EPM/RT - 85-13

MULTIPLE-PATH SEQUENTIAL DECODING FOR INTERSYMBOL INTERFERENCE*

by

Samir KALLEL, M.Sc.A.
and
David HACCOUN, Ph.D.
Département de Génie électrique
Ecole Polytechnique de Montréal

April 1985

* This research was supported in part by the Natural Science and Engineering Research Council of Canada and by the Formation de chercheurs et Action Concertée of Quebec.

MULTIPLE-PATH SEQUENTIAL DECODING FOR INTERSYMBOL INTERFERENCE

by

Samir KALLEL and David HACCOUN

ABSTRACT

In the high-speed transmission of digital data over bandlimited channels an efficient detection technique must take into consideration both the presence of the channel noise and that of the intersymbol interference (ISI) between neighboring pulses. By modelling the ISI as a convolutional encoding of the data the optimum Viterbi decoding can be applied to the ISI problem. However since the computational effort grows exponentially with the memory of the channel, Viterbi decoding becomes impractical when the channel memory exceeds a few symbols. In this paper we present the application of the stack algorithm of sequential decoding and some of its multiple-path variants to the ISI problem, especially over long memory channels. Computer simulation with channels having memory lenths up to 9 symbols show that the achieved error performances fall within the theoretical upper and lower bounds of an optimal decoder at only a small fraction of the computational effort of an optimal decoder. As for the computational variability it can be totally circumvented by simultaneously extending a sufficient number of paths. Finally with a simplified multiple path seuqential decoder all the drawbacks of sequential decoding may be avoided at a cost of a very slight error performance degradation, making the technique very attractive to combat ISI over long memory channels.

I INTRODUCTION

In the high-speed transmission of digital data over bandlimited channels, one of the principal impairment is the intersymbol interference (ISI) between neighboring pulses due to an insufficient channel bandwidth. Over relatively narrow-band channels the transmitted symbols tend to spread in time, and by spilling over adjacent symbols time intervals they interfere with these adjacent symbols and cause detection errors. Therefore a good detection technique must take into consideration both the presence of the channel noise and that of the ISI.

Many effective techniques to minimize the effects of ISI by channel equalization, that is by adjustments of the pulse shapes, have been developed. However these techniques are not optimal and often lead to very complex receiver structures [1] - [3]. Also, by considering ISI as a convolutional encoding of the data, where the memory of the channel is associated with the memory of the convolutional encoder, optimum or quasi-optimum decoding tehcniques using the Viterbi decoding algorithm (as for convolutional codes) have been proposed [3] - [8]. Unfortunately because the computational complexity of the Viterbi decoding algorithm grows exponentially with the memory of the channel, its application becomes impractical when the channel memory exceeds 3 or 4 symbols.

For channels with longer memory, the excessive amount of computations of Viterbi decoding has led to consider the powerful suboptimum decoding technique of sequential decoding, in particular the stack algorithm and some of its variants [9] - [11]. Contrary to Viterbi decoding where all the possibly transmitted sequences (or transmitted paths) are exhaustively examined, in sequential decoding only a small fraction of these sequences are considered by the decoder. As a consequence, the computational effort is on the average very small, but also, unfortunately variable. However the substantial advantage of sequential decoding is that the computational complexity is practically independent of the memory of the convolutional code (or that of the channel). Hence sequential decoding can be applied to the ISI problem especially over arbitrarily long memory channels. As for the computational variability, it has been shown that this drawback can be circumvented by variants of the stack algorithm called multiple-path sequential decoding [12]. In these variants the decoder simultaneously explores some number M, of the most likely paths, instead of the single most likely path of the usual sequential decoding. With such M-path algorithms it has been shown that the computational variability may be drastically reduced at a cost of a modest increase of the average decoding effort [12].

In this paper we present the application of sequential decoding to the ISI problem, in particular the stack algorithm and its multiple-path variants.

Computer simulation results with channels having memory lengths up to 9 symbols show that the error performances of the multiple-path sequential decoding fall within the theoretical upper and lower bounds of an optimal decoder, with the advantage of requiring an average computational effort only a small fraction of that of an optimal Viterbi decoder. Furthermore by properly choosing the number of extended paths M, the computational variability may be practically eliminated, making this technique especially attractive for long memory channels with severe ISI.

We assume the reader familiar with the elements of a baseband digital communication system. Figure 1 shows the discrete time white noise channel model that will be used in this paper [2], [4]. The symbols of a transmitted sequence $\{I_k\}$ may take equally likely values $\pm h$, $\pm 3h$, $\pm \ldots \pm (q-1)h$, where q is the alphabet size of the elements representing the symbols, and where 2h is the distance between successive amplitude symbol levels. The transmitter sends the discrete-time symbols at a rate 1/T symbols/s so that a channel of memory W corresponds to a discrete-time transversal filter that spans a time interval of WT seconds. The tap coefficients $\{f_i\}$ represent the amount of interference caused by neighboring symbols on a given received symbol. The passage of the input sequence $\{I_k\}$ into the discrete channel model results in the output sequence $\{y_k\}$ where each output symbol can be expressed by

$$y_k = Z_k + n_k \quad , \quad k = 0,1,2... \qquad (1)$$

where Z_k is given by:

$$Z_k = \sum_{f=0}^{W} f_j \, I_{k-j} \qquad (2)$$

and where n_k is the noise sample, considered to be Gaussian and white with zero mean and variance σ^2.

Examination of Figure 1 shows that this channel model may be regarded as a special convolutional encoder of constraint length K = (W+1),where the usual modulo-2 adders have been replaced by a single arithmetic adder connected to the shift register stages through real number multipliers. Although the analogy with convolutional encoding does not entail a coding rate or a bandwidth expansion.

Using the channel model of Figure 1 Maximum Likelihood Sequence Estimation is presented in the next section together with an introduction to Viterbi decoding and sequential decoding. Only the stack algorithm (and its M-path variant) of sequential decoding is considered in this paper. The particular sequential decoding likelihood function (or metric) to use for the ISI problem is derived and discussed in section III and results from extensive computer simulations are given in section IV. Trade-offs between error probability, computational effort and choice of metrics are provided for channels having memory 4,6 and 9 symbols.

Finally a simplified multiple-path sequential decoding algorithm suitable for ISI is presented in section V. With this simplified algorithm all the computational variability of sequential decoding is eliminated at hardly any error performance degradation, making it a very attractive alternative for ISI over long memory channels.

II MAXIMUM LIKELIHOOD SEQUENCE ESTIMATION

Using the tree and trellis structures of the transmitted sequences, maximum likelihood sequence estimation is presented together with the optimal trellis search of Viterbi decoding and suboptimal tree search of sequential decoding.

Let S_k be the state of the channel defined as the W most recent inputs preceding the current input I_k.

$$S_k = (I_{k-1}, I_{k-2}, \ldots I_{k-W}), k=1,2,\ldots \qquad (3)$$

where $I_k = 0$ for $k<0$

A sequence of input symbols can then be represented by a succession of possible states. There are q^W distinct states, and from one state we can transit to q new states, with each transition corresponding to a distinct element of the alphabet. All possible state sequences may be represented by a tree, where each node corresponds to a state of the channel, and where q branches emerge from each node. The root node of the tree is the initial state of the channel, and each new input symbol in the channel causes a corresponding transition in the tree. An input sequence of L symbols traces a particular path of L branches in the tree and therefore, there are q^L possible transmitted sequences of length L (see Figure 2).

Examination of the tree shows that at any tree depth $\ell, \ell > W$, there are q^{ℓ} tree nodes but only q^W distinct channel states. Thus, at depth ℓ there are more tree nodes than distinct states, and hence,several nodes must correspond to the same channel state; these nodes are identical and generate identical subtrees. Therefore the tree contains a huge redundancy which can be eliminated by merging together, at any same tree depth beyond W, all nodes corresponding to the same channel state. The redrawing of the tree with merging paths and redundancy eliminated is called a trellis (see Figure 3).

The objective of a maximum likelihood decoder is to determine the path in the tree (or in the trellis) that corresponds to the most likely transmitted sequence given the received sequence.

Let $\underline{Y}_L = (y_0, y_1, .. \, y_{L-1})$ be the received sequence, where the received symbols y_k are given by (1). The decoder observes $\underline{Y}_L$ and chooses the L – branch sequence $\underline{I}_L^{(m)}$ for which the likelihood function $P(\underline{Y}_L | \underline{I}_L)$ is maximum over all distinct paths, that is,

$$P[\underline{Y}_L | \underline{I}_L^{(m)}] \geq P[\underline{Y}_L | \underline{I}_L^{(m')}], \quad m' \neq m \tag{4}$$

Since the noise is white and the channel has memory W, then for any input sequence we can write,

$$P[\underline{Y}_L | \underline{I}_L] = \prod_{k=0}^{L-1} P[y_k | I_k, I_{k-1}, .., I_{k-W}] \tag{5}$$

or,

$$P[\underline{Y}_L | \underline{I}_L] = \prod_{k=0}^{L-1} P\,[y_k | Z_k] = \prod_{k=0}^{L-1} P_n(y_k - Z_k) \tag{6}$$

where $P_n(.)$ is given by:

$$p_n(\alpha) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp - (\alpha^2/2\sigma^2) \;, \; -\infty \leq \alpha \leq \infty \tag{7}$$

Taking the logarithm (6) becomes:

$$P[\underline{Y}_L | \underline{I}_L] = \sum_{k=0}^{L-1} \log \frac{1}{\sqrt{2\pi}\,\sigma} - \frac{(y_k - Z_k)^2}{2\sigma^2} \tag{8}$$

Eliminating constant terms from (8), maximizing $P[\underline{Y}_L | \underline{I}_L]$ corresponds to maximizing the total metric Γ_L defined as:

$$\Gamma_L = \sum_{k=0}^{L-1} -(y_k - Z_k)^2 \tag{9}$$

or equivalently,

$$\Gamma_L = \sum_{k=0}^{L-1} - d_k^2 \tag{10}$$

Where d_k is the Euclidian distance between the received k^{th} symbol y_k and the k^{th} branch symbol Z_k on the tree path of interest. Defining the branch metric for the k^{th} symbol as:

$$\gamma_k = -d_k^2 \tag{11}$$

the total metric is then:

$$\Gamma_L = \sum_{k=0}^{L-1} \gamma_k \tag{12}$$

An optimum sequence decoder will therefore attempt to find that information sequence for which the total metric is maximum. Examples of such decoders that are both powerful and practical are Viterbi decoder and sequential decoder. They are briefly described next.

Viterbi decoding

The Viterbi decoding algorithm is an optimum decoding procedure which determines the path having the largest cumulative metric of all possible distinct paths in the trellis. At each trellis depth, only the best (i.e. most likely) path terminating at each of the q^W distinct states is retained. At each decoding step, these q^W remaining or "surviving" paths are extended into their q single branch extensions and their metrics are computed. Then, for each group of q paths merging at each state, only the path having the largest total metric is retained. The other (q-1) paths are discarded and the procedure is repeated anew. With this procedure clearly, none of the discarded path can ever be the most likely, that is the decoding is optimum [13] - [15].

The computational complexity and amount of memory required by this algorithm both grow with the number of states, that is they grow exponentially with the memory of the channel. Practical applications of Viterbi decoding is therefore limited to channels with memory not longer than a few symbols.

The error probability of Viterbi decoding decreases exponentially with the memory of the encoder [13]. For ISI, Forney [4] has shown that for PAM signaling the symbol error probability $P(\epsilon)$ is bounded by:

$$k_1 Q\left(\frac{d_{min}}{2\sigma}\right) \leq P(\epsilon) \leq k_2 Q\left(\frac{d_{min}}{2\sigma}\right) \tag{13}$$

where d_{min} is the minimum Euclidian distance between any two paths in the tree or in the trellis, where k_1 and k_2 are constants with the same order of magnitude, and where $Q(.)$ is defined as:

$$Q(\alpha) = \frac{1}{2\pi} \int_{\alpha}^{\infty} e^{-u^2/2}\, du \tag{14}$$

Sequential decoding

Sequential decoding is a very efficient tree search algorithm that explores, one or a few paths at a time, only the most likely part of the tree. Hence this technique is suboptimum. Starting from the origin of the tree, the path selected to be searched one step further into its q branch extensions is the path that has the largest accumulated metric among those already examined. Hence, by extending only the path that appears to be the most likely, most of the computations necessary for an optimum decoding can be avoided. The idea is common to various sequential decoding algorithms, with the specific method of searching and selecting the path to be extended depending on the particular algorithms [15] - [16].

As the decoding proceeds the decoder occasionnally retreats in the tree and explores earlier and possibly incorrect paths. This backing up and extension of unlikely paths is minimized by biasing the metric in such a way that on the average it increases along the correct path and decreases along all incorrect paths [16] - [17].

With this decoding procedure the computational effort is on the average very small, but also highly variable with an asymptotic Pareto distribution, that is, a distribution whose tail decreases only algebraically [18]. This variability of the computational effort is one of the principal drawbacks of sequential decoding, and in analysing sequential decoding both the error performance and computational effort must be examined.

There are two main sequential decoding algorithms: the Fano algorithm [16] and the Zigangirov-Jelinek or Z-J algorithm [19]. In this paper only the Z-J or stack algorithm and some of its variants are considered.

In the Z-J, or stack algorithm, all the examined paths are stored in decreasing order of their metric values in a stack. The top of the stack has the largest accumulated metric and will be extended one level further along the q branches emerging from its end node. The operations of the stack decoder are thus the finding of the top node, the extension and storage of its successors, and the proper reordering of the stack. As a node is extended, it is removed from the stack.

The algorithm is then:

1) Compute the metrics of the successors of the top node of the stack and enter then in their proper in the stack.
2) Remove from the stack the node that was just extended.
3) Find the new top node. If it is the final node, Stop. Otherwise go to 1.

The information sequence is divided in blocks of L symbols (L varies from 500 to 2000 symbols) and each block is terminated by a tail of W known symbols. Thus, at the begining of a new block the decoder is resynchronised to the initial channel state. When the top node of the stack is a terminal node of the tree, decoding of the block is complete and the algorithm recovers the decoded path and delivers it to the user. A sequence decoding error occurs whenever that terminal node is not the terminal node of the correct path.

Since the number of computations needed to decode a given block of information is variable, an input buffer is required to store the incoming data waiting to be decoded. An output buffer is also used to smooth out the rate of delivery of the decoded blocks. A simplified block diagram of a sequential decoder using the stack is given in Figure 4.

In order to alleviate the computational variability of sequential decoding, a multiple path stack algorithm has been developed [12]. In this algorithm the M (rather than the single) most likely paths are simultaneously extended. Moreover, remergers may be exploited as in the Viterbi algorithm in order to eliminate redundant and useless paths from the stack, and thus help reduce the required stack storage. With this algorithm, compared to the ordinary stack algorithm, the computational variability is reduced at a cost of a somewhat larger average number of computations. Furthermore the error probability is also improved since some of the errors of the stack algorithm are corrected by the M path algorithm. The M path algorithm with remergers belongs to the class of the Generalized Stack Algorithms [12] and fills the gap between the 1-path sequential decoding and the all-path Viterbi decoding. In fact, here, when the number M of the extended paths equals q^W, this M-path algorithm becomes equivalent to the optimum Viterbi algorithm [12].

III <u>Sequential Decoding Metric for the ISI Problem</u>

As we mentioned above, when applied to sequential decoding, the branch metric given by (11) must be biased in such a way that on the average it increases along the correct path and decreases along all incorrect paths. The metric normally used with sequential decoding is called the Fano metric, and was developed for the decoding of convolutional codes [16] - [17]. However, since here the output of the channel are real numbers, and since also there is no actual coding rate as for ordinary convolutional codes, the Fano metric cannot be used as such for the ISI problem [10] - [11].

We now show that the branch metric to use should have the following form:

$$\gamma_k = \alpha - \beta d_k^2 \qquad (15)$$

where α and β are real numbers, and where d_k is the Euclidian distance between the received symbol and the corresponding symbol being examined in the tree.

In order to help distinguish the correct path from all incorrect paths, on the average, we want the branch metric γ to be positive on the correct path and negative on all incorrect paths. Let Z_k be the kth branch symbol of the correct path and let y_k be the kth received symbol. The average value of the metric increment γ_k along the correct path must be positive, and is given by:

$$\begin{aligned} E[\gamma_k] &= E[\alpha - \beta(y_k - Z_k)^2] \\ &= E[\alpha - \beta n_k^2] \\ &= \alpha - \beta\,\sigma^2 \quad \geq 0 \end{aligned} \qquad (16)$$

where σ^2 is the variance of the additive noise.

Now let Z'_k be the kth branch symbol on any incorrect path such that $Z_k - Z'_k = \epsilon_k \neq 0$. The average value of the metric increment γ'_k along Z'_k must be negative, and is given by:

$$\begin{aligned} E[\gamma'_k \mid \epsilon_k = \epsilon\ ,\ \epsilon \neq 0] &= E[\alpha - \beta(y_k - Z'_k)^2 \mid \epsilon \neq 0] \\ &= E[\alpha - \beta(y_k - Z_k + \epsilon)^2 \mid \epsilon \neq 0] \\ &= E[\alpha - \beta(n_k + \epsilon)^2 \mid \epsilon \neq 0] \\ &= \alpha - \beta\ (\sigma^2 + \epsilon^2) \leq 0 \end{aligned} \qquad (17)$$

Thus we have two conditions to satisfy:

$$\begin{cases} \alpha - \beta\ \sigma^2 \quad \geq 0 \\ \alpha - \beta\ (\sigma^2 + \epsilon^2) \leq 0 \end{cases} \qquad (18)$$

Clearly choosing $\alpha = \beta(\sigma^2 + \epsilon'^2)$, where $|\epsilon'| < |\epsilon|$, satisfies these two conditions. Substituting α in (15) yields:

$$\gamma_k = \beta[(\sigma^2+\epsilon'^2) - d_k^2]$$

$$= N_o\beta[0.5+\epsilon'^2 \frac{1}{N_o} - \frac{1}{N_o} d_k^2] \qquad (19)$$

where $\sigma^2 = \frac{N_o}{2}$ is the variance of the additive white Gaussian noise.

The term $N_o\beta$ is only a scaling factor that has no effect on the decoding procedure and can thus be normalized to 1.

The choice of the value of ϵ' in (19) depends on the particular incorrect branch examined. Therefore the metric given by (19) is not practical since the decoder cannot know in advance which branch is being currently examined. However, it is possible to choose $\epsilon'=\epsilon'_{min}$ such that $|\epsilon'_{min}|<|\epsilon_{min}|$, and where ϵ_{min} is the smallest nonzero Euclidian distance between any incorrect branch and the corresponding branch on the correct path. With such a choice of value for ϵ', the conditions (18) are always satisfied. Moreover should ϵ' be too small, then the metric would fall rapidly. As a consequence, for small values of ϵ' and under noisy conditions, the metric drops of the correct path are important. The decoder will therefore explore many incorrect paths before recovering the correct path (see Figure 5). A large and variable computational effort may be thus expected. In order to minimize the computational effort, one must choose relatively larger values of ϵ', and hence accept that conditions (18) may not be satisfied for some of the incorrect branches. However when ϵ' is too large, the metric rises rapidly and drops relatively slowly. As a consequence, the small metric drops of the correct path will induce a smaller computational effort, but under severe noise the incorrect path metrics will also tend to rise quickly, possibly leading to more numerous error events and a degradation of the error performance.

Therefore a trade-off between the error probability and the computational effort is but unavoidable. Examination of this trade-off and determination of the proper range of values to choose ϵ' from has been conducted using computer simulation described next.

IV COMPUTER SIMULATIONS

Several ISI channels and several variants of the sequential decoding stack algorithm have been simulated on a VAX-750 Computer [10]. The purpose of the simulation was to examine for different variants of the M-path algorithm the trade-off relationships between the computational effort and the error performance as parameter ϵ' varies. Computational distribution curves as well as error performance for different signal to noise ratios have also been obtained for different channels and different M-path algorithms.

Three worst case theoretical ISI channels having the following taps have been investigated [20],

- channel 1, memory 4,

$f_0 = 0.29$, $f_1 = 0.50$, $f_2 = 0.58$, $f_3 = 0.50$, $f_4 = 0.29$

- channel 2, memory 6,

$f_0 = 0.19$, $f_1 = 0.35$, $f_2 = 0.46$, $f_3 = 0.50$, $f_4 = 0.46$, $f_5 = 0.35$, $f_6 = 0.19$

- channel 3, memory 9,

$f_0 = 0.12$, $f_1 = 0.23$, $f_2 = 0.32$, $f_3 = 0.39$, $f_4 = 0.42$, $f_5 = 0.42$, $f_6 = 0.39$, $f_7 = 0.32$, $f_8 = 0.23$, $f_9 = 0.12$

The transmitted symbols are equally likely to be +1 or -1. For each simulation, 100,000 symbols have been randomly generated and divided in 200 blocks of 500 symbols each. Each block is terminated by a tail of W known symbols -1, corresponding the an initial channel state (-1,-1,...-1).

The signal to noise ratio is defined as:

$$SNR = \frac{E_s}{N_o} \tag{20}$$

where E_s is the received symbol signal energy and where $\frac{N_o}{2} = \sigma^2$ is the variance of the noise. For convenience E_s has been normalized to 1 so that:

$$SNR = \frac{1}{N_o} \tag{21}$$

The M-path sequential decoding algorithm with M varying from 1 (Z-J algorithm) to M=40 has been used. A flow diagram of the algorithm is shown in Figure 6.

Simulation results

The results concern both the error performance and the computational effort (mean value and distribution) as well as how these performances are influenced by the parameter ϵ'.

Influence of ϵ'

For each of the simulated channels, the error probabilities and the average decoding effort have been obtained for different values of ϵ' as both M and SNR vary. For convenience, results are given as a function of a parameter λ related to ϵ' by:

$$\lambda = \left(\frac{\epsilon'}{f_0}\right)^2 \qquad (22)$$

where f_0 is the first tap value of the channel.

Moreover, in order to facilitate the comparisons between the different algorithms, the average number of computations to decode one symbol has been normalized to indicate only that part, C_{var}, of the average decoding effort due to the computational variability, that is:

$$C_{var} \cong C_{av} - M \qquad (23)$$

where C_{av} is the true average number of computations, and M is the number of paths simultaneously extended by the algorithm. With this definition, clearly, regardless of M, C_{var} is also an indicator of the computational variability since it is equal to zero if there were no computational variability.

For each of the three simulated channels, the error probability $P(\epsilon)$ and C_{var} have been plotted as a function of parameter λ for different values of M. Results are given for two SNR values for each channel and are shown in Figures 7 to 12.

Figures 7,8,9 and 11 show that the error probability monotonely increases and the computational effort decreases with increasing values of λ. However Figures 10 and 12 show that beyond some values of λ and for M=1, the computational effort tends to increases. This phenomenon may be explained as follows: when $\frac{\epsilon'}{N_o}$ becomes large, that is for relatively high signal to noise ratios and/or large values of ϵ', the metric rises rapidly and falls slowly. As a consequence, when the metric of the correct path drops, the decoder may have to follow many incorrect paths over relatively long periods since their metrics also rise rapidly and fall slowly. Therefore, the computational effort may be expected to grow both in average value and variability. Figure 13 illustrates this phenomenon.

Clearly as M gets larger, the correct path becomes extended sooner preventing the examination of a large number of incorrect paths and hence reducing the computational variability.This is well illustrated in Figures 10 and 12.

Moreover as M increases, in all cases, we can notice that both the error probability and the computational effort become relatively insensitive to the parameter λ. Also as M increases, the error probability tends to stabilize towards a minimum value (probably to that of the optimum Viterbi decoder), while the computational variability becomes negligible. The value of M required to stabilize the error probability to its minimum value and also to allow negligible computational variability increases with the channel memory. Furthermore, for each channel, as the SNR increases, the negligible computational variability is reached for a somewhat smaller value of M. However this required value of M, which then becomes pratically equal to C_{av}, is still very small compared to the average number of computations q^W of the Viterbi algorithm. Clearly, in this latter case, since the branch metric does not have to be biased, the error probability is independent of the parameter λ, and C_{var} is strictly zero.

Error Performances and Computational Effort

In performing the simulations to obtain the error performance and computational effort as a function of the SNR, parameter ϵ' was chosen as to minimize the error probability. Whenever the error probability is insensitive to ϵ', then ϵ' is chosen to minimize C_{var}.

The M-path sequential decoding performances are compared, for channel 1 and 2, to both Viterbi decoding performance* and the theoretical performance bounds given by (13). For channel 3 (memory 9), because of the considerable amount of computations required by the Viterbi algorithm, only the theoretical performance bounds are considered. The results are given in Figures 14 to 16.

Figures 14 and 15 show that when $M \geq 3$, the error performances of the M-path sequential decoding approches considerably the error performance of Viterbi decoding. As for the computational effort, at $P(\epsilon) \leq 10^{-4}$, for channel 1,

* The Viterbi error performance curves have been obtained by simulations using the simplified M-path algorithm that will be described in the next section with $M = q^{W}$

a 3-path sequential decoder requires less than 20% of the computational effort of the optimum Viterbi decoder (see Figure 17), whereas for channel 2, a 3-path sequential decoder requires an even smaller fraction, a mere 5% of the computational effort of the Viterbi decoder (see Figure 18).

For channel 3, Figure 16 shows that the error performance very slightly overshoots the theoretical upper bound at the relatively small values of SNR. However for $M \geq 10$ the improvement of the error probability becomes negligible, suggesting that the best performance (i.e. Viterbi decoding) is pratically reached. But when comparing the average decoding efforts, a Viterbi decoder requires $2^9=512$ computations per decoded symbol, whereas the 10-path algorithm has required approximatively 10 computations per decoded symbol, a mere 2% of Viterbi decoding (see Figure 21).

The observed distributions of the number of computations per decoded symbol are shown in Figures 17 to 22. As expected, the variability always decreases as M or SNR increases. For an error performance $P(\epsilon)<10^{-4}$, this variability is shown to become negligible for very modest values of M, of the same order of magnitude as the channel length.

As a consequence of the absence of any computational variability, a considerably simplified decoder structure can be considered. Such a decoder extends M paths simultaneously as the ordinary M-path sequential decoder. However, this simplified decoder never backs-up in the tree, and thus eliminates the main drawback of sequential decoding, that is the computational variability, while preserving both its good error performance and its small average decoding effort. This decoder called "simplified multiple path sequential decoder" is discussed and evaluated in the next section.

V Simplified multiple path sequential decoder

The simplified M-path algorithm is a tree search algorithm that retains at any tree depth only the M-most likely paths. At each decoding step, each of the M remaining or "surviving" paths is extended into its q single branch extensions. The metrics are then computed and from these qM resultant paths only the M paths with the largest metrics are retained. All the other paths are discarded. The extended paths are all at the same tree depths and there is no backing-up in the tree. The average number of computations per decoded symbol is thus constant and equal to M, and the computational variability is completely eliminated. Furthermore in order to insure that all surviving paths have distinct channel states, remergers are exploited as in the Viterbi decoding algorithm. The simplified M-path algorithm is easily implemented using a stack structure, and is another particular case of the Generalized Stack Algorithm [12]. Clearly, whenever M is equal to the total number of states q^W, this simplified algorithm becomes equivalent to the Viterbi Algorithm.

Furthermore, as for Viterbi decoding, all the "surviving" paths do not stay distinct over their entire length, but have a tendency to stem from a single node several branches earlier. Hence, as illustrated in Figure 23, it is not necessary to wait until the unique decoded path has been obtained before starting to deliver the decoded symbols. After a delay of δ branches (or δ symbols), the decoder may deliver, at each decoding step, the oldest symbol from any one of the "surviving" paths. However, in order to minimize the error events, it may be preferable to deliver the oldest symbol of the surviving path having the highest metric since this path is currently the most likely. Therefore, with this procedure, the decoder needs to store the path history of the surviving paths over only the past δ symbols. M registers each δ symbols long are used for this purpose. As shown in Figure 24, at each decoding step, the M surviving paths stored in M registers are extended. From the qM resulting paths the M new surviving paths are selected and stored in M other registers with the oldest symbol of the surviving path having the highest metric delivered to the user. The value of the path history length δ that should be retained for each channel and for each algorithm,has been estimated using computer simulations. The results are given next.

Path history length δ

For each of the three simulated channel, different values of δ have been tested. A simulation consists of 10,000 symbols chosen to be equaly likely +1 or -1. At each simulation we count the number of times NC where the M surviving paths do not stem from a single node δ branches earlier. The M surviving paths are issued from a single node δ branches earlier if at that depth they all have the same identical state. The number NE of symbols decoded in error are also counted. The results obtained from these simulations are given in Tables 1,2 and 3, where each Table corresponds to one of the three simulated channels.

We notice from Tables 1,2 and 3 that the number NC tends to zero and NE stabilizes to some values as the value of δ increases. Clearly, regardless of M, one should select that value of δ which corresponds to NC=0. Hence, from Tables 1,2, and 3 we can see that the value of δ that should be chosen gets larger as both M and the channel memory increase. An upper bound on δ is however readily provided by the corresponding path history length of

Viterbi decoding (the so-called memory of the decoder), since then all the possible distinct paths rather than M paths are extended by the decoder. In all cases it appears that δ should be equal to 4 to 6 times the memory of the channel.

Error Performance of the simplified algorithm

This simplified sequential decoder has been simulated in order to compare its performance to those of both sequential decoding and Viterbi decoding. A simulation run consists of 100,000 symbols divided in 10 blocks of 10,000 symbols each, and δ is chosen to be equal to 6 times the length of the simulated channel. The obtained results are given in Figures 25, 26 and 27.

Figures 25, 26 and 27 show that as expected the performances of the simplified M-path sequential decoding are very close to those of both Viterbi and sequential decoding. As for the computational effort, the simplified decoder requires only a very modest number M of computations per decoded symbol, with M of the same order of magnitude as the channel length, $(M \approx W+1)$, whereas the number of computations required by the Viterbi decoder is q^W. Naturally neither algorithm suffers from any computational variability.

Table 4 compares the average number of computations per decoded symbols and the error probability of the

Viterbi, multiple-path and simplified decoders. Table 4 shows that the M-path sequential decoder can achieve the error performance of a Viterbi decoder but at a very small fraction of the average decoding effort. Compared to the sequential decoders the simplified M-paht decoders show a very slight error performance degradation but as shown in Table 4 this degradation may be circumvented by a small increase of the number of extended paths M. However this slight increase of M presents a minimal practical consequences on the complexity of the decoder and is more than compensated by the total absence of any computational variability.

Finally from a practical point view the actual metric computation could be simplified and speeded-up if an integer metric were used instead of the real number metric defined by (11). Naturally the harder the metric quantization, the larger the error performance degradation. Using computer simulation it has been observed that by quantizing the entire range of exact metric values into 10 to 50 levels (i.e. less than 6 bits) the ensuing degradation is but insignifiant, as illustrated in Figures 28 to 30. With this metric quantization the squaring operation in (19) is eleminated and hence the decoder may be very easily implemented using simple logic circuits and small memory storage. This simplified sequential decoder may therefore become a very attractive alternative to Viterbi decoding to combat ISI over relatively long memory channels.

VI <u>CONCLUSION</u>

In this paper we have presented the application of sequential decoding, in particular the multiple-path variant of the stack algorithm, to the intersymbol interference problem. The likelihood function or metric to use in ISI has been investigated. This metric departs somewhat from the traditional Fano metric of sequential decoding but maintains the usual overall dynamic behaviour that helps the decoder discriminate the correct path from all incorrect paths. Using computer simulation we have examined the effect of this metric on both the error performance and computational variability as the number of extended paths M varies.

From extensive computer simulation with channels of memory 4, 6 and 9, we show that the error performance of a multiple-path sequential decoder falls within the theoretical upper and lower bound of an optimal decoder, but at an average decoding effort of only a very small fraction of that of an optimal decoder. As for the computational variability it can be practically eliminated by properly choosing the number of simultaneously extended paths M.

Finally we have presented a simplified multiple-path sequential decoder that offers a constant (and small) average computational effort with no computational variability. This computational effort is of the same order of magnitude as the channel memory, whereas for Viterbi decoding it is exponentially increasing with the channel memory. In addition both decoders allow a real-time operation with the same constant delay. Compared to either Viterbi or sequential decoders the simplified decoder provides an error performance only very slightly degraded, but its structure and storage memory requirements are considerably simpler and smaller than either decoders. By its simplicity and good performances this decoder appears an extremely attractive alternative to Viterbi decoding for ISI over long memory channels. Moreover it may also find applications in other problems of trellis decoding, as for example in multilevel phase signals decoding [21].

REFERENCES

[1] R.W. Lucky, J. Salz and E.J. Weldon, "Principles of Data Communications", McGraw Hill, New York, 1968, Chap.6.

[2] J.G. Proakis, "Digital Communications", McGraw Hill, New York, 1983, Chap.6.

[3] J.G. Proakis, "Advances in Equalization for Intersymbol Interference", in "Communication Systems and Random Processe Theory" J. Skwirzynski ed., NATO Advanced Studies Institute Series, Darlington, U.K., Aug. 1978.

[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] F.R. Magee and J.G. Proakis, "Adaptive Maximum Likelihood Sequence Estimation for Digital Signaling in the Presence of Intersymbol Interference", IEEE Trans. on Inform. Theory, Vol. IT-19, pp. 120-124, Jan. 1973.

[6] J.F. Hayes, "The Viterbi Algorithm Applied to Digital Data Transmission", IEEE Comm. Magazine, Vol. 13, pp. 15-20, March 1975.

[7] G. Ungerboeck, "Adaptive Maximum Likelihood Receiver for Carrier-Modulated Data Transmission Systems", IEEE Trans. Commun., Vol. COM-22, pp. 624-636, May 1974.

[8] S. Crozier, H. Wilson, W. Moreland, J. Camelon, P. McLane, "Micro Processor Based Implementation and Testing of a Simple Viterbi Detector", Canadian Electr. Engin. Joun., Vol. 6, pp.3-8, Jan. 1981.

[9] D. Haccoun, P. Haurie, "Application of Multiple-Path Sequential Decoding to the Intersymbol Interference Problem", Book of Abstracts, IEEE Intern. Symp. on Inform. Theory, Les Arcs, France, June 1982.

[10] S. Kallel, "Résolution de l'interférence entre symboles par décodage séquentiel", M.Sc.A. Thesis, Depart. Elect. Eng., Ecole Polytechnique de Montréal, Montreal, Canada, Dec. 1984.

[11] S. Kallel, D. Haccoun, "Multiple Path Sequential Decoding for the Intersymbol Interference Problem", Twelfth Biennial Symposium on Communications,

Kingston, Ontario, Canada, June 1984.

[12] D. Haccoun and M.J. Ferguson, "Generalized Stack Algorithms for the Decoding of Convolutional Codes", IEEE Trans. Inform. Theory, Vol. IT-21, pp. 638-651, Nov. 1975.

[13] A.J. Viterbi, "Convolutional Codes and their Performance in Communication Systems", IEEE Trans. on Commun, Tech., Vol. COM-19, pp.751-772, Oct. 1971.

[14] G.D. Forney, "The Viterbi Algorithm", Proc. IEEE, Vol. 61 pp. 268-178, March 1973.

[15] V.K. Bhargava, D. Haccoun, R. Matyas, P. Nuspl, "Digital Communications by Satellite", John Wiley, New York, 1981.

[16] R.M. Fano, "A Heuristic Discussion of Probabilistic Decoding", IEEE Trans. Inform. Theory, Vol. IT-9, pp. 64-74, Apr. 1963.

[17] J.L. Massey, "Variable-Length Codes and the Fano Metric", IEEE Trans. Inf. Theory, Vol. IT-18, pp.196-198, Jan. 1972.

[18] I.M. Jacobs and E.R. Berlekamp", A Lower Bound to the Distribution of Computation for Sequential Decoding", IEEE Trans. Inform. Theory, Vol. IT-13, pp. 167-174, Apr. 1967.

[19] F. Jelinek, "A Fast Sequential Decoding Algorithm Using a Stack", IBM Joun. Res. Develop., Vol. 13, pp. 675-685, Nov. 1969.

[20] F.R. Magee and J.G. Proakis, "An Estimate of the Upper Bound on Error Probability for Maximum Likelihood Estimation on Channels having a Finite Duration Pulse Response", IEEE Trans. Inf. Theory, Vol. IT-19, pp. 699-702, Sept. 1973.

[21] G. Ungerboeck, "Channel Coding With Multilevel/Phase Signals", IEEE Trans. on Inform. Theory, Vol. IT-28, pp. 55-67, Jan. 1982.

LIST OF FIGURES

Figure 1: Discrete channel model.

Figure 2: Binary tree corresponding to the channel: $f_0=0.5$, $f_1=1.0$, $f_2=0.5$.

Figure 3: Binary trellis corresponding to the channel: $f_0=0.5$, $f_1=0.5$, $f_2=0.5$.

Figure 4: Block diagram of a stack algorithm.

Figure 5: Influence of a small ϵ' value on the path metric behaviour.

Figure 6: Flow chart of the simulated M-path algorithm.

Figure 7: Influence of $\lambda = (\frac{\epsilon'}{f_0})^2$ on P_e and C_{var}, channel 1, $E_s/N_o = 11$ dB

Figure 8: Influence of $\lambda = (\frac{\epsilon'}{f_0})^2$ on P_e and C_{var}, channel 1, $E_s/N_o = 13$ dB

Figure 9: Influence of $\lambda = (\frac{\epsilon'}{f_0})^2$ on P_e and C_{var} ,channel 2, $E_s/N_o = 15$ dB

Figure 10: Influence of $\lambda = (\frac{\epsilon'}{f_0})^2$ on P_e and C_{var} ,channel 2, $E_s/N_o = 17$ dB

Figure 11: Influence of $\lambda = (\frac{\epsilon'}{f_0})^2$ on P_e and C_{var} ,channel 3, $E_s/N_o = 17$ dB

Figure 12: Influence of $\lambda = (\frac{\epsilon'}{f_0})^2$ on P_e and C_{var} ,channel 3, E_s/N_o = 19 dB

Figure 13: Effect of a too large ϵ' value on the path metric behaviour.

Figure 14: Comparisons of the observed Error Performances for the M-Path and Viterbi decoders (channel 1).

Figure 15: Comparisons of the observed Error Performances for the M-Path and Viterbi decoders (channel 2).

Figure 16: Comparisons of the observed Error Performances for the M-Path decoders (channel 3).

Figure 17: Computational distributions for M=1,3 and 4 paths, channel 1, E_s/N_o=15 dB.

Figure 18: Computational distributions for M=1 and 3 paths, channel 1, E_s/N_o=17 dB.

Figure 19: Computational distributions for M=1,3 and 5 paths, channel 2, E_s/N_o=17 dB.

Figure 20: Computational distributions for M=1 and 3 paths, channel 2, E_s/N_o=19 dB.

Figure 21: Computational distributions for M=1, 3, 10 and 20 paths, channel 3, E_s/N_o=19 dB.

Figure 22: Computational distributions for M=1, 3, 5 and 10 paths, channel 3, E_s/N_o=21 dB.

Figure 23: Illustration of the tree exploration of the 3-path simplified algorithm.

Figure 24: Conceptual structure of the simplified sequential decoder.

igure 25: Comparisons of the observed error Performances of the M-Path, Simplified and Viterbi decoders (channel 1).

Figure 26: Comparisons of the observed error Performances of the M-Path, Simplified and Viterbi decoders (channel 2).

Figure 27: Comparisons of the observed error Performances of the M-Path, Simplified and Viterbi decoders (channel 3).

Figure 28: Effect of the metric quantization on the error performance of the simplified M-path decoder (channel 1).

Figure 29: Effect of the metric quantization on the error performance of the simplified M-path decoder (channel 2).

Figure 30: Effect of the metric quantization on the error performance of :e simplified M-path decoder (channel 3).

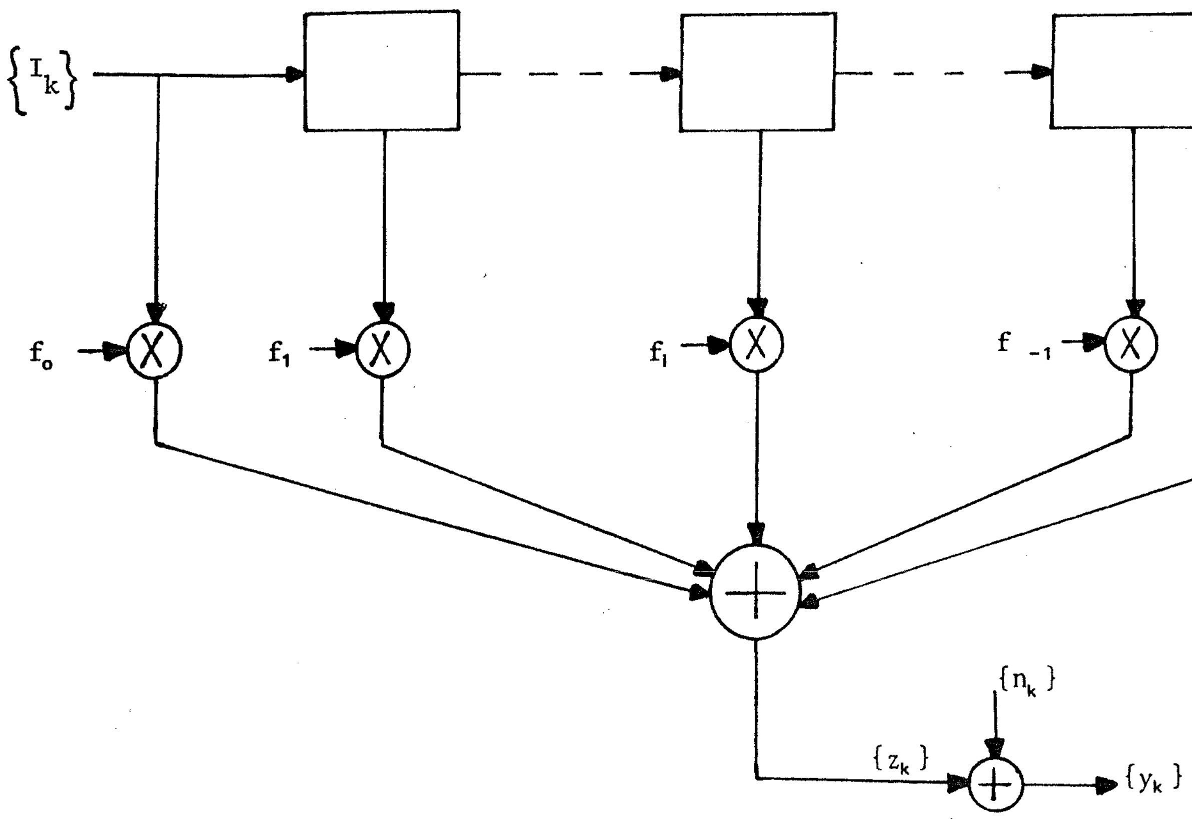


Figure 1: Discrete channel model

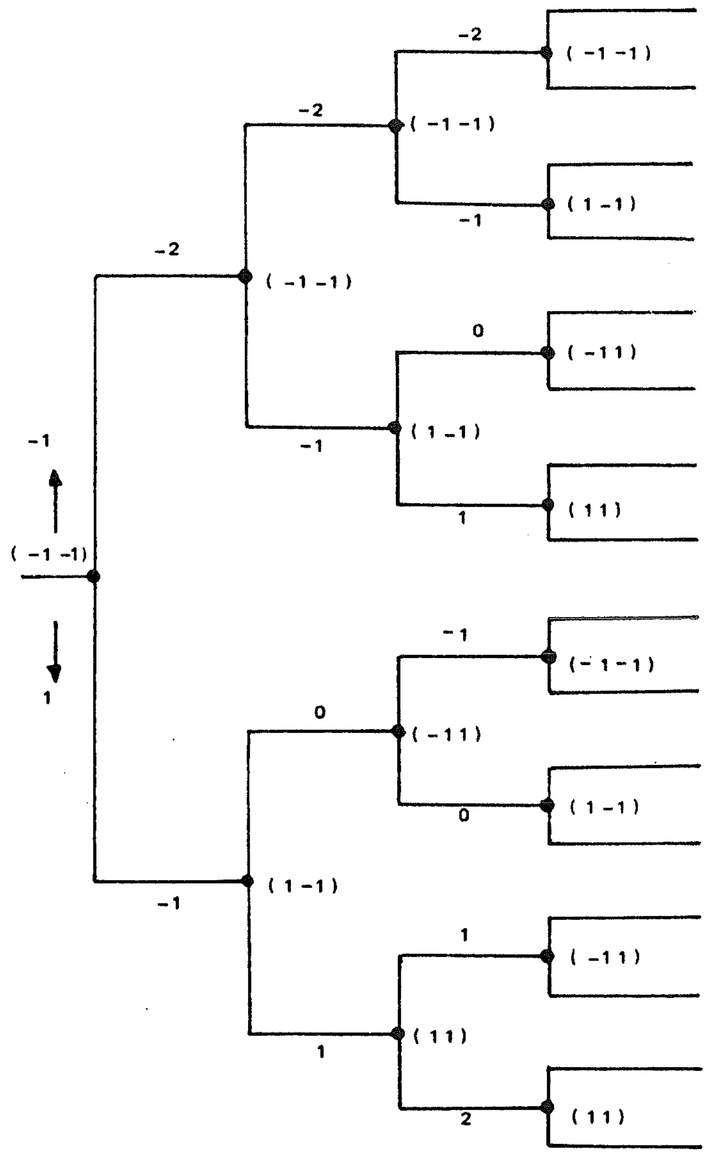


Figure 2: Binary tree corresponding to the channel: $f_o = 0.5$, $f_1 = 1.0$, $f_2 = 0.5$

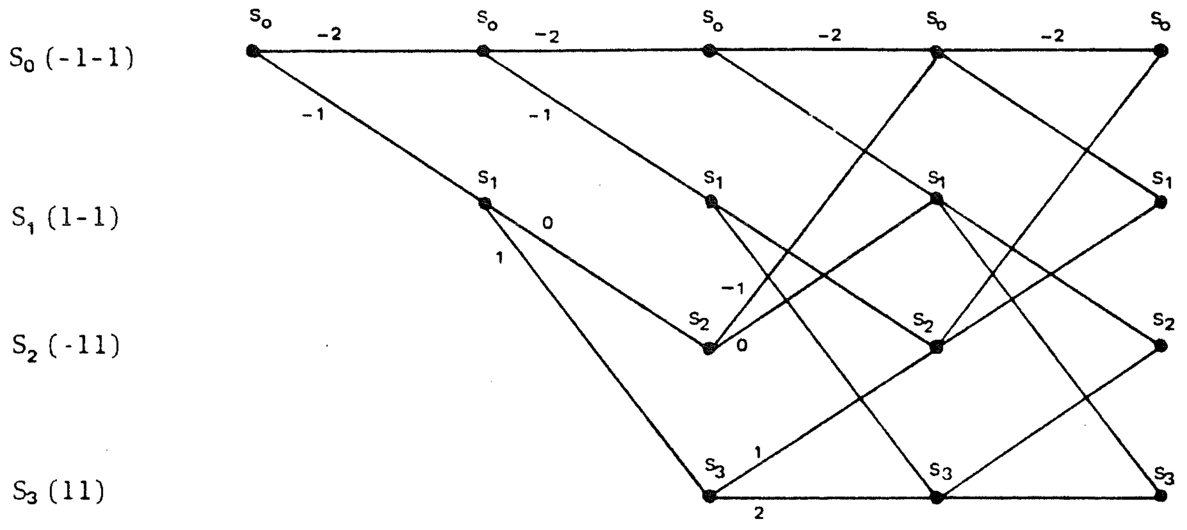


FIGURE 3: Binary trellis corresponding to the channel: $f_0 = 0.5$, $f_1 = 1.0$, $f_2 = 0.5$

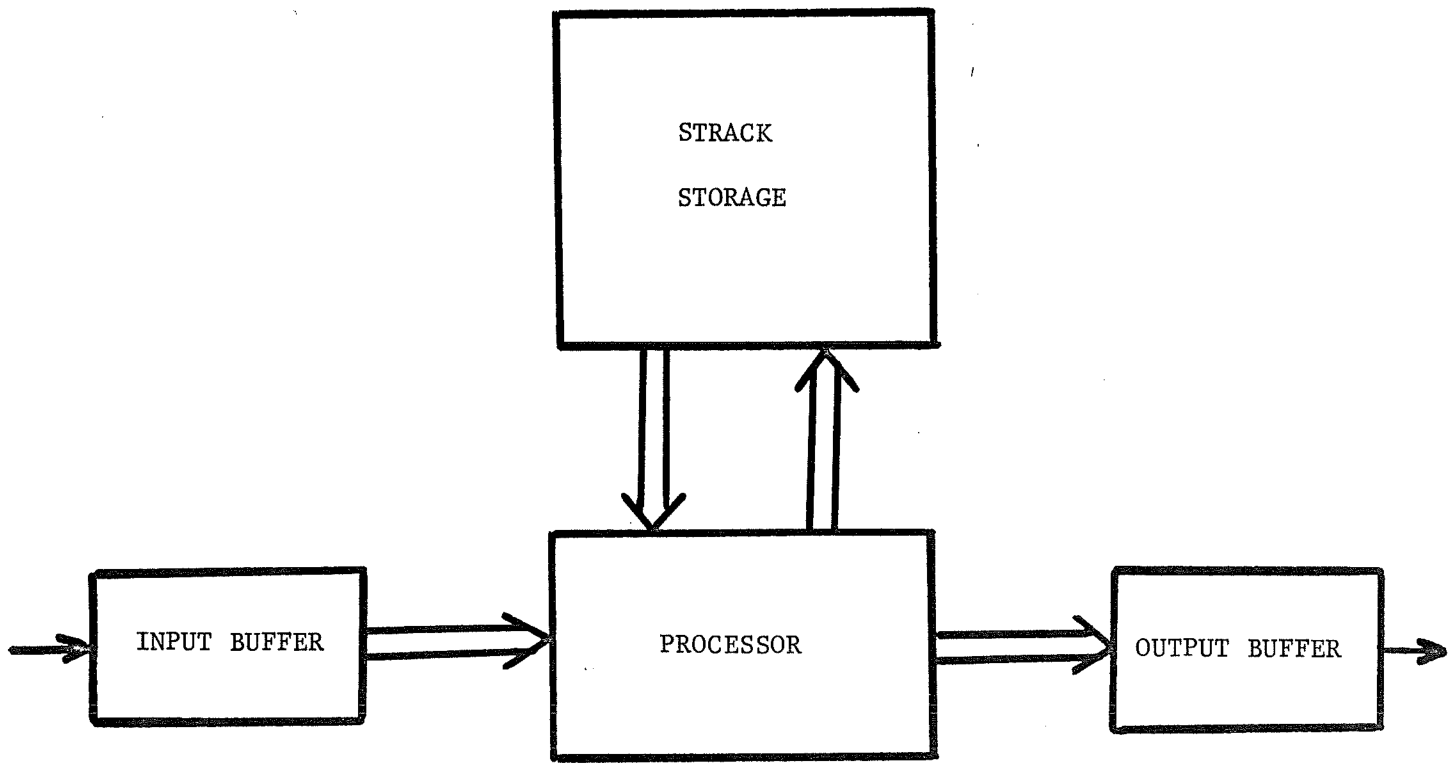


Figure 4: Block diagram of a stack algorithm

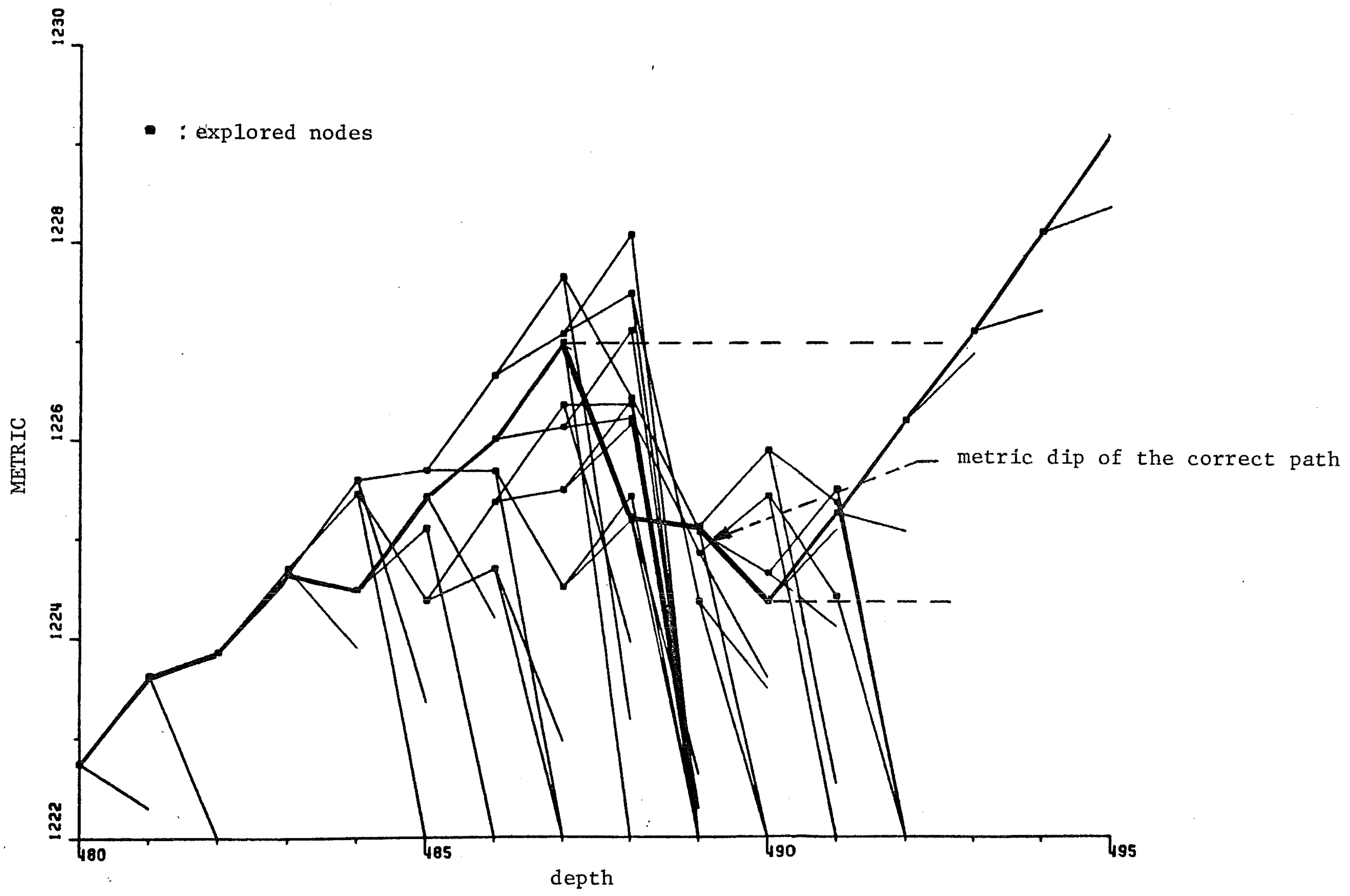


Figure 5: Influence of a small ε value on the path metric behaviour

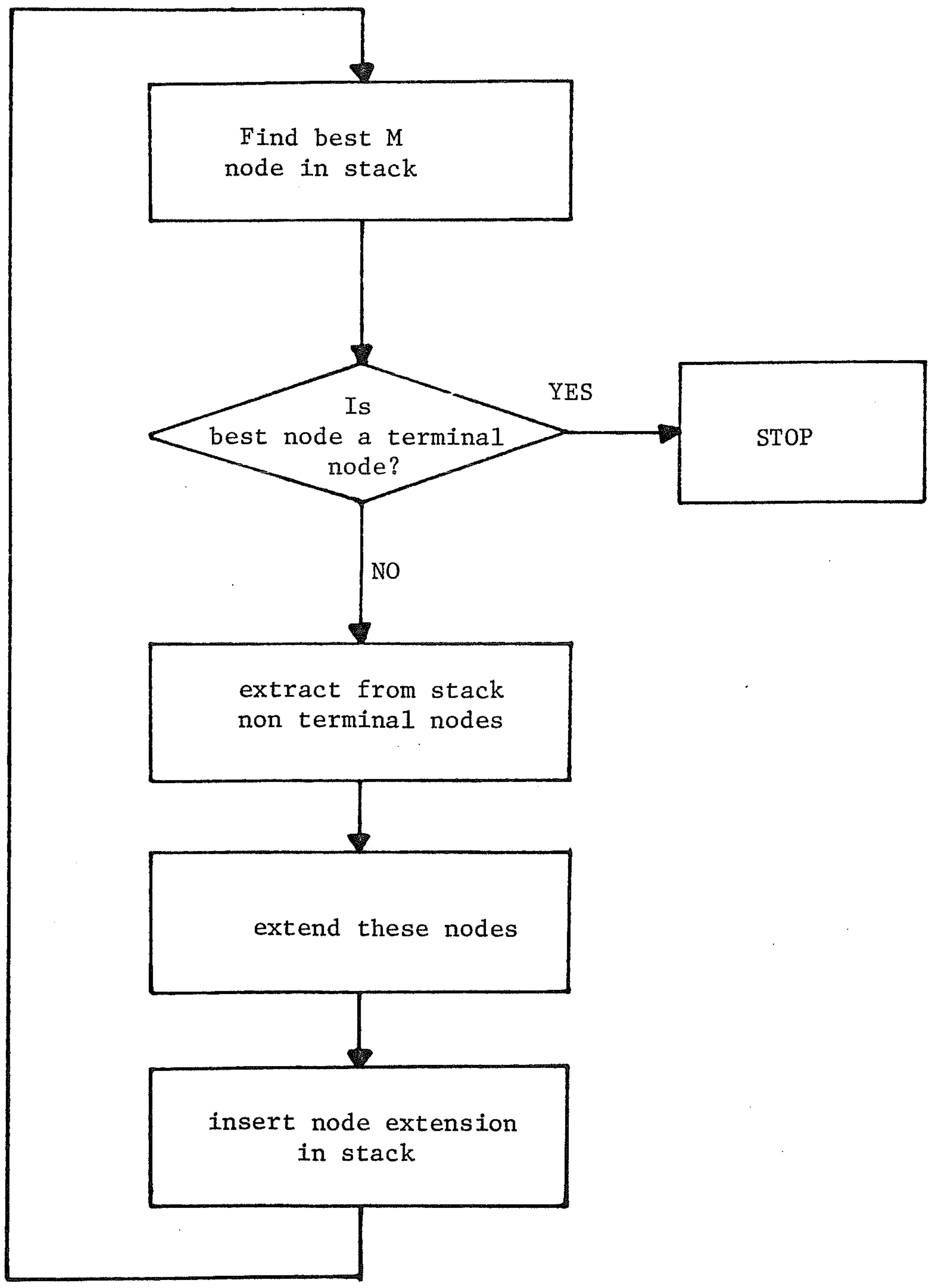
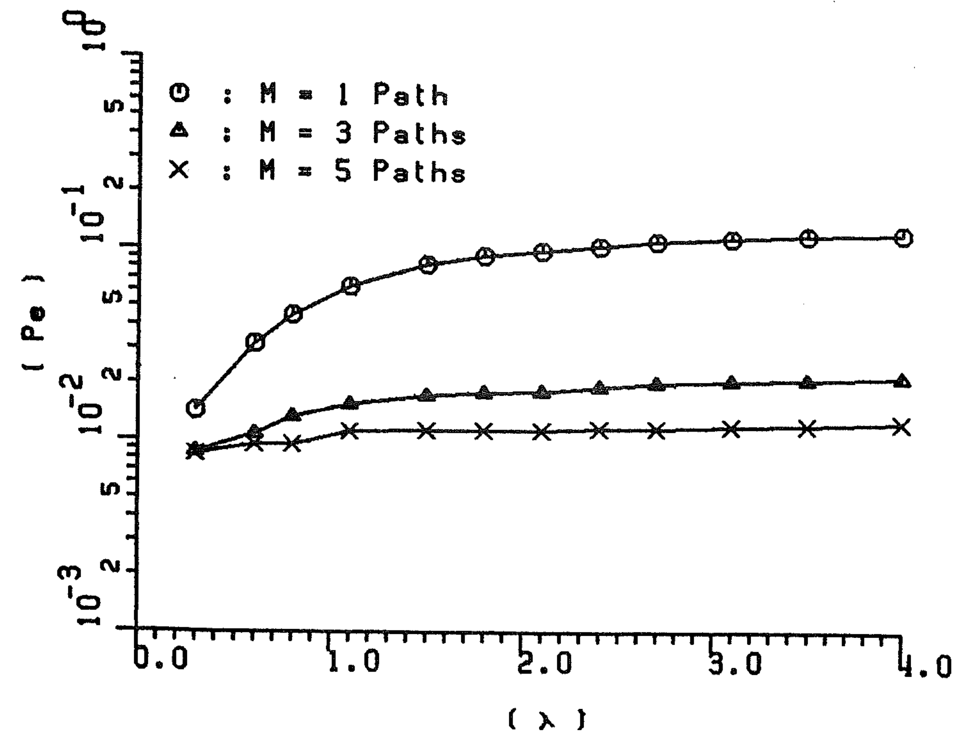

Find best M node in stack
Is best node a terminal node?
YES
STOP
NO
extract from stack non terminal nodes
extend these nodes
insert node extension in stack

Figure 6: Flow chart of the simulated M-path algorithm



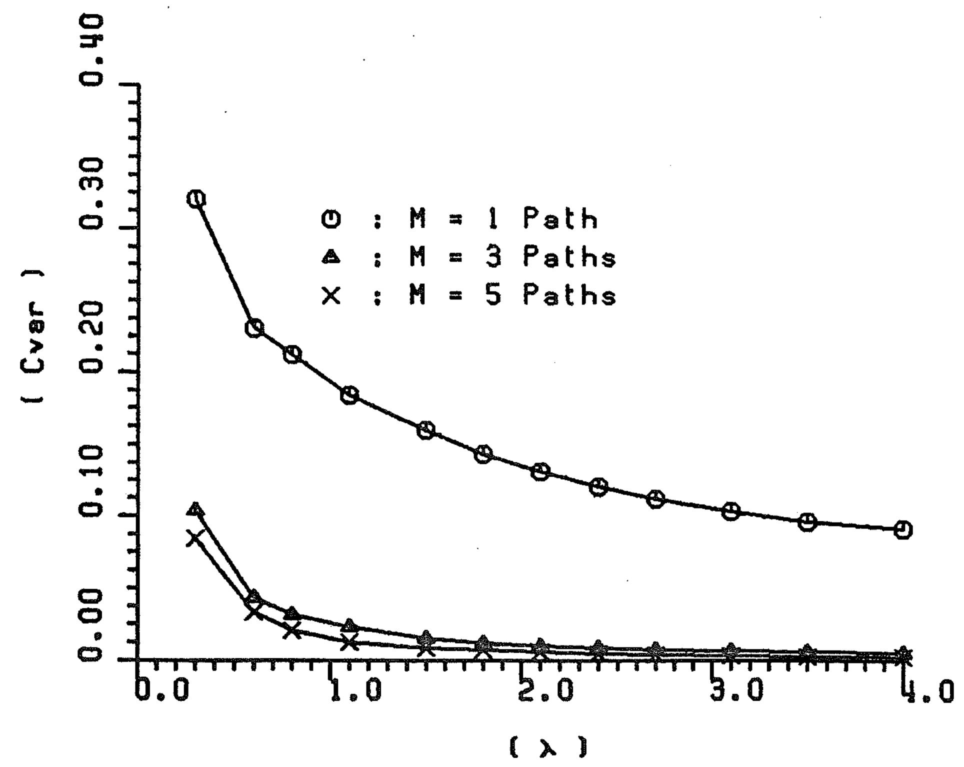


Figure 7: Influence of $\lambda = \left(\frac{\varepsilon'}{f_o}\right)^2$ on P_e and C_{var},
Channel 1, E_s/N_o = 11 dB

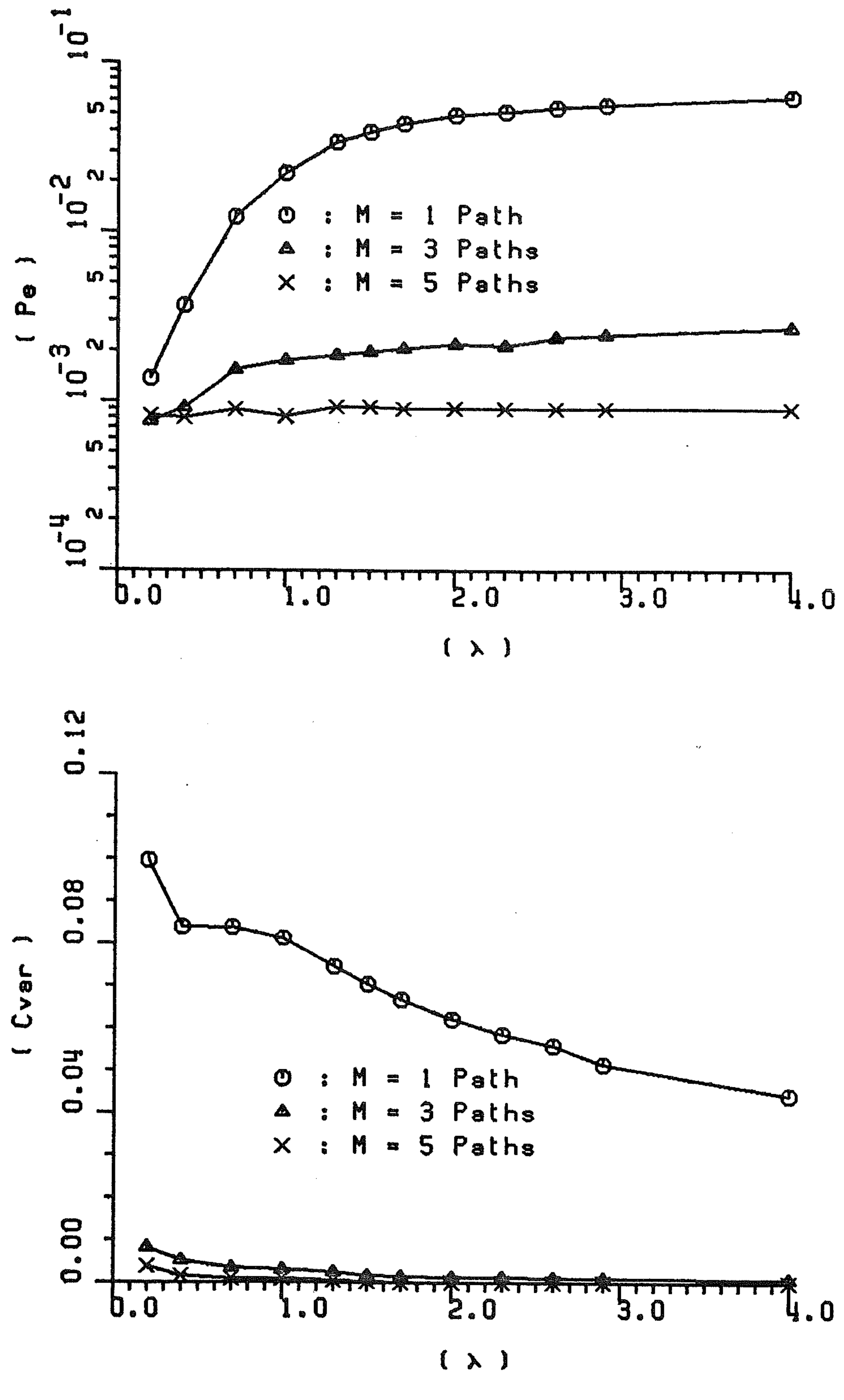
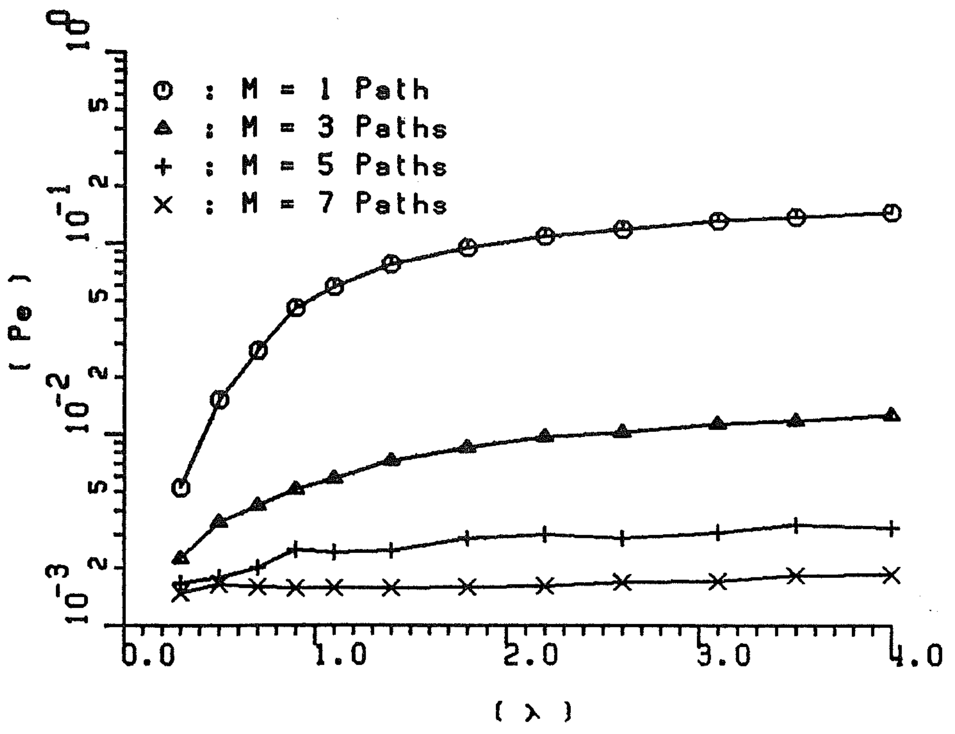


Figure 8: Influence of $\lambda = \left(\frac{\varepsilon'}{f_o}\right)^2$ on P_e and C_{var},

Channel 1, E_s/N_o = 13 dB



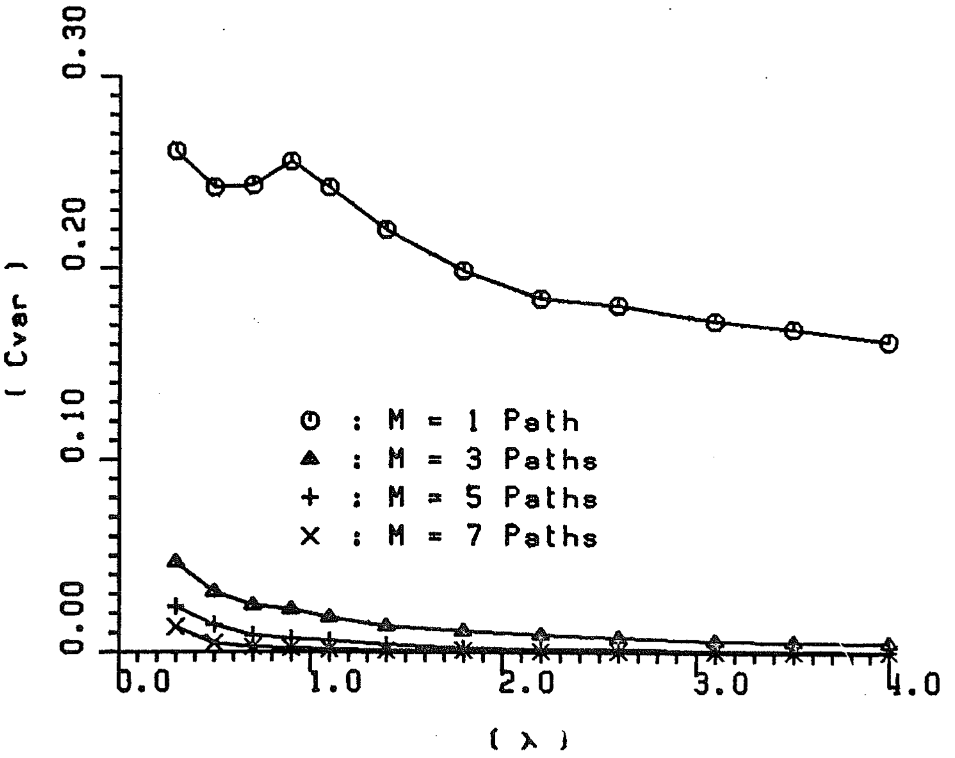
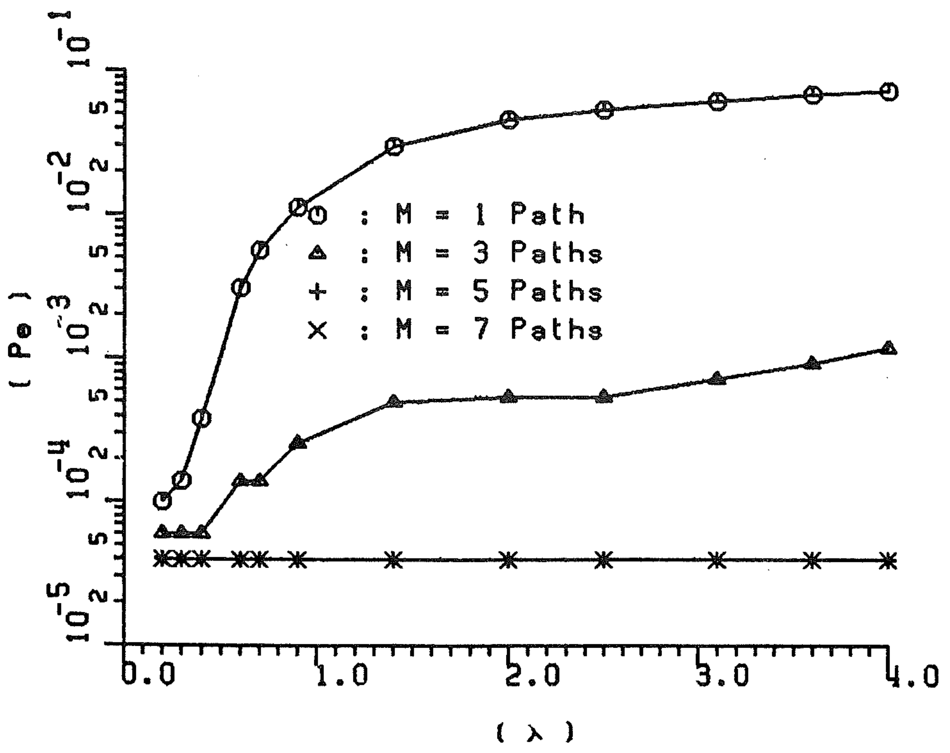


Figure 9: Influence of $\lambda = \left(\frac{\varepsilon'}{f_o}\right)^2$ on P_e and C_{var},

Channel 2, E_s/N_o = 15 dB



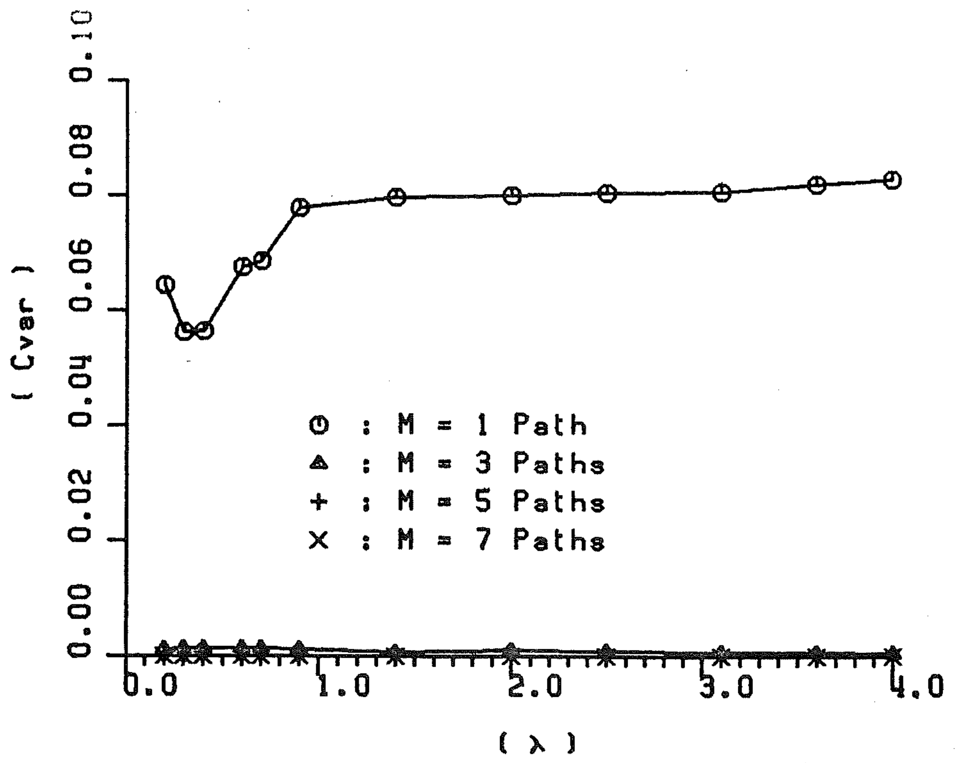
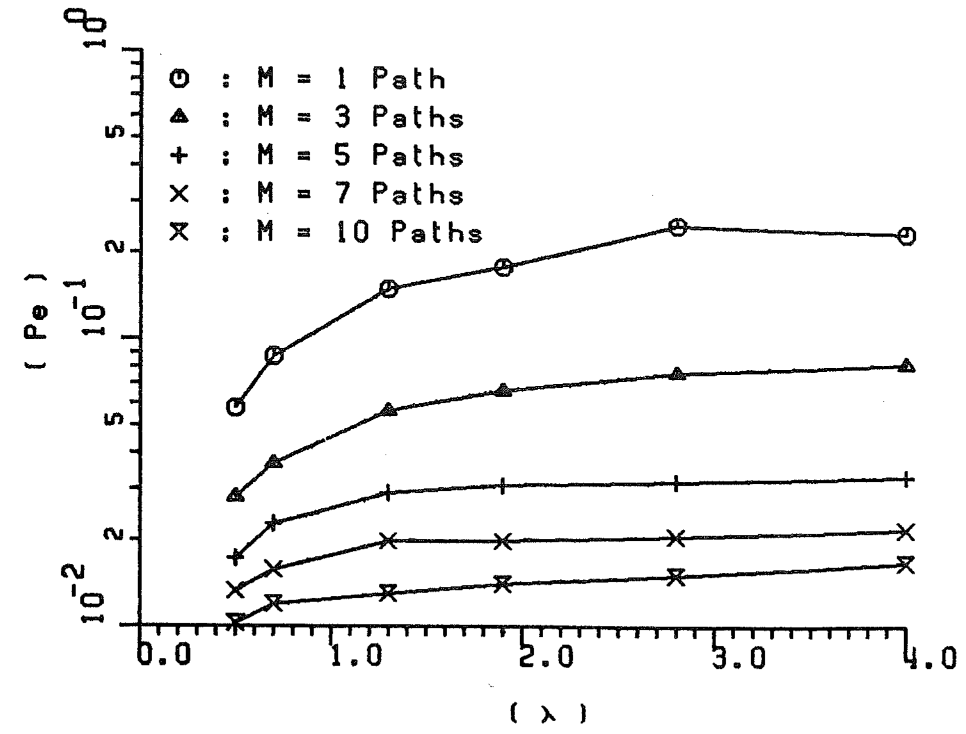


Figure 10: Influence of $\lambda = \left(\frac{\varepsilon'}{f_o}\right)^2$ on P_e and C_{var},

Channel 2, E_s/N_o = 17 dB



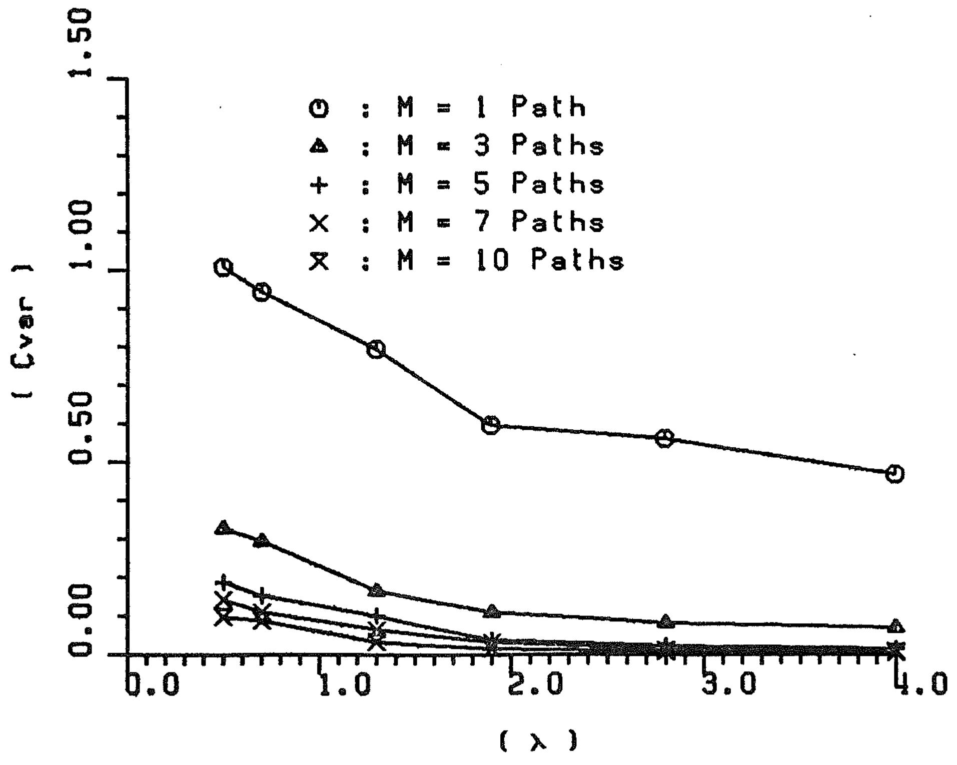
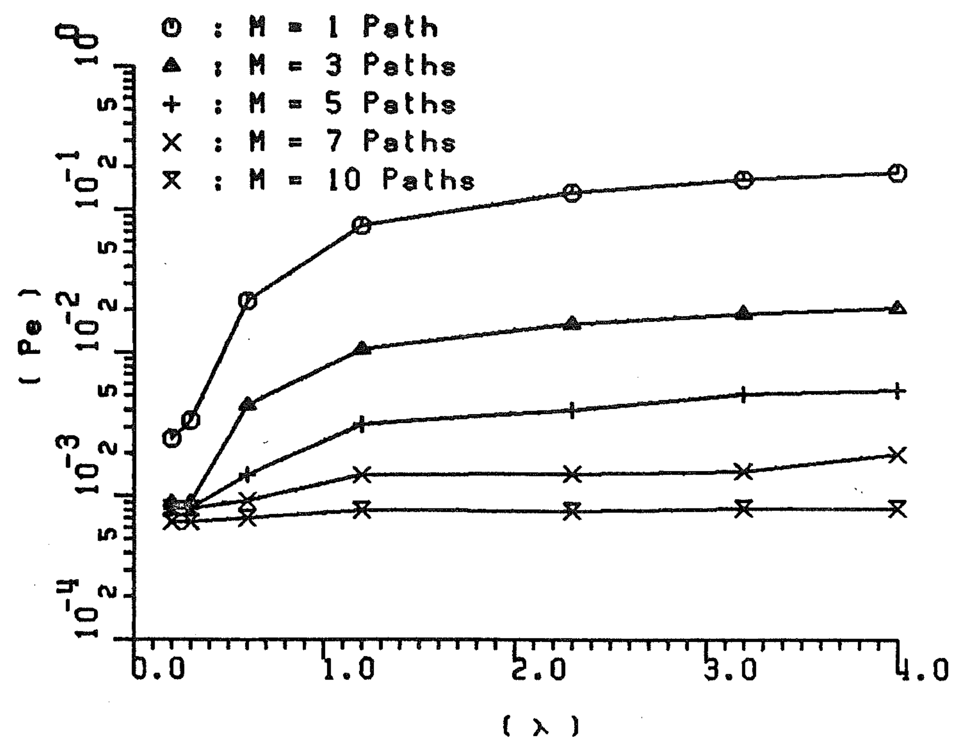


Figure 11: Influence of $\lambda = \left(\frac{\varepsilon'}{f_o}\right)^2$ on P_e and C_{var},

Channel 3, E_s/N_o = 17 dB



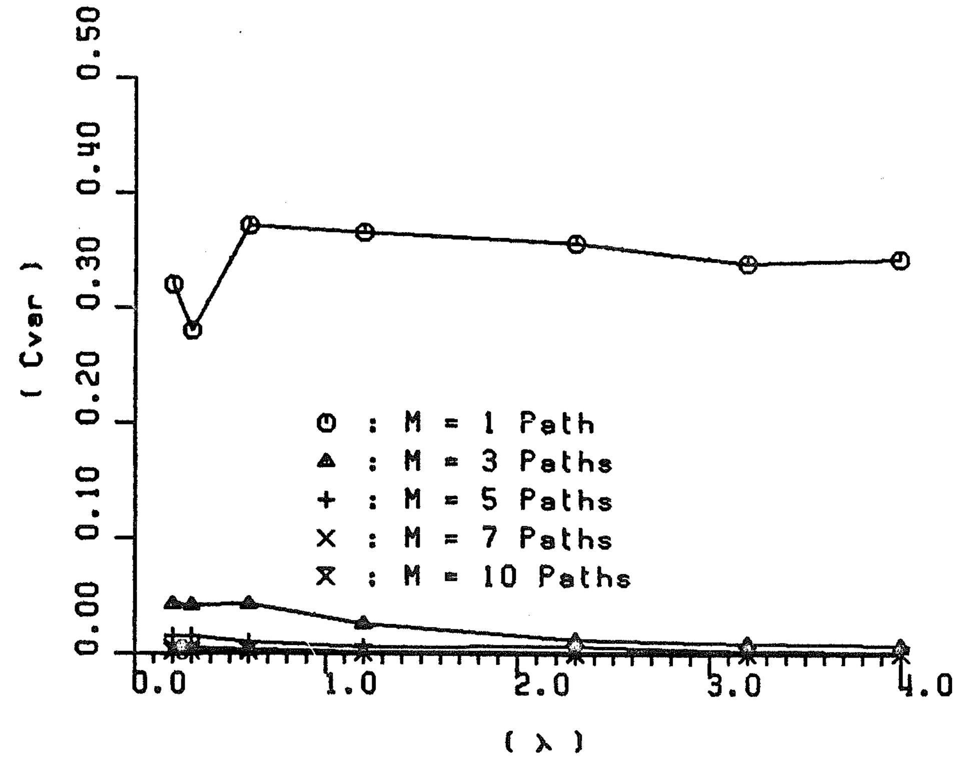


Figure 12: Influence of $\lambda = \left(\frac{\varepsilon'}{f_o}\right)^2$ on P_e and C_{var},

Channel 3, E_s/N_o = 19 dB

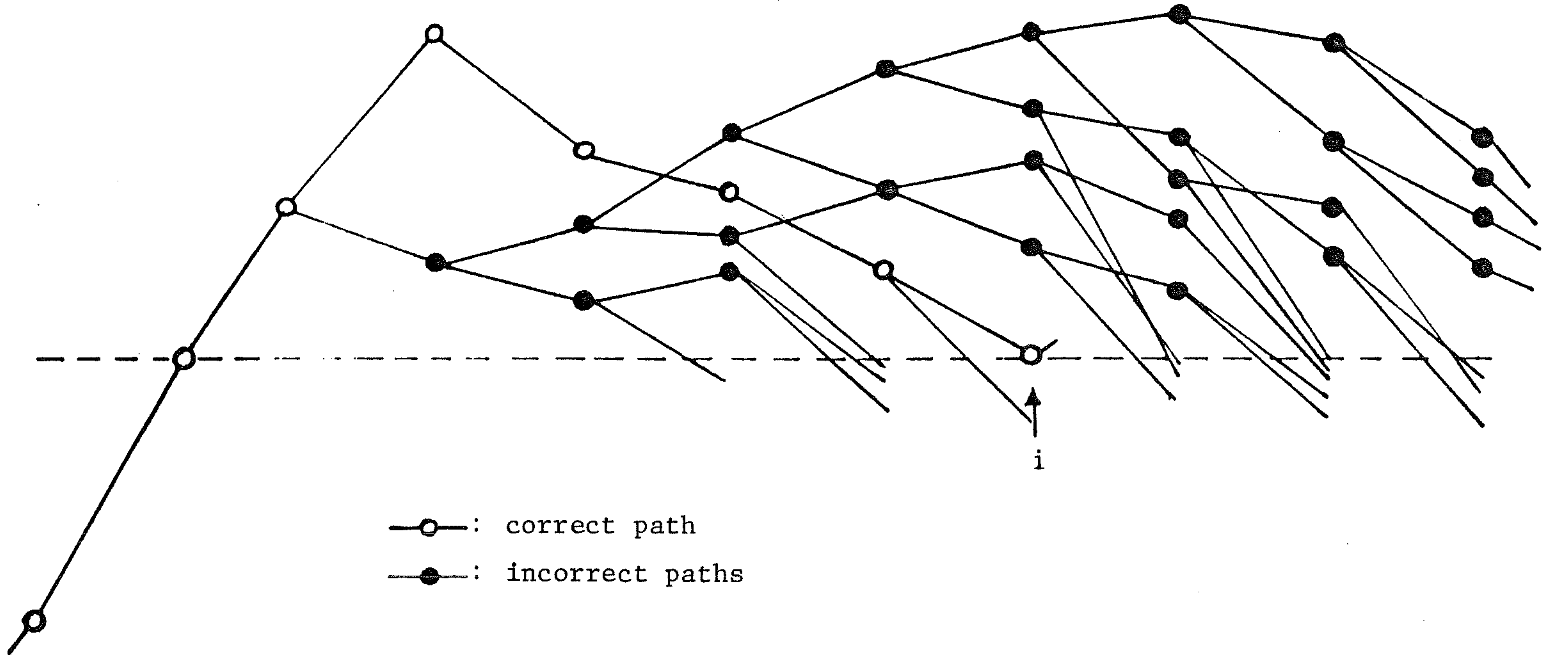


Figure 13: Effect of too large ε' value on the path metric behaviour

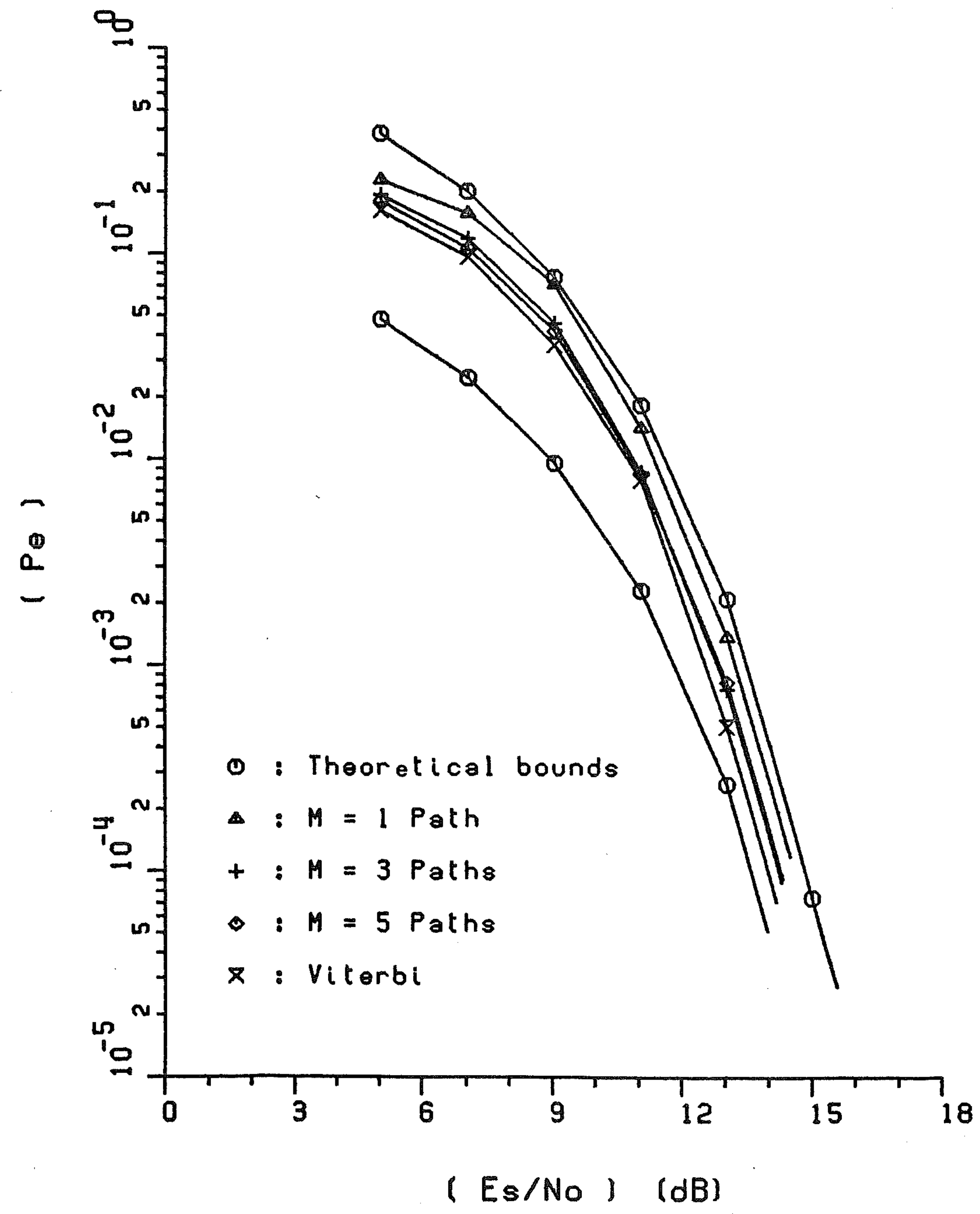


Figure 14: Comparisons of the observed Error Performances for the M-Path and Viterbi decoders (channel 1)

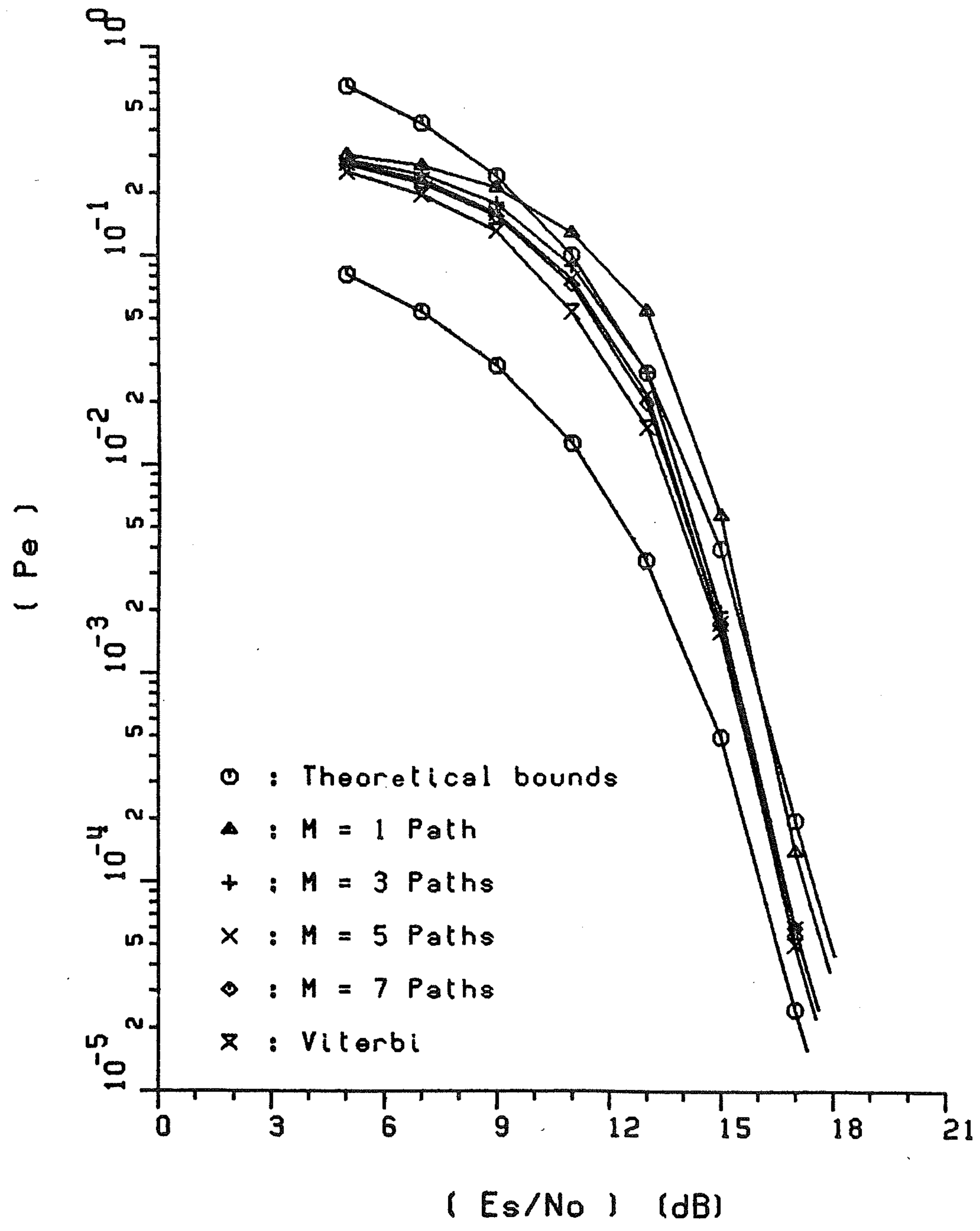


Figure 15: Comparisons of the observed Error Performances for the M-Path and Viterbi decoders (channel 2)

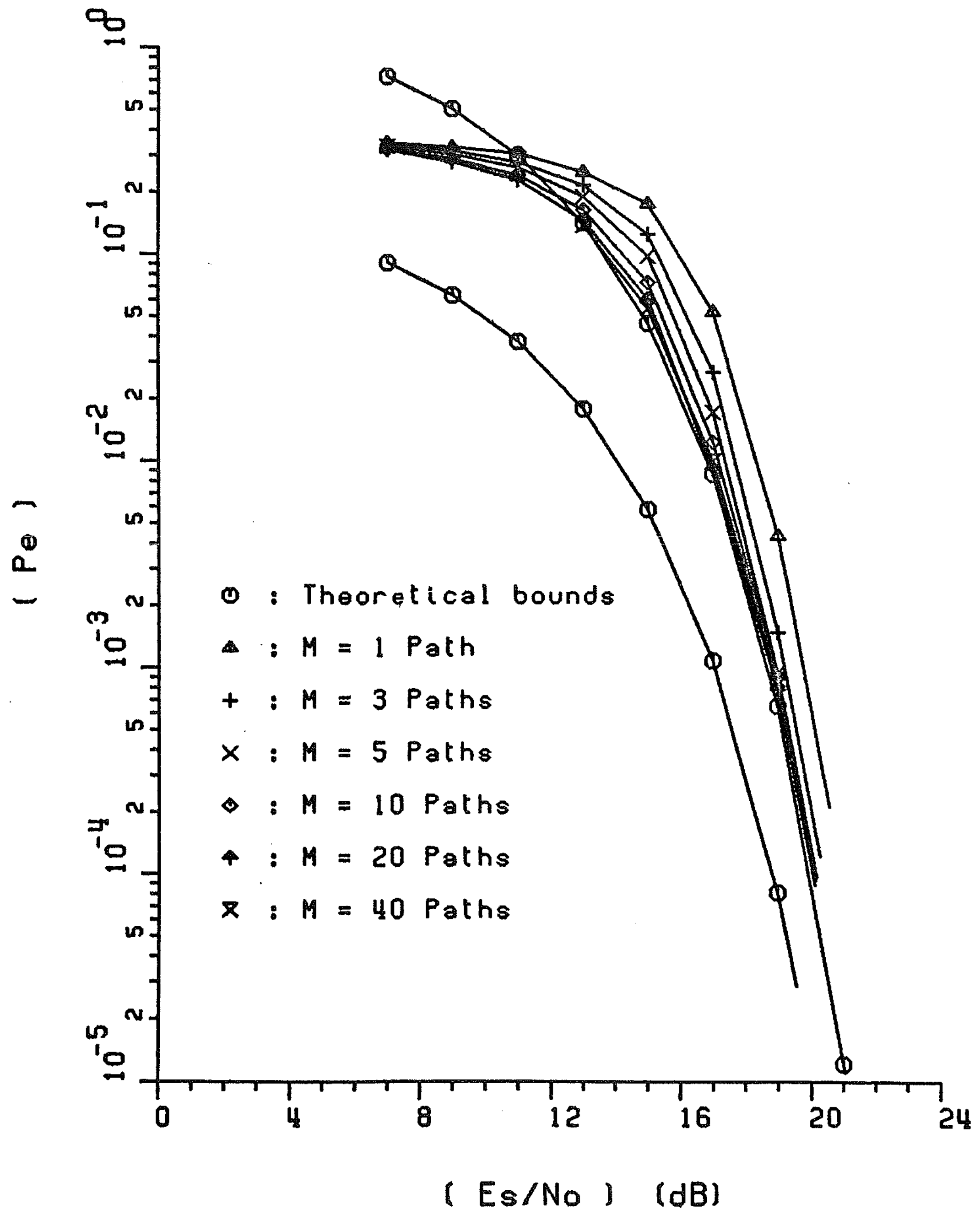


Figure 16: Comparisons of the observed Error Performances for the M-Path decoders (channel 3)

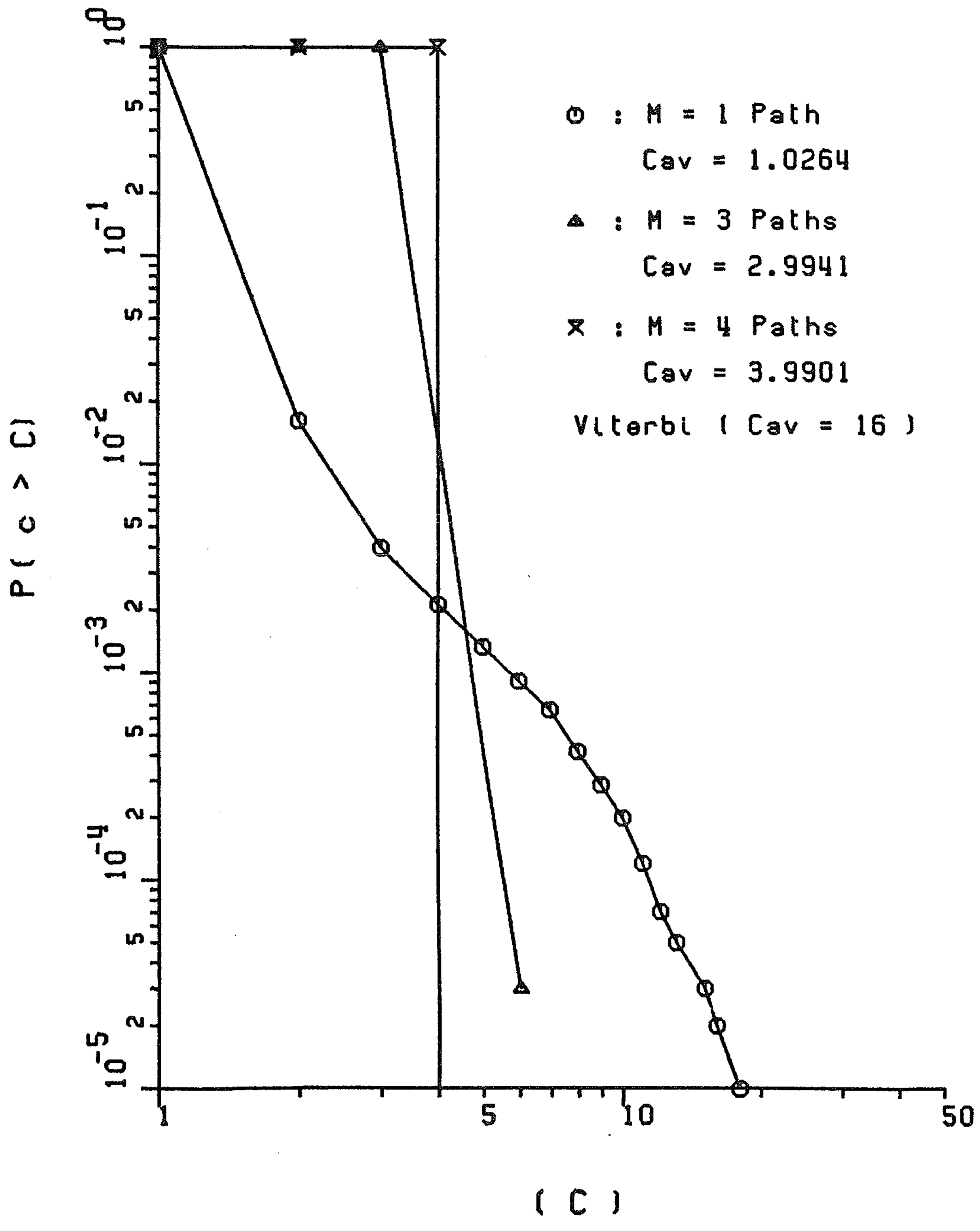


Figure 17: Computational distributions for M = 1, 3 and 4 paths, channel 1, E_s/N_o = 15 dB

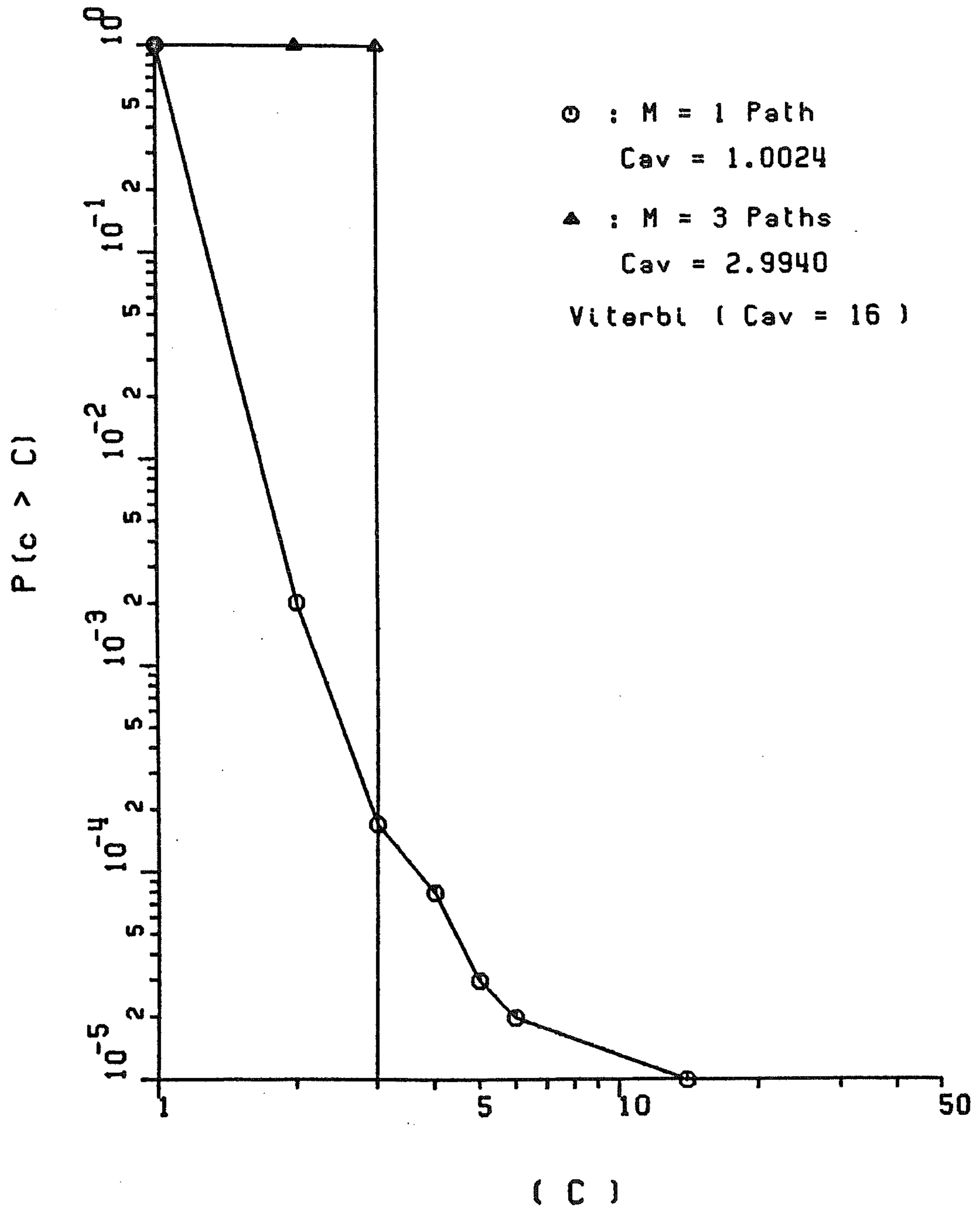


Figure 18: Computational distributions for M = 1 and 3 paths, channel 1, E_s/N_o = 17 dB

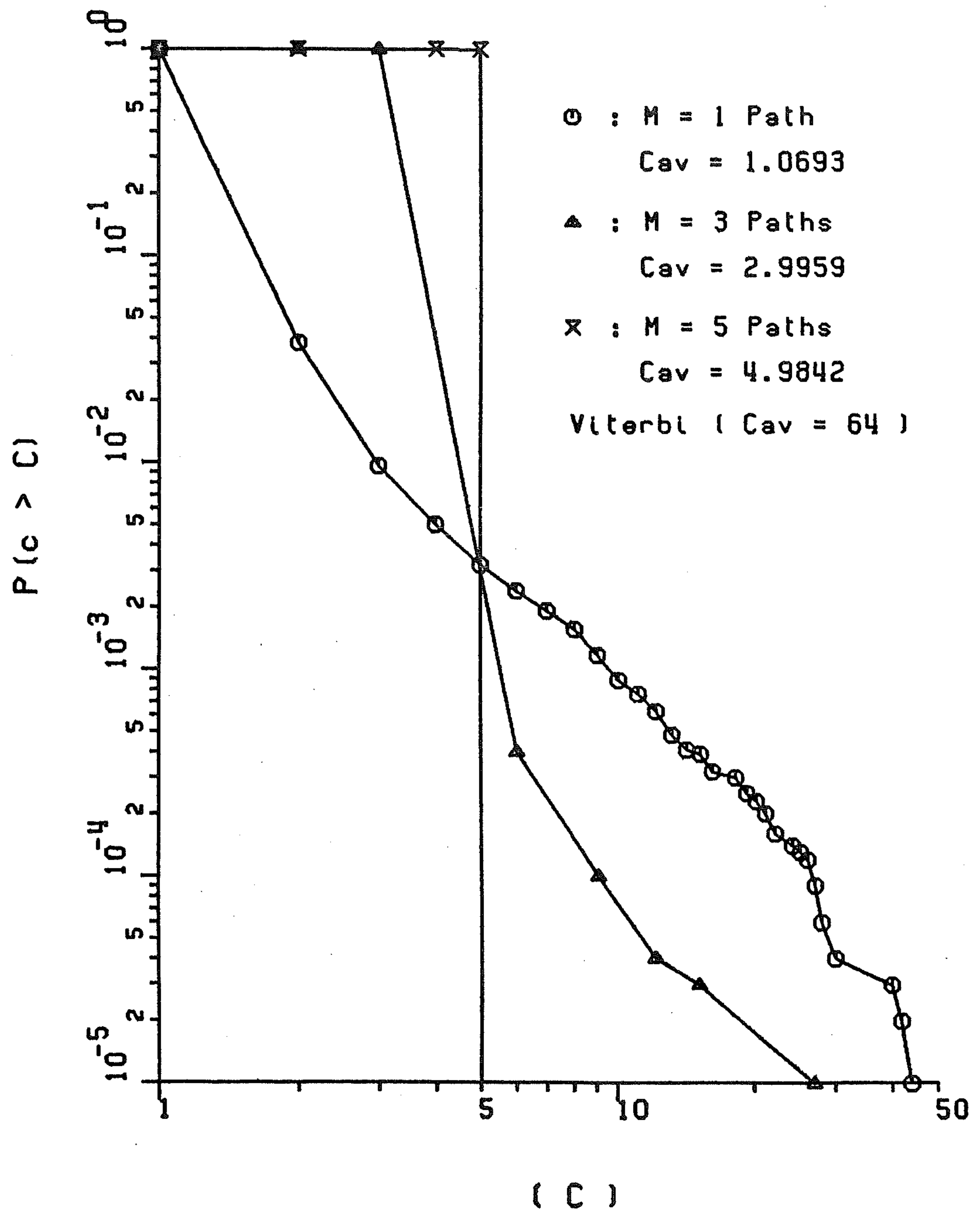


Figure 19: Computational distributions for M = 1, 3 and 5 paths, channel 2, E_s/N_o = 17 dB

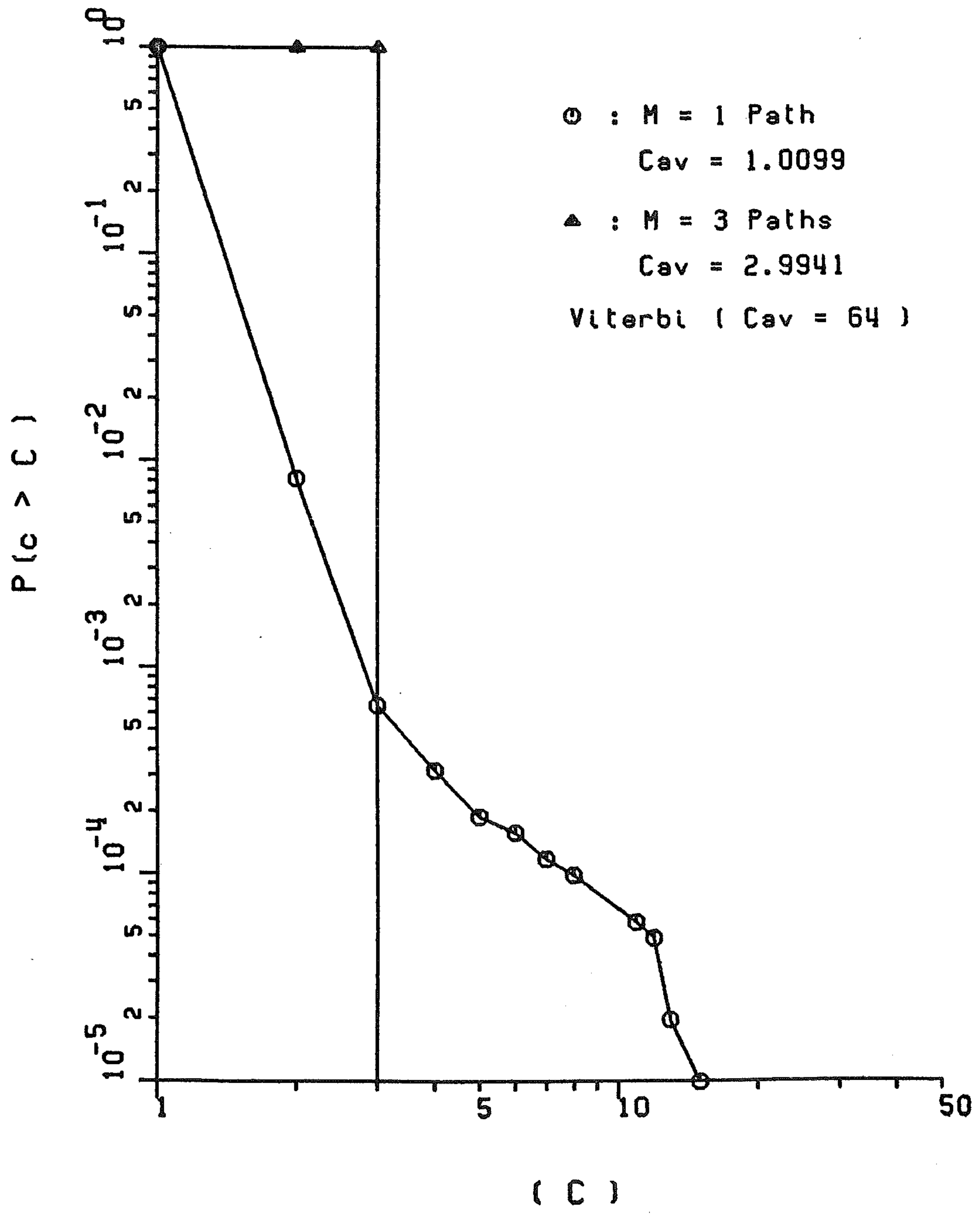


Figure 20: Computational distributions for M = 1 and 3 paths, channel 2, E_s/N_o = 19 dB

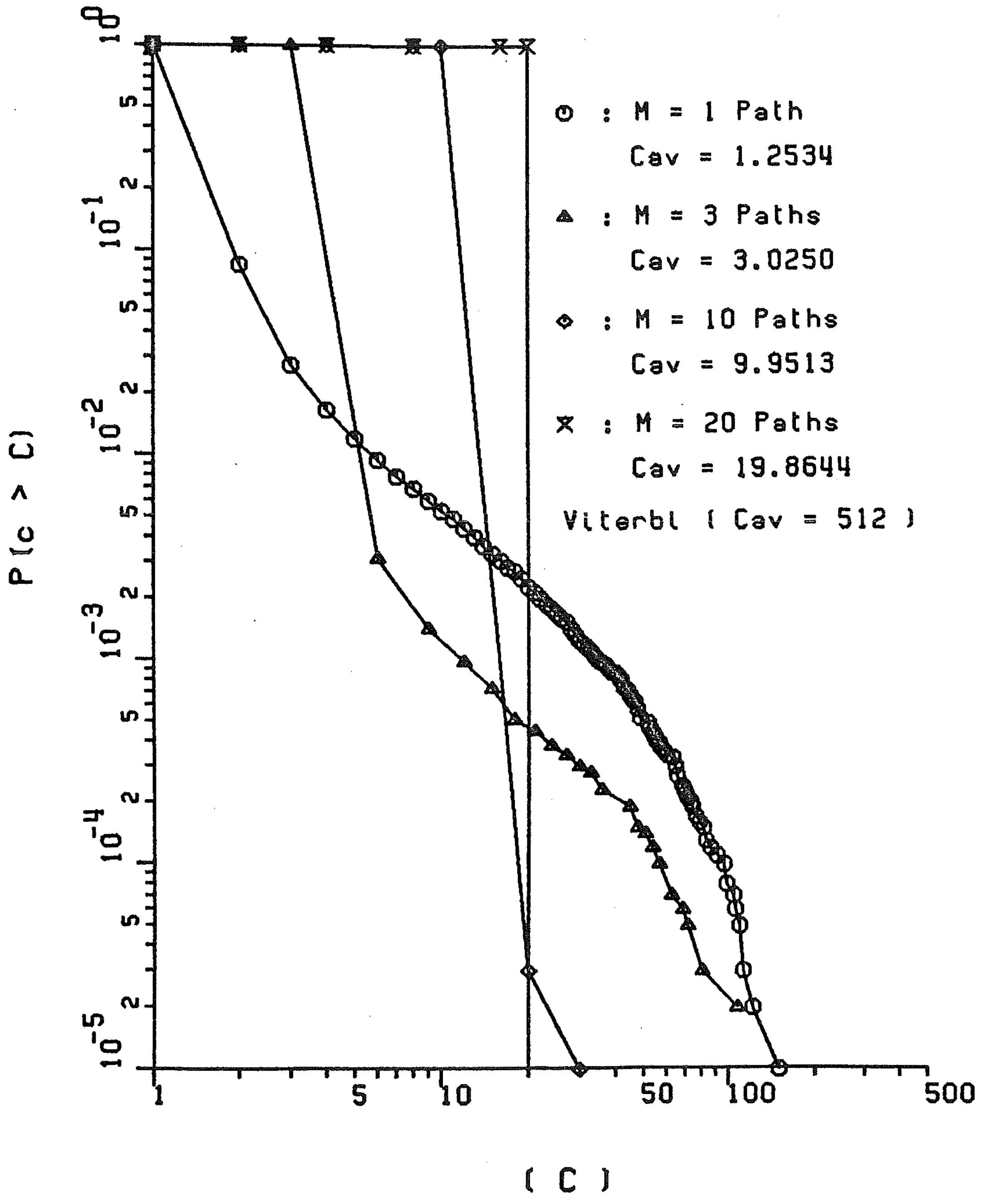


Figure 21: Computational distributions for M = 1, 3, 10 and 20 paths, channel 3, E_s/N_o = 19 dB

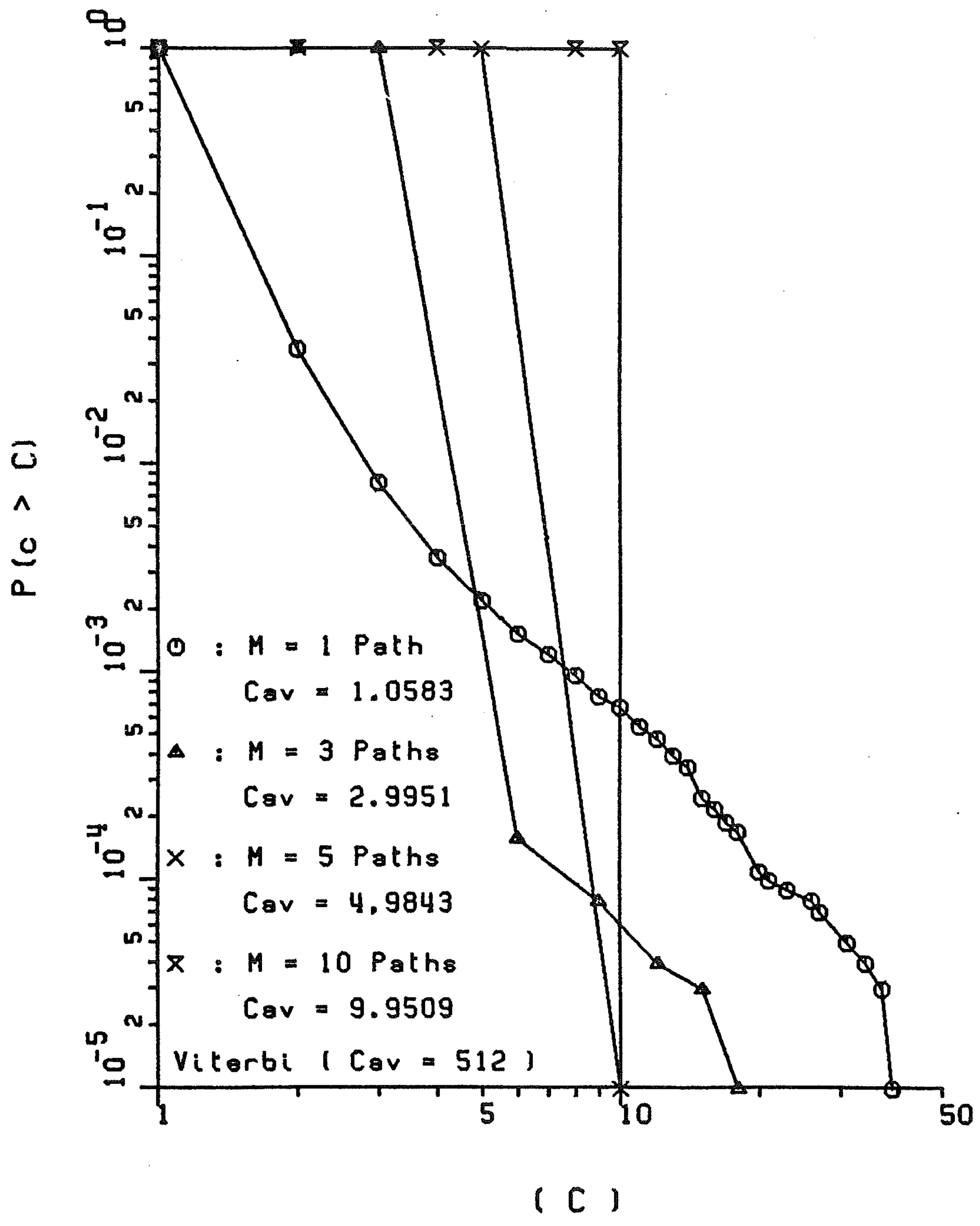


Figure 22: Computational distributions for M = 1, 3, 5 and 10 paths, channel 3, E_s/N_o = 21 dB

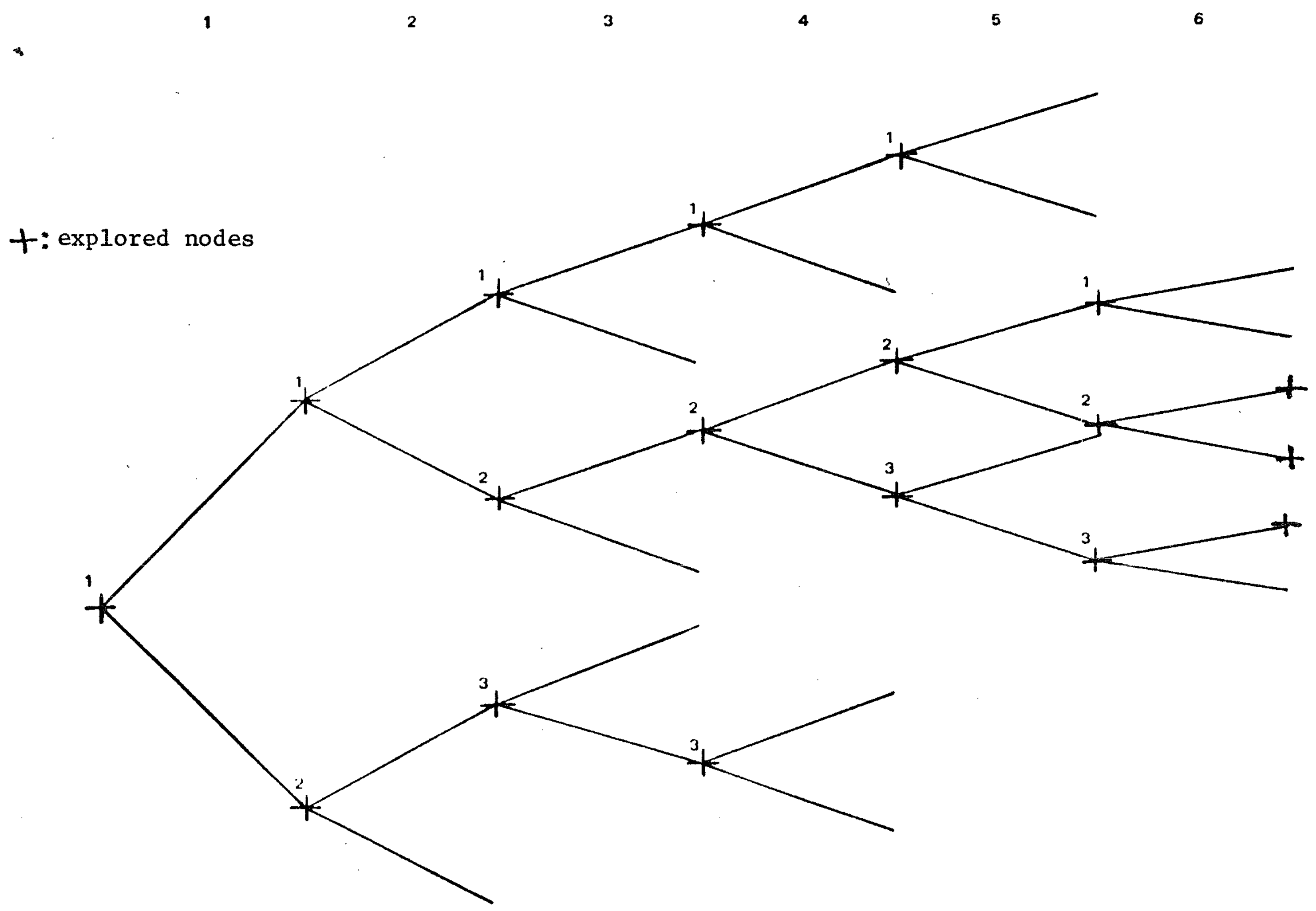


Figure 23: Illustrations of the tree exploration of the 3 - Path simplified algorithm

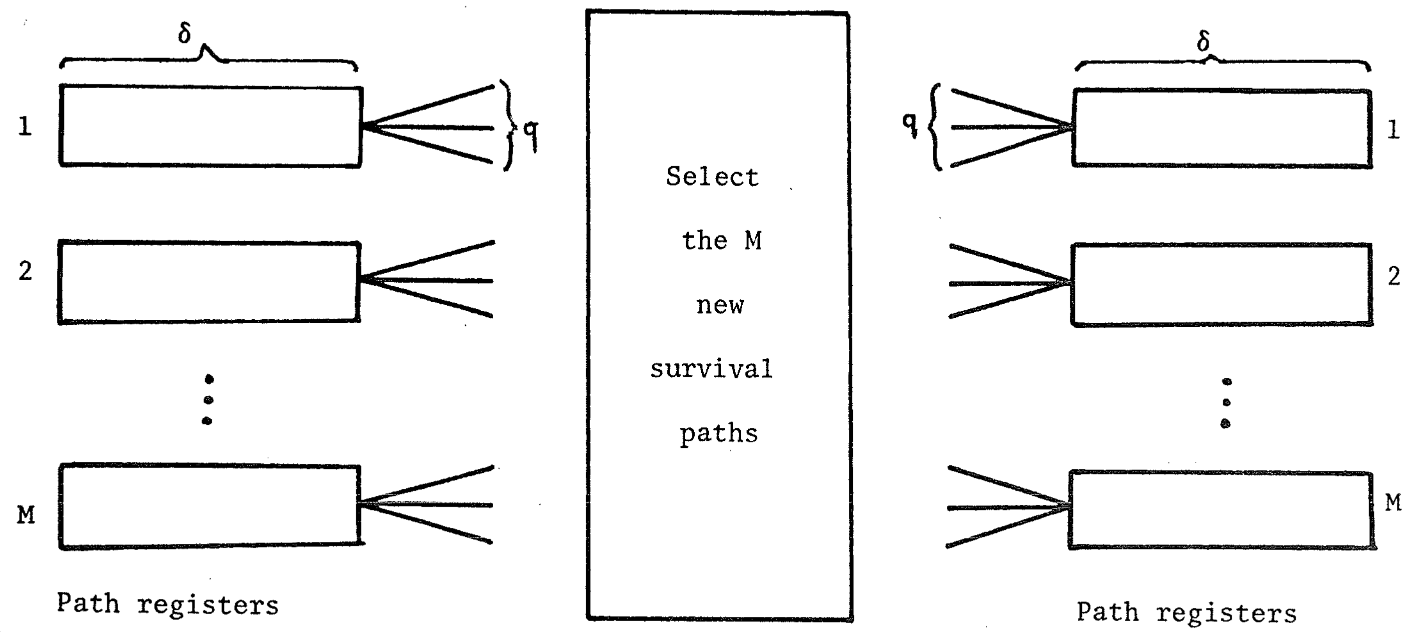


Figure 24: Conceptual structure of the simplified sequential decoder

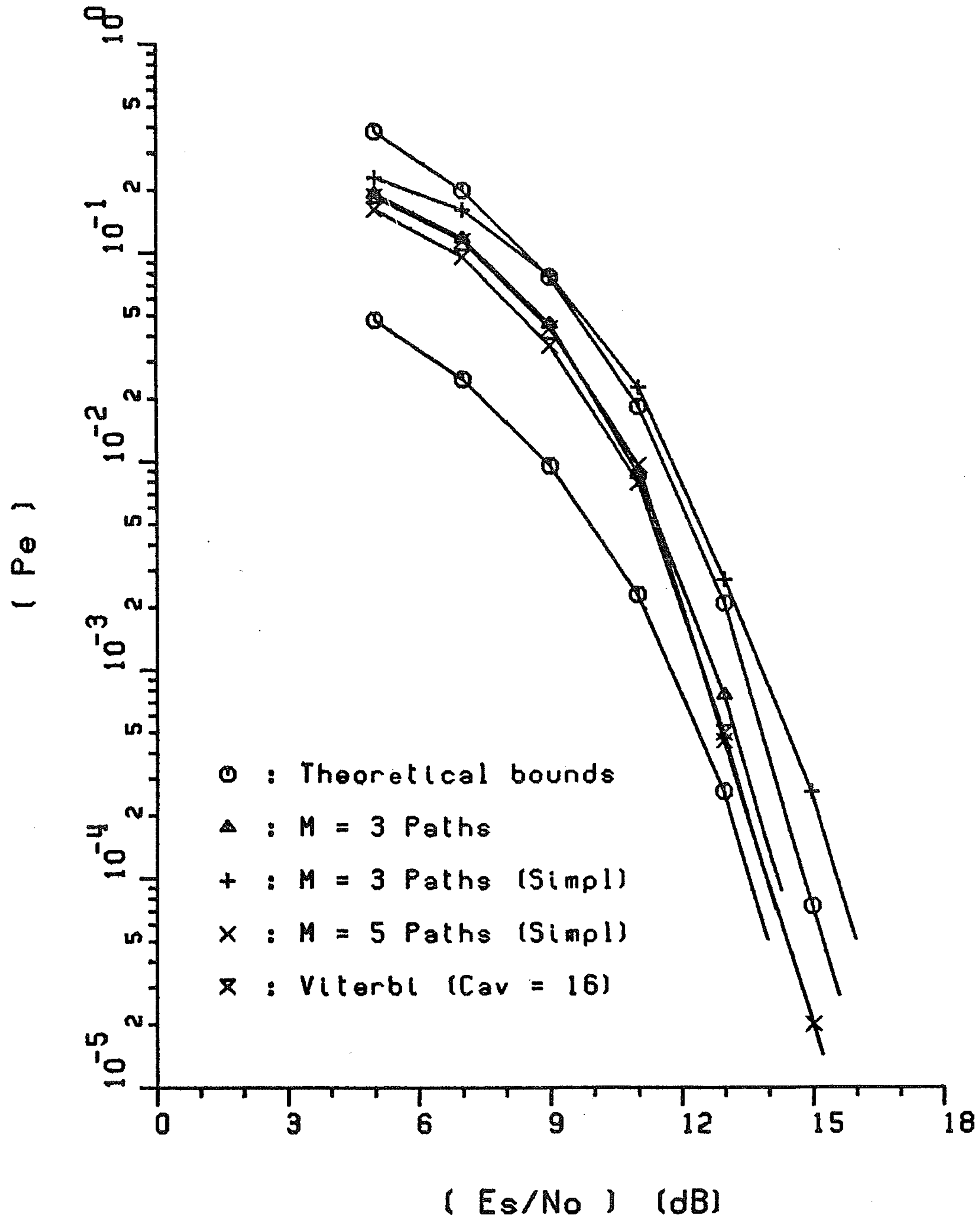


Figure 25: Comparisons of the observed error Performances of the M-Path simplified and Viterbi decoders (channel 1)

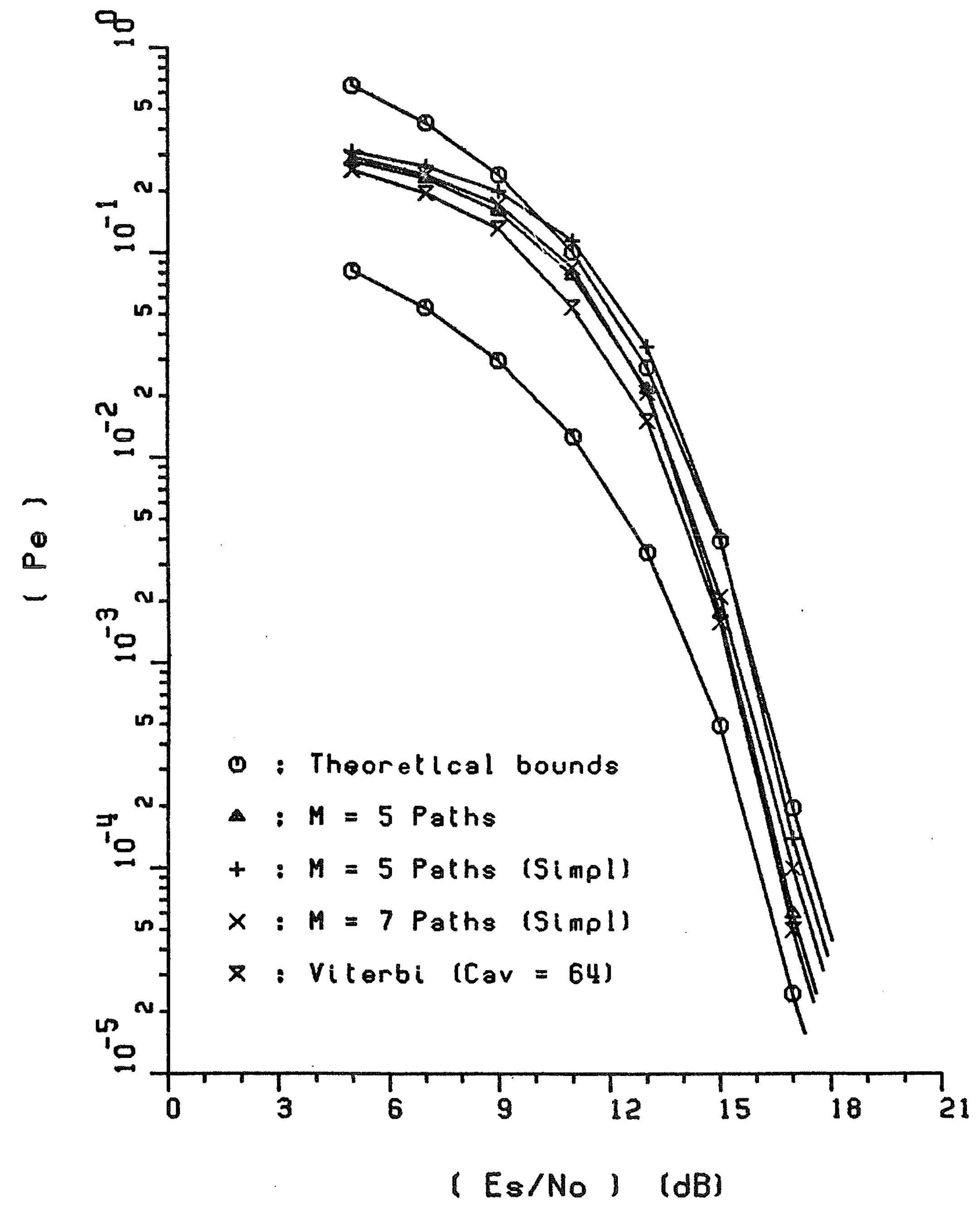


Figure 26: Comparisons of the observed error performances of the M-Path, simplified and Viterbi decoders (channel 2)

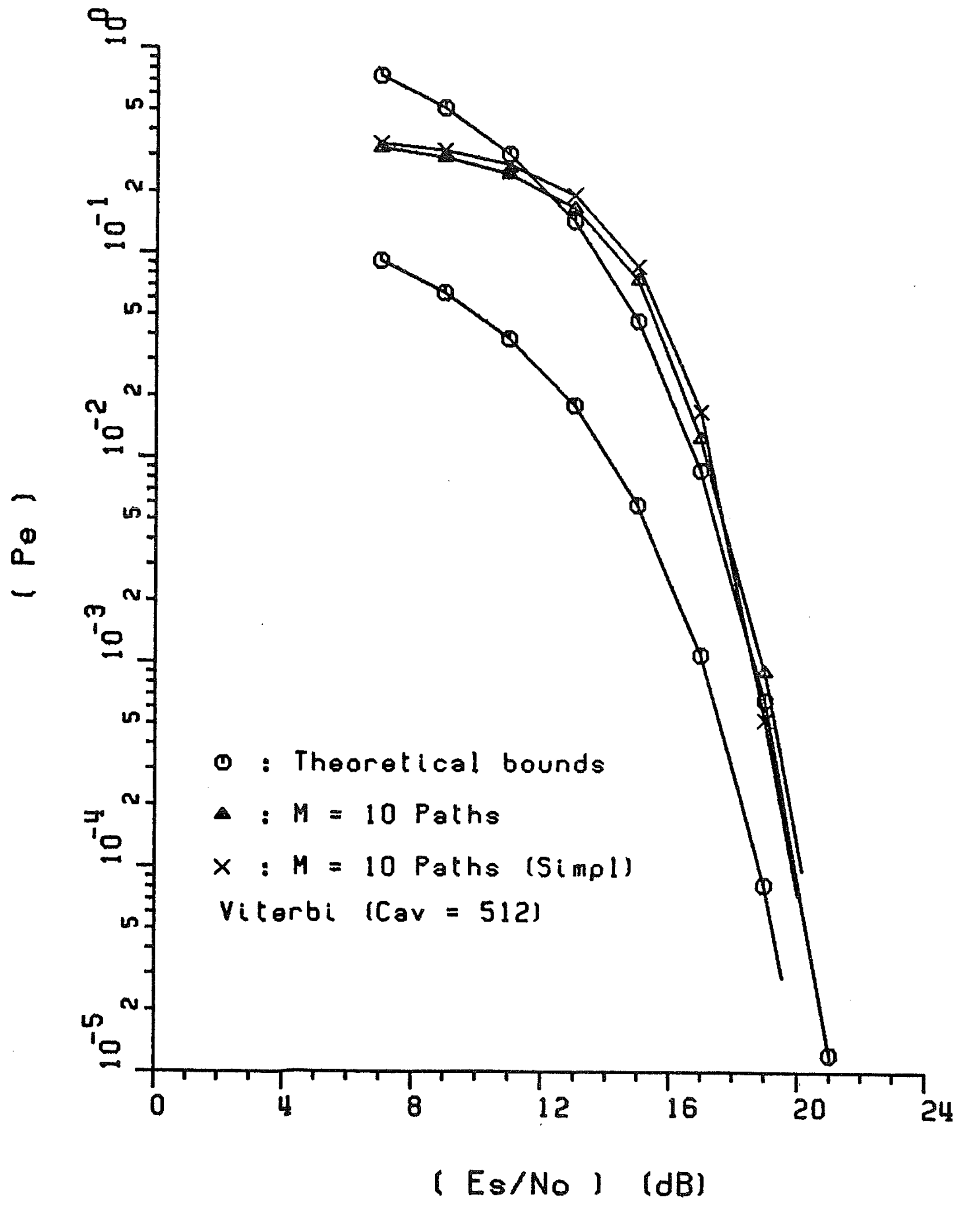


Figure 27: Comparisons of the observed error performances of the M-Path and the simplified decoders (channel 3)

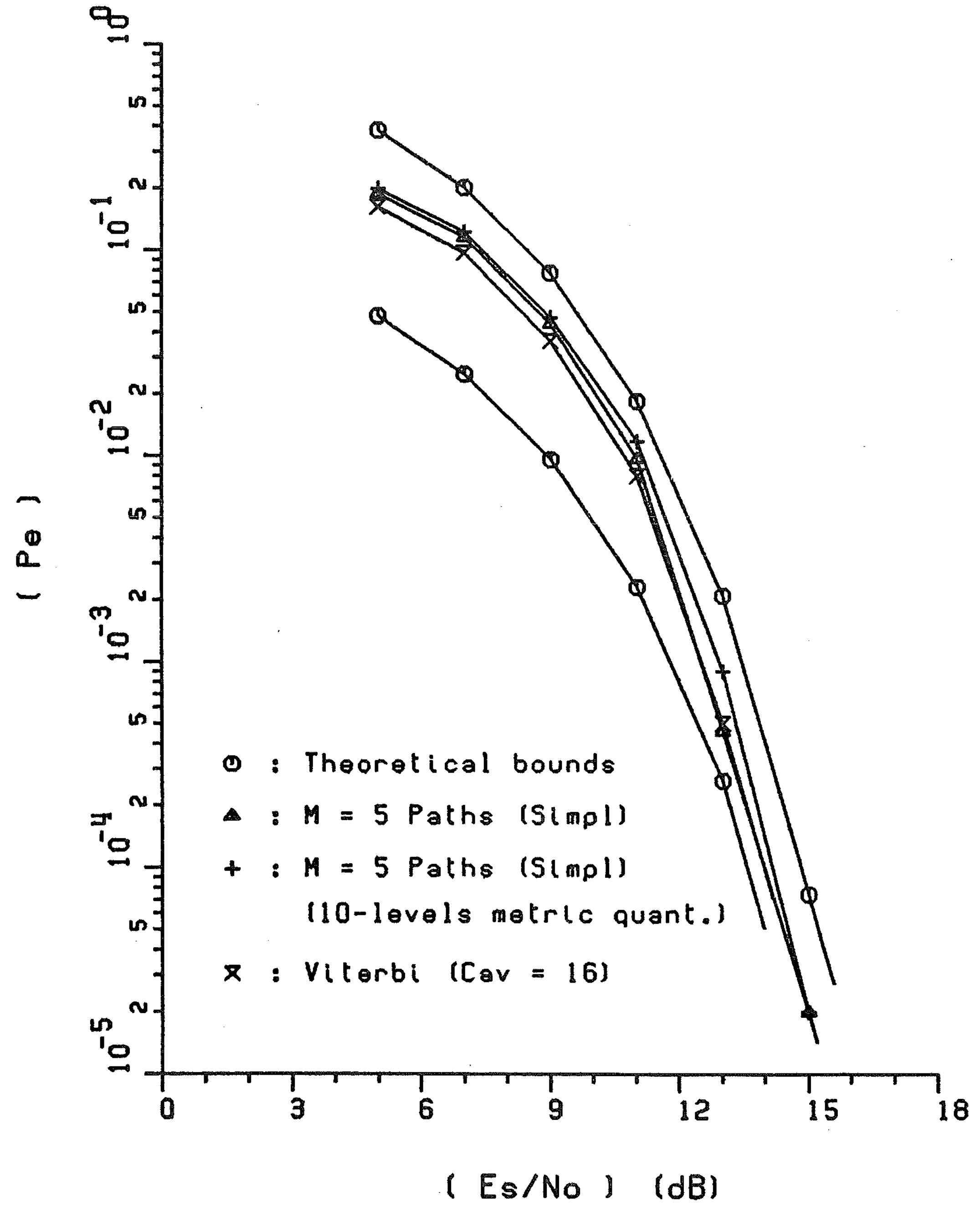


Figure 28: Effect of the metric quantization on the error performance of the simplified M-Path decoder (channel 1)

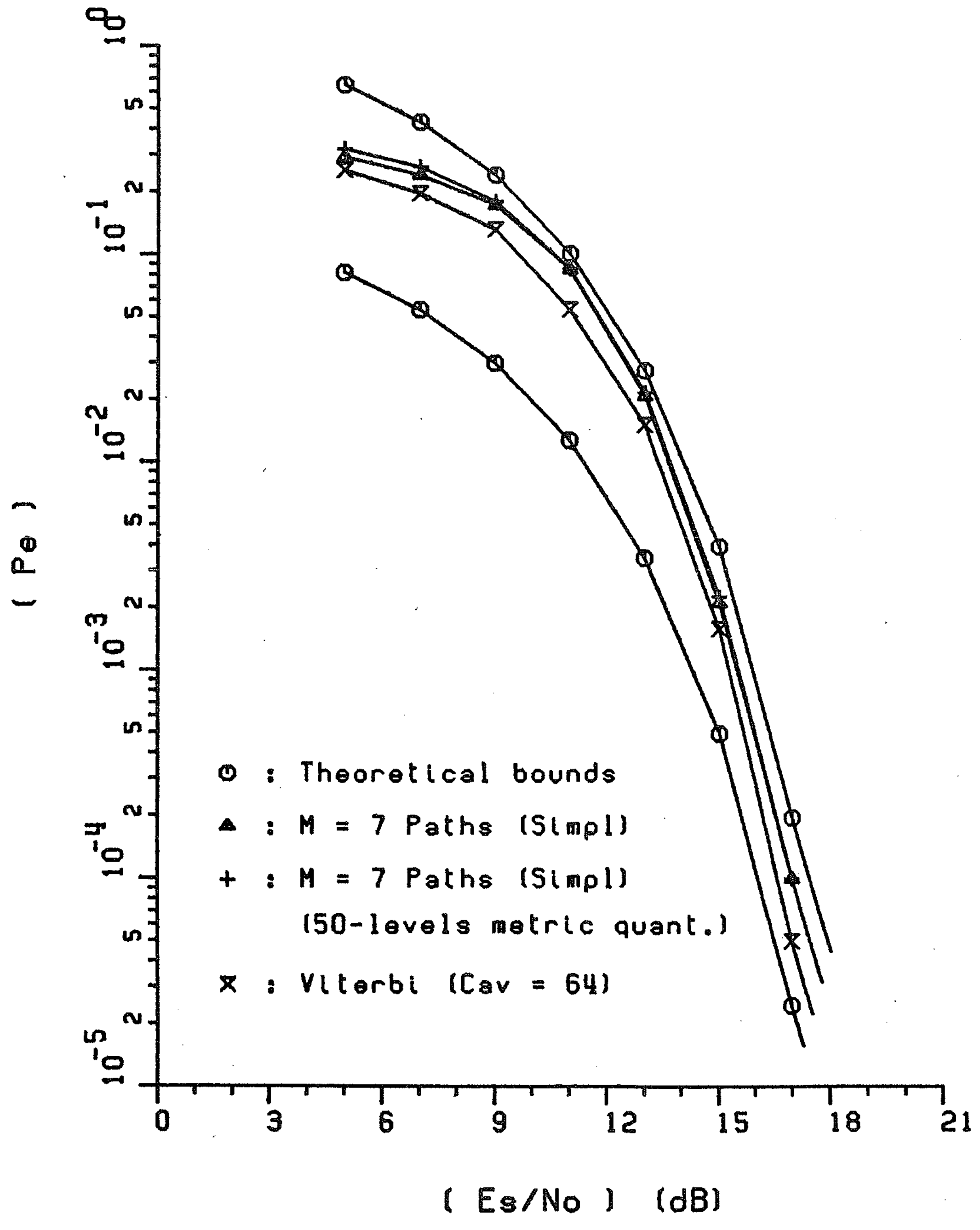


Figure 29: Effect of the metric quantization on the error performance of the simplified M-Path decoder (channel 2)

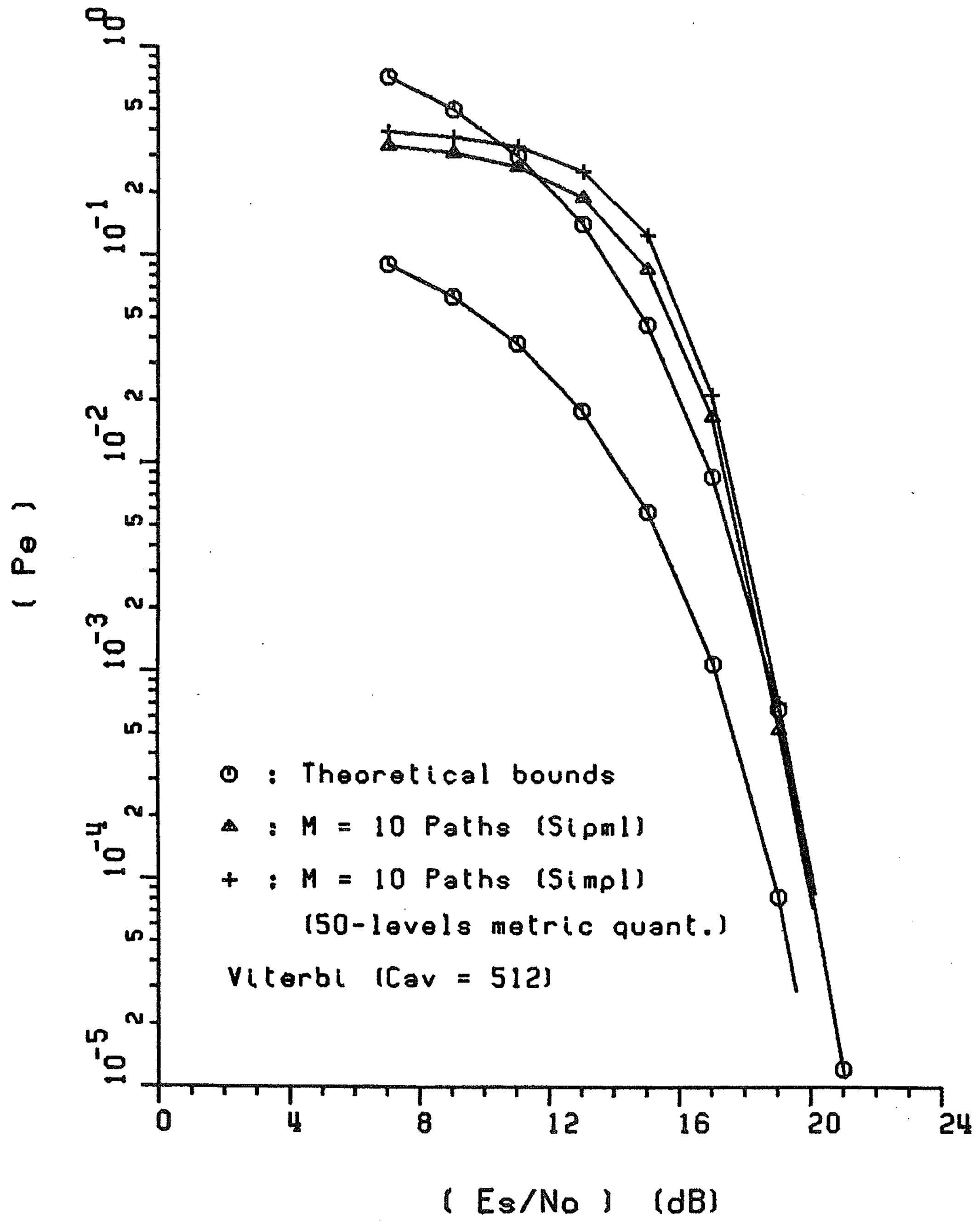


Figure 30: Effect of the metric quantization on the error performance of the simplified M-Path decoder (channel 3)

LIST OF TABLES

Table 1: Effects of the path history length on survival paths (channel 1).

Table 2: Effects of the path history length on survival paths (channel 2).

Table 3: Effects of the path history length on survival paths (channel 3).

Table 4: Performance comparisons between Viterbi, M-path Sequential and Simplified M-path decoders.

		E_s/N_o (dB)							
		11		13		15		17	
M	δ	NE	NC	NE	NC	NE	NC	NE	NC
	15	262	99	47	37	0	11	0	16
3	20	265	14	48	0	0	0	0	0
	25	265	0	48	0	0	0	0	0
	30	-	-	-	-	-	-	-	-
	15	110	572	11	218	0	113	0	52
5	20	114	145	11	16	0	5	0	0
	25	116	44	11	0	0	0	0	0
	40	116	0	11	0	0	0	0	0
	20	83	248	6	89	0	39	0	23
7	25	82	48	6	9	0	0	0	0
	35	82	5	6	0	0	0	0	0
	40	82	0	6	0	0	0	0	0

Table 1: Effects of the path history length on survival paths (channel 1).

		E_s/N_o (dB)							
		13		15		17		19	
M	S	NE	NC	NE	NC	NE	NC	NE	NC
3	15	878	390	250	231	18	116	0	54
	20	865	64	246	37	18	21	0	0
	25	865	15	245	8	18	0	0	0
	30	863	0	246	0	18	0	0	0
5	20	339	738	29	343	0	106	0	44
	25	327	291	28	84	0	8	0	0
	30	320	109	28	38	0	0	0	0
	45	314	7	28	0	0	0	0	0
7	25	194	573	22	213	0	67	0	16
	30	178	248	22	56	0	7	0	0
	35	173	97	22	17	0	0	0	0
	50	171	0	22	0	0	0	0	0

Table 2: Effects of the path history length on survival paths (channel 2).

		E_s/N_o (dB)							
		13		15		17		19	
M	δ	NE	NC	NE	NC	NE	NC	NE	NC
3	20	2583	156	1560	151	367	246	0	65
	30	2579	22	1556	0	368	32	0	0
	40	2580	0	1556	0	368	0	0	0
	50	-	-	-	-	-	-	-	0
5	30	1698	371	483	379	13	255	0	10
	40	1689	86	482	87	12	24	0	0
	50	1692	0	482	16	12	0	0	0
	60	1692	0	482	0	12	0	0	0
7	40	1457	224	246	221	4	76	0	0
	50	1461	18	240	39	4	30	0	0
	60	1461	0	240	11	4	11	0	0
	70	1461	0	240	0	4	0	0	0
10	50	909	163	124	93	4	30	0	0
	60	909	37	123	28	4	11	0	0
	70	909	0	122	11	4	0	0	0
	80	909	0	122	0	4	0	0	0

Table 3: Effects of the path history length on survival paths (channel 3).

	M	M-Path			Simplified M-Path $C_{av} = M$
		P_e	C_{av}	C_{av}/C_{vit}	P_e
Channel 1 Memory 4 E_s/N_o = 13 dB	1	1.36×10^{-3}	1.01	6.9%	——
Viterbi: $P_e = 5\times10^{-4}$	3	7.6×10^{-4}	3.00	18.7%	2.73×10^{-3}
$C_{vit}^{*} = 16$	5	7.5×10^{-4}	4.98	31.2%	5.1×10^{-4}
Channel 2 Memory 6	1	1.4×10^{-4}	1.07	1.67%	——
E_s/N_o = 17 dB	3	6.0×10^{-5}	2.99	4.68%	1.58×10^{-3}
Viterbi: $P_e = 5\times10^{-5}$	5	6.0×10^{-5}	4.98	7.78%	1.4×10^{-4}
$C_{vit} = 64$	7	6.0×10^{-5}	6.97	10.9%	9.0×10^{-5}
Channel 3 Memory 9	1	4.38×10^{-3}	1.25	0.24%	——
E_s/N_o = 19 dB	5	9.4×10^{-4}	4.99	0.97%	4.28×10^{-3}
Viterbi: P_e = ——	10	8.9×10^{-4}	9.93	1.94%	5.2×10^{-4}
$C_{vit} = 512$	40	8.7×10^{-4}	39.66	7.74%	——

* C_{vit}: Average number of computations of Viterbi Algorithm.

Table 4: Performance comparisons between Viterbi, M-path Sequential and Simplified M-path decoders.

FOR INTERSYMBOL INTERFERENCE