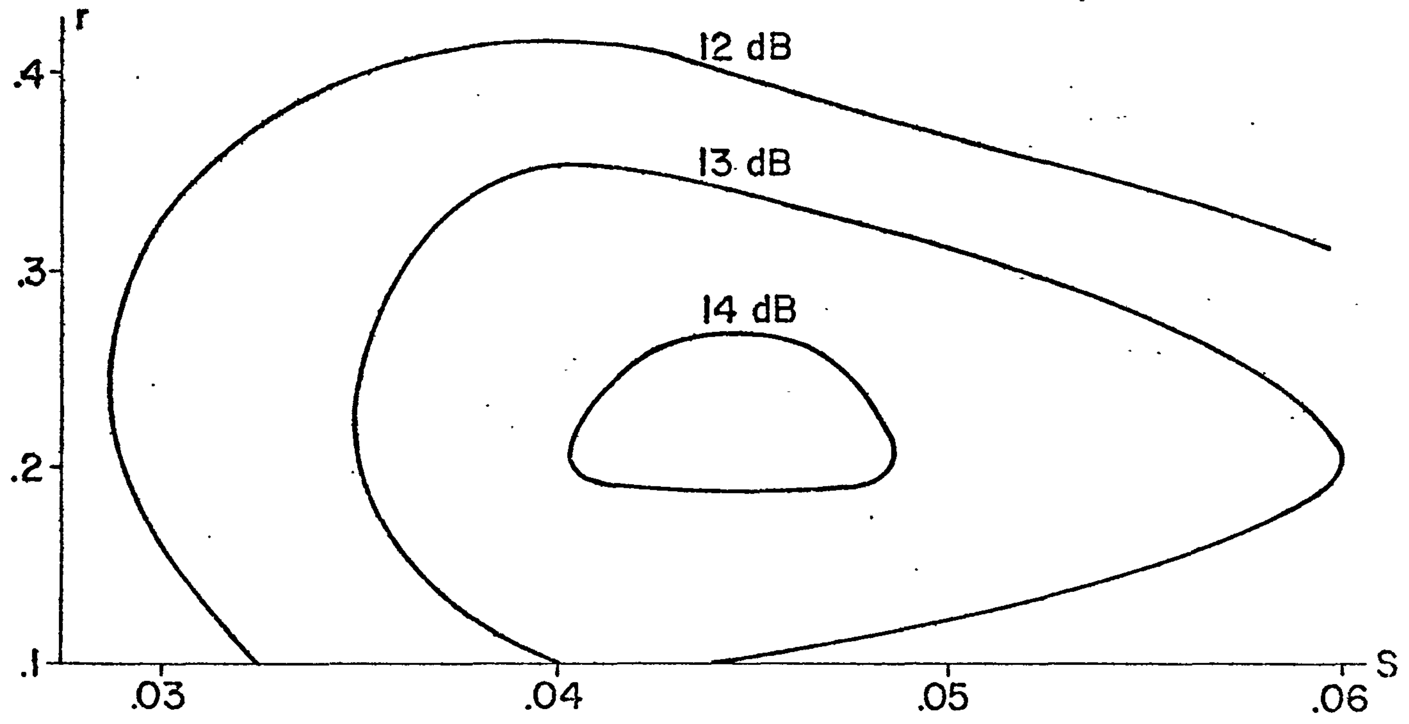


Fig. 7.5 (b) SNR vs. R, s

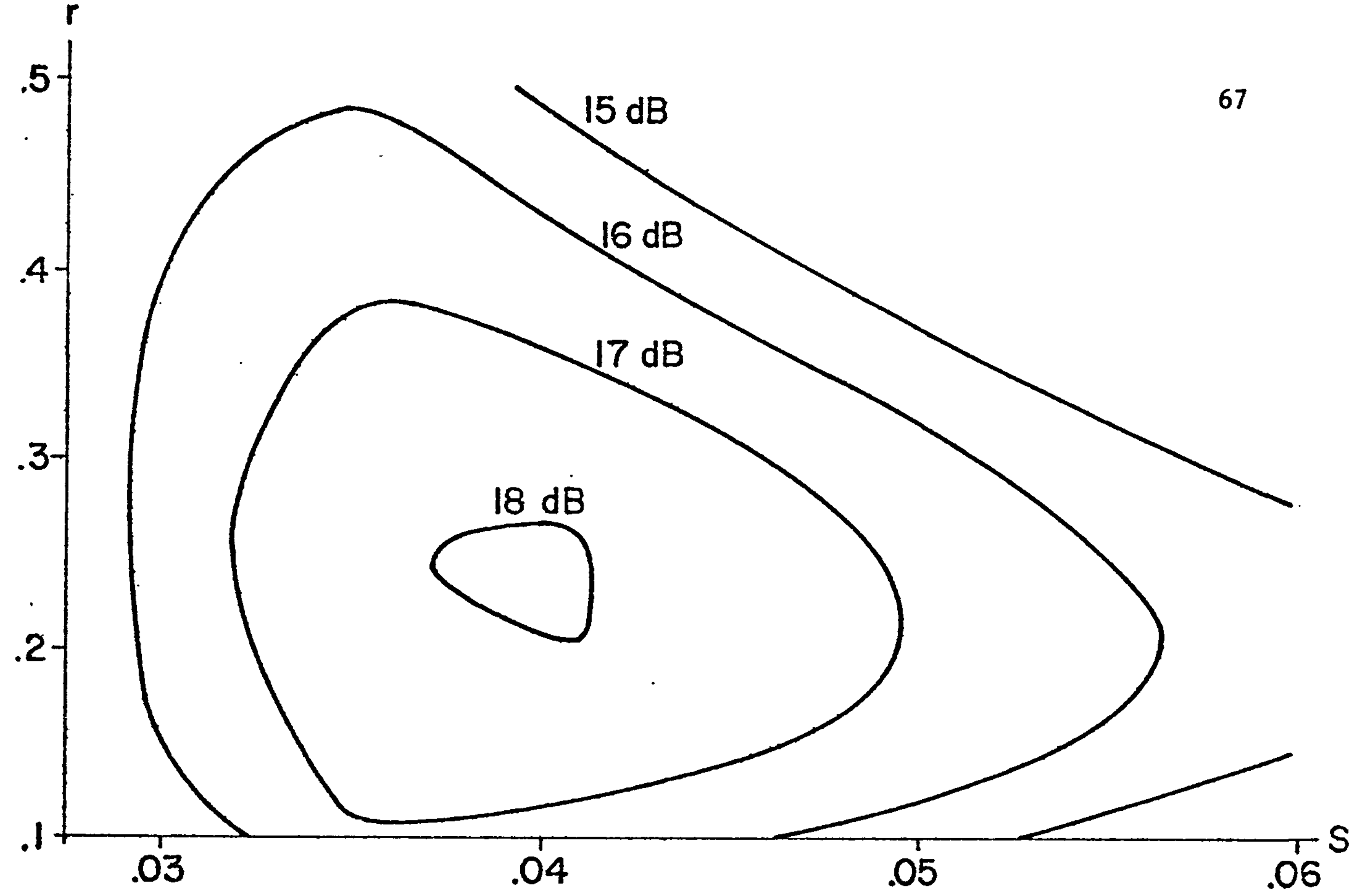


Fig. 7.5 (c)

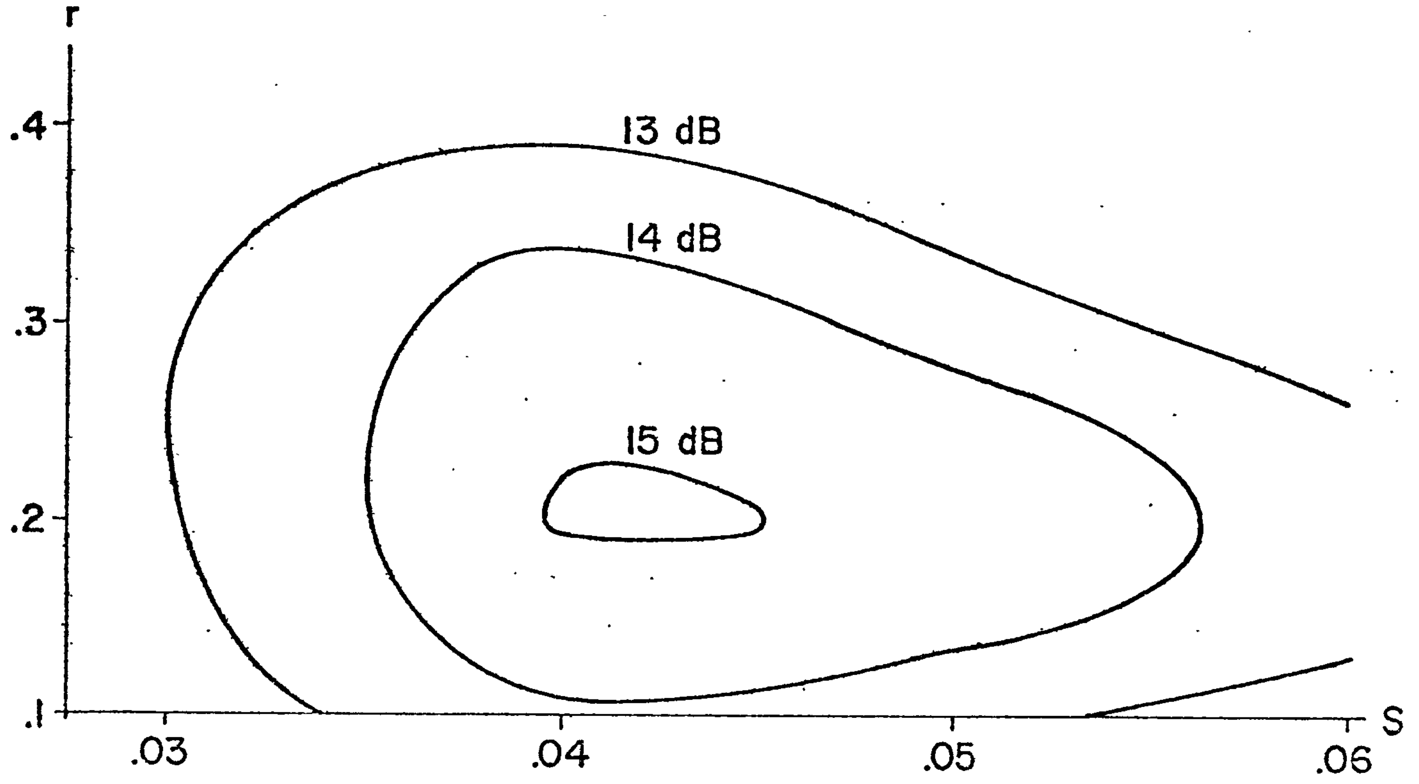


Fig. 7.5 (d)

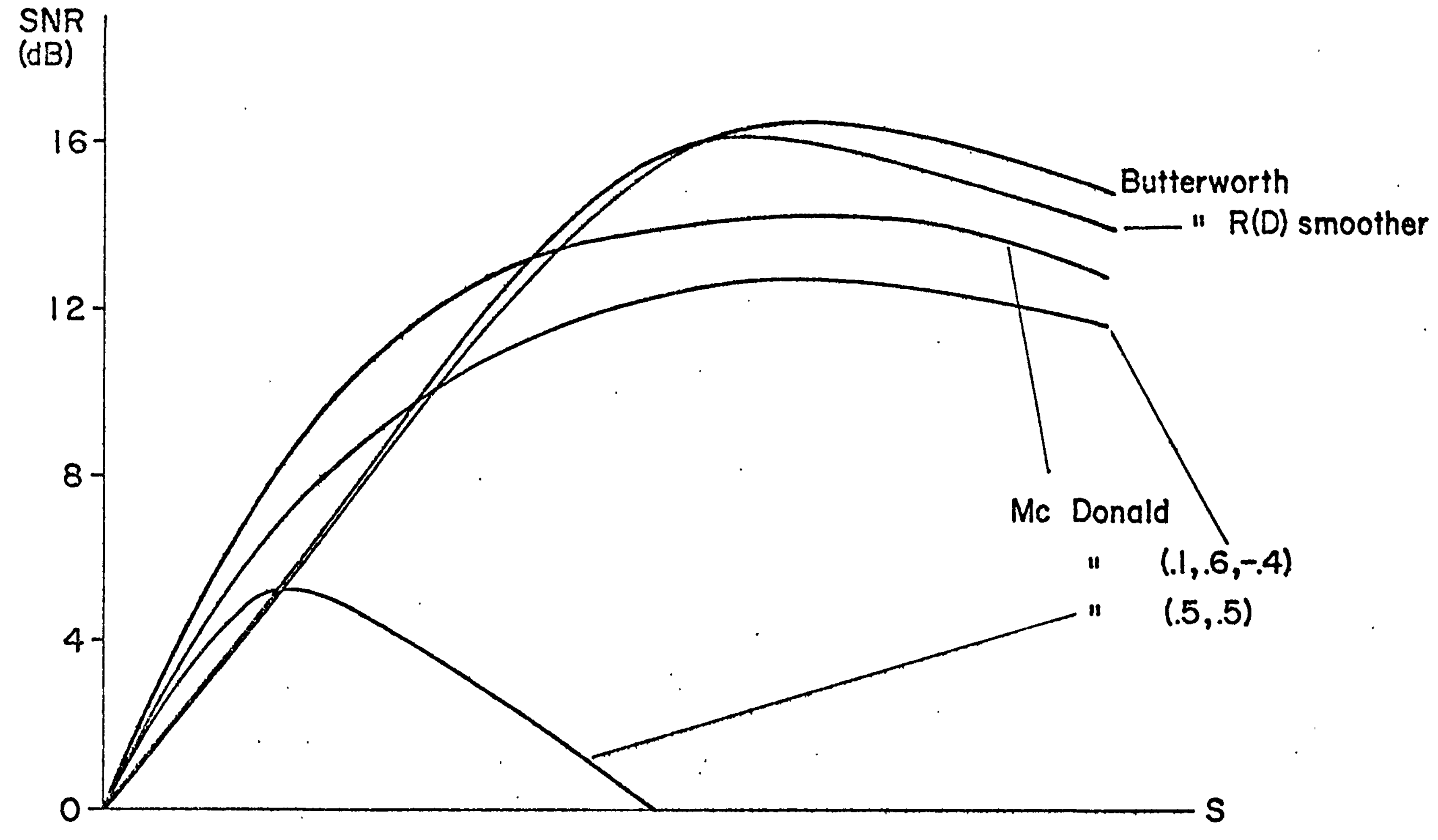


Fig. 7.6 SNR vs. s: M=1, f_s = 8 KHz, R=2

than that at $R = 1$.

The best filter at $M = 1$ was the Butterworth-matched with rate-distortion smoother, which gave a maximum SNR of 16.5 dB. The other smoothers tried degraded the performance of the system at $M = 1$. The most striking example of a smoother causing poorer performance occurred with the (.5,.5)-smoothed, McDonald-matched filter. Here the maximum SNR was only 5.2 dB, actually 1.1 dB worse than the same filter at $R = 1$.

The reason for this apparently anomalous result becomes clear when one attempts to calculate the SNI of this reconstruction filter. To obtain the correct form of the filter for the application of the SNI calculation algorithm, it is necessary to find the inverse of the filter transfer function; an inverse which, for this filter, does not converge. This implies that the ideal, perfect-quantization encoder matched to this reconstruction filter would be unstable. The coarse quantizers we have employed act as limiters to prevent complete encoder failure; however, the four-level ($R = 2$) quantizer does not limit the encoder instability to the same degree as the two-level ($R = 1$) quantizer. The greater tendency to instability thus apparently outweighs the advantage of finer quantization, and poorer performance results at the higher code rate.

This latent instability is not unique to this one filter; in fact, all our heuristically-chosen smoothers, possessing as they do a zero at $f_s/2$, produce reconstruction filters with divergent inverses. Of the smoothers we used, only the rate-distortion derived smoothers are free of this characteristic. Despite this seemingly severe failing, the heuristic smoothers often yield good performance, particularly at larger values of M. It appears that the multi-path search tends to damp out possible encoder instability; the ability of such a search to postpone the choice of a particular path map digit until the consequences of that choice have been evaluated can explain this effect.

The use of larger values of M again improved the performance of all the filters (see Fig. 7.7). The unsmoothed Butterworth-matched filter again showed the least improvement (1.35 dB), while the McDonald-matched (.5,.5)-smoothed gained 10.6 dB from M = 1 to M = 8. Other filters showed gains of from 1.8 to 6.9 dB. Generally, the better the performance of the filter at M = 1, the less was gained at higher M, and the worse it performed at M = 8. The best SNR obtained with any filter was 21.0 dB, with the McDonald-matched, (1.0, .6, -.4)-smoothed filter.

As was found for the code rate of one bit/sample, the code trees produced by the smoothed reconstruction filters appear to have better paths which are harder to find than the somewhat poorer, easily-found paths associated with the unsmoothed filters. The strong dependence on M of the choice of the best code tree agrees with results found in the previously cited digital-data case.

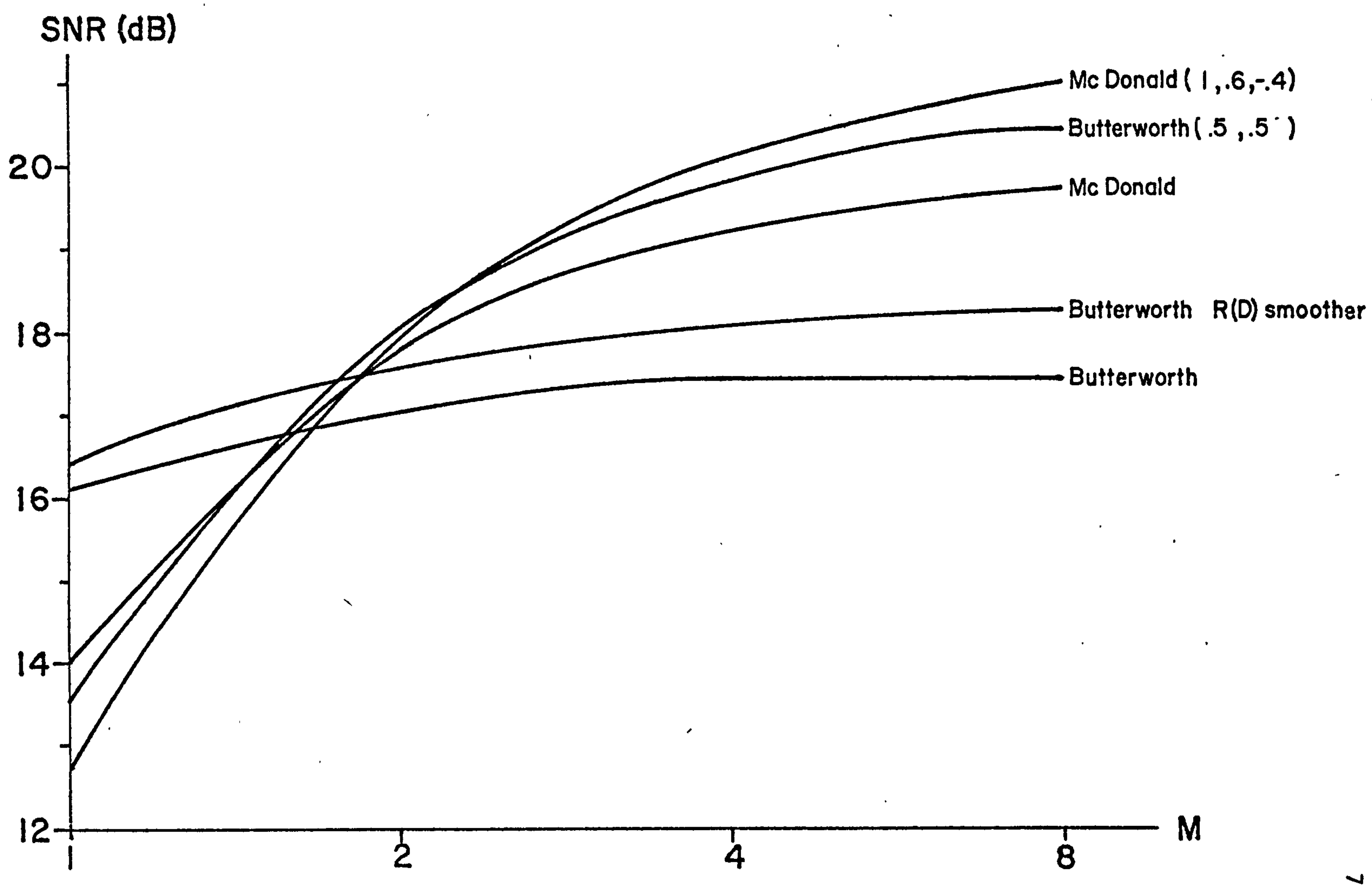


Fig. 7.7 Maximum SNR vs. M: f_s=8 KHz, R=2

3) $R = 2$, $f_s = 5$KHz.

Informal listening tests suggested that speech limited to bandwidths of between 2KHz and 2.5KHz, rather than the traditional 4KHz, is reasonably intelligible in the absence of distortion or other interference. To determine the improvement in SNR which the M-algorithm could yield for such a source, tests were performed at a sampling frequency of 5KHz, with the speech filtered before sampling as described earlier.

A code rate of two bits/sample was employed, with $r = .23$ as for the 8 KHz case. The filters tested were matched to:

1) The actual data autocorrelation function

2) A two-pole Butterworth spectrum, $f_c = 500$ Hz

3) A two-pole Butterworth spectrum, $f_c = 1$ KHz

Smoothers with coefficients (.5,.5) and (1.0, .6, -.4) were applied to the filters of 2) and 3) .

Fig. 7.8 shows the maximum SNR attained by each filter as M varied from 1 to 8. It is evident that the 500 Hz Butterworth and the data-matched filters perform nearly identically, while the 1 KHz Butterworth-matched is somewhat worse. With smoothing, the performance is degarded at $M = 1$, but is improved by up to 2.5 dB at $M = 8$. The best SNR (about 15 dB) was obtained with the 500 Hz Butterworth-matched, (1.0, .6, -.4)-smoothed filter at $M = 8$.

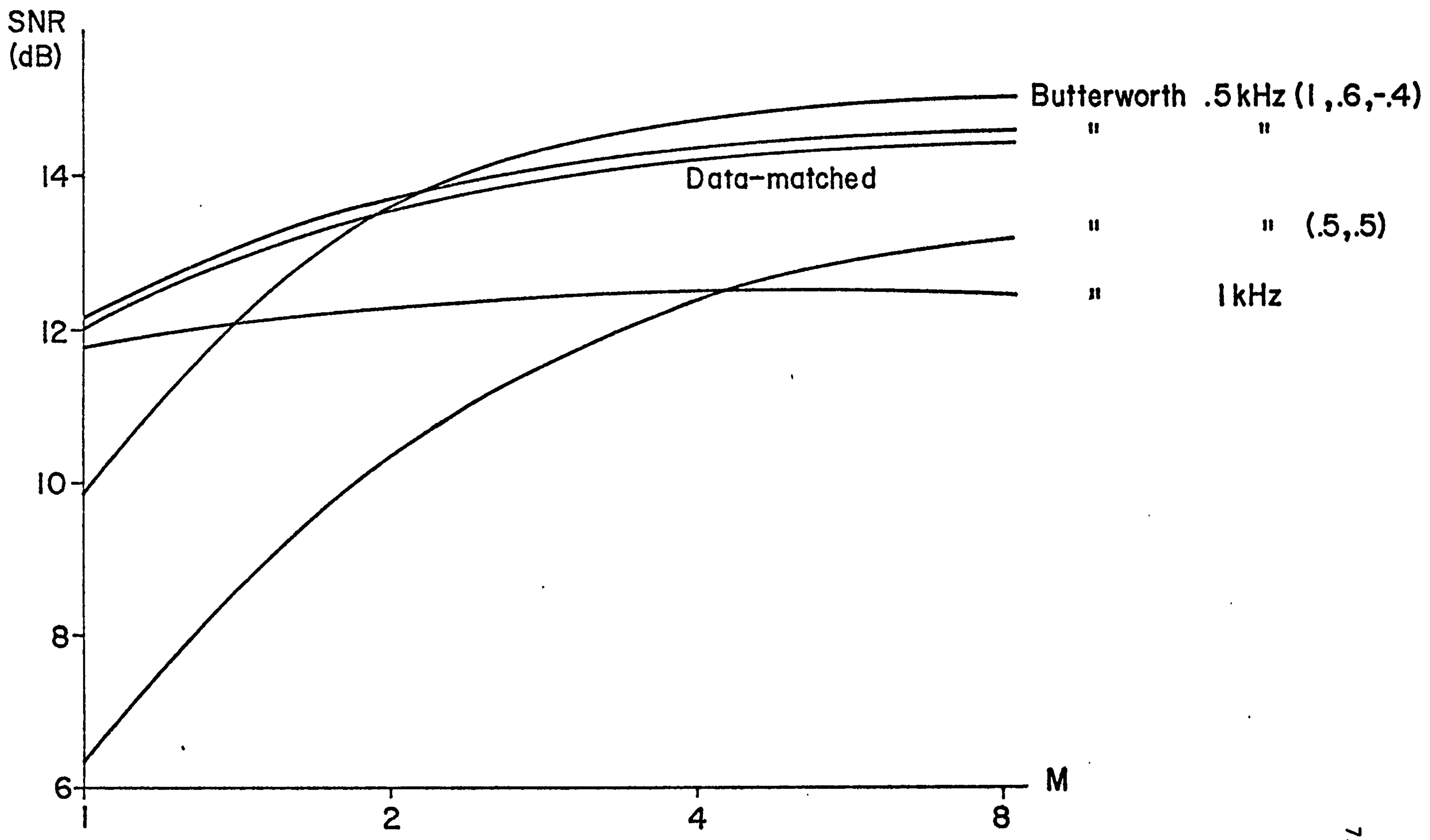


Fig. 7.8 Maximum SNR vs. M: f_s = 5 KHz, R=2

Table 7.1 summarizes the results obtained for the three choices of sampling frequency and code rate, listing the highest value of SNR produced by each filter at each value of M. The SNR improvement directly attributable to the use of the multi-path search algorithm is revealed by Fig. 7.9, in which the maximum SNR attained with any of these filters is plotted as a function of M for each choice of sampling frequency and code rate. The use of M = 8 yields values of SNR from 2.8 to 4.6 dB greater than those found at M = 1, with the largest improvement occurring at the higher code rate and sampling frequency. The pronounced flattening of the curves by M = 8 suggests that most of the improvement available through multi-path searching has already been obtained by this point.

TABLE 7.1

MAXIMUM SNR ATTAINABLE WITH EACH

RECONSTRUCTION FILTER (dB)

A) $R = 1$, $f_s = 8$ KHz

Filter	M = 1	M = 2	M = 4
McDonald	7.87	8.33	8.47
" (.5,.5)	6.25	10.35	11.75
" (1.,.8,-.2)	6.71	10.35	11.45
" (1.,.6,-.4)	7.69	10.27	11.23
" (1.,.5,-.5)	7.67	10.08	10.77
Butt. 1 KHz	6.25	6.41	6.45
" (.5,.5)	7.90	10.68	11.13
" (1.,.8,-.2)	8.44	10.27	10.65
" (1.,.6,-.4)	8.03	9.53	9.79

TABLE 7.1 - CONTINUED

B) $R = 2$, $f_s = 8$ KHz

Filter	M = 1	M = 2	M = 4	M = 8
McDonald	14.01	17.79	19.33	19.74
" (.5,.5)	5.21	9.24	13.51	15.83
" (1.,.6,-.4)	12.71	17.93	20.16	21.01
Butt. 1 KHz	16.12	17.07	17.42	17.47
" (.5,.5)	13.53	18.08	19.86	20.46
" (1.,.6,-.4)	14.66	17.28	18.18	18.34
" R(D) smoother	16.45	17.59	18.12	18.27

C) $R = 2$, $f_s = 5$ KHz

Filter	M = 1	M = 2	M = 4	M = 8
Data-matched	12.05	13.66	14.26	14.47
Butt. 500 Hz	12.23	13.71	14.39	14.63
" (.5,.5)	6.25	10.35	12.36	13.20
" (1.,.6,-.4)	9.82	13.61	14.66	15.07
Butt. 1 KHz	11.80	12.27	12.41	12.43
" (.5,.5)	10.66	13.49	14.64	14.89
" (1.,.6,-.4)	10.85	12.86	13.39	13.49

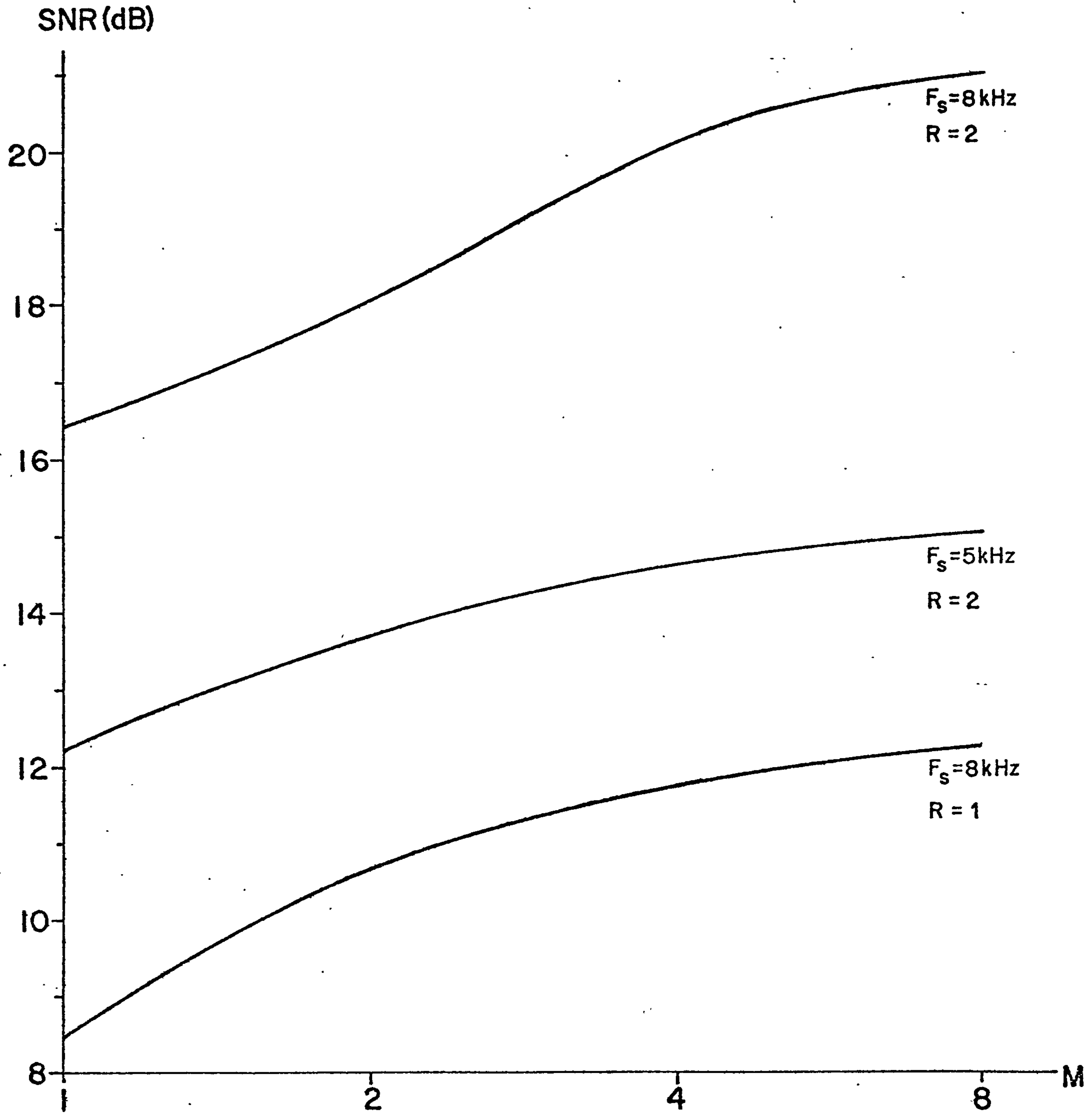


Fig. 7.9 Maximum SNR vs. M: f_s = 8 KHz, R=1; f_s = 5 KHz, R=2; f_s = 8KHz, R=2

CHAPTER VIII

CONCLUSIONS

We have applied the concepts of tree coding and of rate-distortion theory to the design of source coding systems for continuous-alphabet sources, and have shown that a simple search algorithm (the M-algorithm) coupled with appropriate code trees can perform significantly better than conventional methods relying on classical quantization. The amounts of storage and computation required to implement the improved coders, while larger than those of conventional systems, are well within present limits of practical feasibility.

With an uncorrelated source, the room for improvement between conventionally-attainable performance and the rate-distortion limit is not large, typically only one or two dB when mean-square error is the fidelity criterion. About half this potential improvement can be realized with the M-algorithm at a moderate level of complexity. With the correlated speech source, however, far greater gains can be made. We have observed improvements in SNR of nearly six dB over conventionally-optimized DPCM, again at only a moderate complexity level.

It is felt that the non-stationary nature of the speech source may contribute substantially to this unexpectedly large improvement. Certainly the question of the effect of multi-path coding on non-stationary sources merits further study.

We have seen that the decoders required for single and multi-path DPCM are identical in structure; this implies that in a broadcast system where one transmitter serves many receivers the quality of reproduction of the entire system could be improved by modifying the one encoder to use a multi-path algorithm. In such a case, the extra cost and complexity of the single more complex encoder would be negligible in comparison to the total system.

Though we have concentrated on DPCM systems for speech transmission, it is important to recognize that the concept of multi-path tree searching can be applied to any source coding scheme, and that some improvement in performance over the single-path encoding algorithm can always be obtained.

APPENDIX A

PERFORMANCE OF THE EXHAUSTIVE SEARCH ON THE EXPONENTIAL METRIC TREE

We assume a binary metric tree in which the increase in path metric (μ, the metric increment) from one level to the next along any path is chosen according to the probability density $f_\mu(x)$. We denote the probability density of λ_k, the least metric attainable with an exhaustive search of the tree from a node at level k to a node at the last level (L) as $f_{\lambda_k}(x)$. Extending the search one level further back into the tree involves choosing for each node at level k-1 that branch of the two available which possesses the smaller path metric; these metrics are the sum of two components, one distributed as $f_{\lambda_k}(x)$ and one as $f_\mu(x)$, and have density function $f_{\Sigma_k}(x)$, where

$$f_{\Sigma_k}(x) = f_{\lambda_k}(x) * f_\mu(x) \tag{A.1}$$

Then $f_{\lambda_{k-1}}(x)$, the density of the minimum of two r.v.'s with density $f_{\Sigma_k}(x)$, is given by

$$f_{\lambda_{k-1}}(x) = 2f_{\Sigma_k}(x)\left[1-F_{\Sigma_k}(x)\right] \tag{A.2}$$

If $f_\mu(x)$ is known, we may iterate (A.1) and (A.2) to obtain the density of the minimum metric attainable with a tree of any number of levels. (At the last level of the tree, $f_{\lambda_L}(x) = \delta(x)$; thus supplying a starting point for the iteration.)

In particular, let $f_\mu(x) = e^{-x}u(x)$. We assume that $f_{\lambda_k}(x)$ may be expressed in the form

$$f_{\lambda_k}(x) = \sum_{n=1}^{\infty} a_n e^{-nx} u(x) \qquad (A.3)$$

Then

$$\begin{aligned} f_{\Sigma_k}(x) &= f_{\lambda_k}(x) * e^{-x}u(x) \\ &= a_1 x e^{-x} + \sum_{n=2}^{\infty} \frac{a_n}{n-1} e^{-x} \\ &\quad - \sum_{n=2}^{\infty} \frac{a_n}{n-1} e^{-nx} \end{aligned} \qquad (A.4)$$

If $a_1 = 0$, then $f_{\Sigma_k}(x)$ can be written as

$$f_{\Sigma_k}(x) = \sum_{n=1}^{\infty} b_n e^{-nx}, \qquad (A.5)$$

where

$$\begin{aligned} b_1 &= \sum_{n=2}^{\infty} \frac{a_n}{n-1}, \\ b_n &= \frac{-a_n}{n-1}, \quad n > 1 \end{aligned} \qquad (A.6)$$

Now from (A.2), with

$$F_{\Sigma_k}(x) = \int_0^x \sum_{n=1}^{\infty} b_n e^{-n\nu} d\nu,$$

we have

$$f_{\lambda_{k-1}}(x) = \sum_{n=1}^{\infty} c_n e^{-nx}, \qquad (A.7)$$

with

$$\begin{aligned} c_1 &= 0, \\ c_n &= 2 \sum_{j=1}^{\infty} \frac{b_j b_{n-j}}{n-j}, \quad n > 1 \end{aligned} \qquad (A.8)$$

The c_n derived at level k become the a_n at level k-1, and it is thus consistent to assume that $a_1 = 0$. As we have said, $f_{\lambda_L}(x) = \delta(x)$; then $f_{\lambda_{L-1}}(x) = 2e^{-2x}$, and we may iterate to any desired depth. Given $f_{\lambda_k}(x)$, the expected value of the path metric, $\overline{\lambda}$, is

$$\overline{\lambda} = \sum_{n=1}^{\infty} \frac{a_n}{n^2} \tag{A.9}$$

for the assumed form of $f_{\lambda_k}(x)$.

The iteration of the relations defining the a_n at level k in terms of the a_n at level k-1 has been programmed on a CDC 6400 computer. The resulting values of D_L, the average path metric (or $\frac{\overline{\lambda}}{L}$) are shown in table A.1 for $1 \leq L \leq 7$. Comparison of these results with experiment (see Fig. 4.1) shows very good agreement, validating the analysis described above. However, the particular form assumed for $f_{\lambda_k}(x)$ produces values of a_n which increase seemingly without limit as L increases. This eventually leads to a breakdown in the computational procedure due to finite computer word length; an alternate form for $f_{\lambda_k}(x)$ may exist which would be more practical for numerical evaluation.

TABLE A.1

AVERAGE PATH METRIC FOR THE EXHAUSTIVELY-SEARCHED EXPONENTIAL METRIC TREE

No. of Levels Searched	Average Per-Letter Path Metric
1	.5000
2	.4584
3	.4300
4	.4091
5	.3928
6	.3796
7	.3687

APPENDIX B

PERFORMANCE OF THE M-ALGORITHM ON THE EXPONENTIAL METRIC TREE

We derive a method of calculating the performance of the M-Algorithm (with $M = 2$, 1 infinite) on a probabilistically-generated binary metric tree and apply the method to the exponential tree of Appendix A.

Assume the algorithm has reached some level k. Denote the lesser of the path metrics of the two paths followed as ρ and the greater as $\rho + \gamma$. Denote the corresponding quantities at the next level as ρ' and $\rho' + \gamma'$. The average per-letter distortion, D, is then the expected value of $\Delta\rho = \rho' - \rho$.

We first calculate the conditional expectation of $\Delta\rho$ for a given value of γ, $E(\Delta\rho|\gamma)$. We next derive the conditional density function $f_{\gamma'}(x|\gamma)$. An iteration of this relation yields a steady-state solution for $f_\gamma(x)$, and we find the average distortion as

$$D = \int_0^\infty E(\Delta\rho|x)\, f_\gamma(x)\, dx$$

Four path metrics are evaluated at level k+1; these are $\rho+\mu_1$, $\rho+\mu_2$, $\rho+\gamma+\mu_3$, and $\rho+\gamma+\mu_4$, where the μ_i are chosen independently from the density function $f_\mu(x)$. We then have

$$\Delta\rho = \text{Min}\ (\rho+\mu_1, \rho+\mu_2, \rho+\gamma+\mu_3, \rho+\gamma+\rho_4) - \rho$$
$$= \text{Min}\ (\text{Min}\ (\mu_1,\mu_2), \gamma + \text{Min}\ (\mu_3,\mu_4)\ )$$

If we let $\hat{\mu}$ represent the minimum of two r.v.'s distributed as the μ_i, then

$$f_{\hat{\mu}}(x) = 2f_{\mu}(x)\ (1-F_{\mu}(x)\ )$$
$$F_{\hat{\mu}}(x) = 1-(1-F_{\mu}(x)\ )^2 \quad ,$$

and

$$f_{\Delta\rho}(x|\gamma) = f_{\hat{\mu}}(x)\ (1-F_{\hat{\mu}}(x-\gamma)\ ) + f_{\hat{\mu}}(x-\gamma)\ (1-F_{\hat{\mu}}(x)\ ) \qquad (B.2)$$

Assuming the form of Appendix A for $f_{\mu}(x)$,

$$f_{\hat{\mu}}(x) = 2e^{-2x}\ u(x)$$
$$F_{\hat{\mu}}(x) = 1 - e^{-2x}\ u(x)$$

Substitution into (B.2) gives

$$f_{\Delta\rho}(x|\gamma) = 0 \quad , x<0$$
$$= 2e^{-2x} \quad , 0<x<\gamma$$
$$= 4e^{-4x}e^{2\gamma} \quad , \gamma<x \qquad (B.3)$$

The conditional expectation of $\Delta\rho$ is then

$$E(\Delta\rho|\gamma) = \int_{-\infty}^{\infty} x\ f_{\Delta\rho}(x|\gamma)\ dx$$
$$= \frac{1}{2}\ (1 - \frac{e^{-2\gamma}}{2}) \qquad (B.4)$$

We now consider the conditional density function $f_{\gamma'}(x|\gamma)$. We first calculate the probability that $\gamma' > x$, given γ. We denote this probability as $G_{\gamma'}(x|\gamma) = 1 - F_{\gamma'}(x|\gamma)$. The event $\gamma' > x$ can occur in four mutually exclusive ways:

1) $\mu_2 > \mu_1+x,\ \mu_3 > \mu_1+x-\gamma,\ \mu_4 > \mu_1+x-\gamma$
2) $\mu_1 > \mu_2+x,\ \mu_3 > \mu_2+x-\gamma,\ \mu_4 > \mu_2+x-\gamma$
3) $\mu_1 > \mu_3+x+\gamma,\ \mu_2 > \mu_3+x+\gamma,\ \mu_4 > \mu_3+x$
4) $\mu_1 > \mu_4+x+\gamma,\ \mu_2 > \mu_4+x+\gamma,\ \mu_3 > \mu_4+x$

1) and 2) occur with probability

$$P_1 = \int_0^\infty f_\mu(\nu)\ G_\mu(\nu+x)\ G_\mu^2(\nu+x-\gamma)\ d\nu \quad ,$$

and 3) and 4) with probability

$$P_3 = \int_0^\infty f_\mu(\nu)\ G_\mu(\nu+x)\ G_\mu^2(\nu+x+\gamma)\ d\nu$$

where $G_\mu(x) = 1-F_\mu(x)$. Performing these integrations for the assumed distribution of μ gives

$$\begin{aligned} G_{\gamma'}(x|\gamma) &= 0 && , x<0 \\ &= e^{-x}\ (1-e^{-2\gamma}\sinh(2x)\ ) && , 0<x<\gamma \\ &= e^{-3x}\ \cosh(2\gamma) && , x>\gamma \end{aligned}$$

$$\begin{aligned} \text{Now } f_{\gamma'}(x|\gamma) &= -\frac{d}{dx}\ G_{\gamma'}(x|\gamma) \\ &= 0 && , x<0 \\ &= e^{-x} + \frac{e^{-2\gamma}}{2}(e^{x}+3e^{-3x}) && , 0<x<\gamma \\ &= 3e^{-3x}\ \cosh(2\gamma) && , x>\gamma \end{aligned} \qquad (B.5)$$

We can now find $f_{\gamma'}(x)$ as

$$f_{\gamma'}(x) = \int_0^\infty f_{\gamma'}(x|\gamma)\ f_\gamma(\nu)\ d\nu \qquad (B.6)$$

By iteration of (B.6), $f_\gamma(x)$ can be found for any desired level of the tree. At k=1, $\gamma=0$. This leads immediately to

$$f_\gamma(x) = 3e^{-3x} \qquad , k = 2$$

and after some calculation to

$$f_\gamma(x) = 5.4e^{-3x} + 3.2e^{-4x} \qquad , k = 3$$

These results for the first few levels suggest that we assume the following form for $f_\gamma(x)$:

$$f_\gamma(x) = \sum_{n=3}^{\infty} a_n e^{-nx} \qquad (B.7)$$

Performing the integration in (B.6) we find that

$$f_{\gamma'}(x) = \sum_{n=3}^{\infty} b_n e^{-nx} \text{ , where}$$

$$b_3 = \sum_{n=3}^{\infty} a_n \frac{3n}{(n+2)\cdot(n-2)}$$

$$b_n = -a_{n-1} \frac{4n}{(n-1)\cdot(n+1)\cdot(n-3)} \quad ,n>3 \quad \text{(B.8)}$$

The fact that the assumed form for $f_\gamma(x)$ is consistent between levels allows the iteration of (B.6) to be performed on the coefficients of (B.7) rather than on the actual density functions, allowing a simple implementation of the iteration on a digital computer. We have employed a CDC 6400 computer for this purpose, repeating the iteration until the solution stabilized. (This occurred at k=10.) The resulting coefficients for the steady-state solution of (B.7) are:

$$a_3 = 7.114$$
$$a_4 = -7.588$$
$$a_5 = 3.162$$
$$a_6 = -.723$$
$$a_7 = .105$$
$$a_8 = -.011$$
$$a_9 = .001$$

Performing the integration of (B.1), we find that

$$D = \frac{1}{4}\left(2 - \sum_{n=3}^{\infty} a_n \frac{1}{n+2}\right)$$

$$= \underline{\underline{.3675}}$$

APPENDIX C

OPTIMAL QUANTIZATION WITH A RANGE OF VARIANCE

A one-bit quantizer with quantization levels (Q,-Q) is to be used to reproduce source letters chosen from the $\mathcal{N}(0,\sigma^2)$ distribution, where σ^2 can range from .5 to 2. We wish to minimize the maximum ratio of mean-square quantization error (D) to source variance over this range.

For a given value of σ^2,

$$D = \sigma^2 - 2\alpha\sigma Q + Q^2 \qquad \text{(C.1)}$$

where $\alpha = \sqrt{\frac{2}{\pi}}$, and

$$\hat{D} = \frac{D}{\sigma^2} = 1 - \frac{2\alpha Q}{\sigma} + \frac{Q^2}{\sigma^2} \qquad \text{(C.2)}$$

The quantizer optimized for $\sigma^2 = 1$ has $Q = \alpha$, and yields the following values of $\hat{D}$:

$$\begin{aligned} \hat{D} &= .364 \quad , \sigma^2 = 1 \\ &= .474 \quad , \sigma^2 = .5 \\ &= .419 \quad , \sigma^2 = 2 \end{aligned}$$

The maximum value of $\hat{D}$ is minimized by equating $\hat{D}(.5)$ and $\hat{D}(2)$:

$$1 - \frac{2\alpha Q}{\sqrt{.5}} + \frac{Q^2}{.5} = 1 - \frac{2\alpha Q}{\sqrt{2}} + \frac{Q^2}{2} \qquad \text{(C.3)}$$

Solving (C.3) gives $Q = \frac{4\alpha}{3\sqrt{2}}$, and yields

$$\begin{aligned} \hat{D} &= .366 \quad , \sigma^2 = 1 \\ &= .434 \quad , \sigma^2 = .5,\ 2 \end{aligned}$$

REFERENCES

[1] T. Berger, Rate Distortion Theory: A Mathematical Basis for Data Compression, Prentice-Hall Inc., Englewood Cliffs, N. J., 1971.

[2] F. Jelinek and J. B. Anderson, "Instrumentable Tree Encoding of Information Sources," IEEE Trans. Inform. Theory, vol. IT-17, pp. 118-119, Jan. 1971.

[3] J. B. Anderson, "A Stack Algorithm for Source Coding with a Fidelity Criterion," Internal report CRL-4, Commun. Res. Lab., McMaster Univ., Hamilton, Ont.

[4] J. B. Anderson and F. Jelinek, "A 2-Cycle Algorithm for Source Coding with a Fidelity Criterion," IEEE Trans. Inform. Theory, vol. IT-19, pp. 77-92, Jan. 1973.

[5] R. G. Gallager, "Tree Encoding for Sources with a Distortion Measure," Unpublished, Dept. of Elec. Eng., M.I.T., Cambridge, Mass., 1972.

[6] J. B. Anderson, Personal Communication, Jan. 1974.

[7] J. B. Anderson, "Tree Coding for A-to-D Conversion of Speech," Internal report CRL-9, Commun. Res. Lab., McMaster Univ., Hamilton, Ont.

[8] J. B. O'Neal, Jr. and R. W. Stroh, "Differential PCM for Speech and Data Signals," IEEE Trans. Commun., vol. COM-20, pp. 900-912, Oct. 1972.

[9] J. Max, "Quantizing for Minimum Distortion," Trans. IRE, vol. IT-6, pp. 7-12, 1960.

[10] R. A. McDonald, "Signal-to-Noise and Idle Channel Performance of Differential Pulse Code Modulation Systems--Particular Applications to Voice Signals," Bell Syst. Tech. J., vol. 45, pp. 1123-1151, Sept. 1966.

[11] Speech Analysis/Synthesis Survey test tape, International Conference on Speech Communication and Processing, Boston, Mass., 1972.

[12] R. W. Stroh and M. D. Paez, "A Comparison of Optimum and Logarithmic Quantization for Speech PCM and DPCM Systems," IEEE Trans. Commun., vol. COM-21, pp. 752-757, June 1973.