(NASA-CR-170163) TRELLIS PHASE CODES FOR POWER-BANDWITH EFFICIENT SATELLITE COMMUNICATIONS Final Report, 10 Apr. 1979 - 22 Feb. 1981 (Virginia Univ.) 79 p HC A05/MF A01 CSCL 17B G3/32

N83-22497

Unclas 15260

Final Report

NAS5-25634

TRELLIS PHASE CODES FOR POWER/BANDWIDTH EFFICIENT SATELLITE COMMUNICATIONS

Submitted to:

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

Submitted by:

S. G. Wilson
Principal Investigator

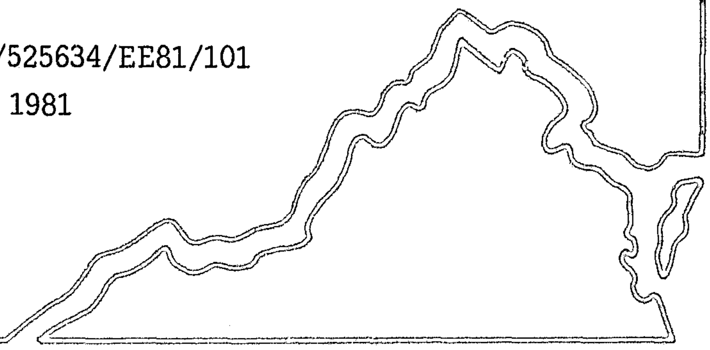
J. H. Highfill
Senior Scientist

C-D Hsu
Research Assistant

R. Harkness
Research Assistant

Report No. UVA/525634/EE81/101

April 1981



UNIVERSITY OF VIRGINIA

COMMUNICATIONS SYSTEMS LABORATORY
DEPARTMENT OF ELECTRICAL ENGINEERING
SCHOOL OF ENGINEERING AND APPLIED SCIENCES
UNIVERSITY OF VIRGINIA

CSL

Final Report

NAS5-25634

TRELLIS PHASE CODES FOR POWER/BANDWIDTH EFFICIENT SATELLITE COMMUNICATIONS

Submitted to:

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

Submitted by:

S. G. Wilson
Principal Investigator

J. H. Highfill
Senior Scientist

C-D Hsu
Research Assistant

R. Harkness
Research Assistant

Department of Electrical Engineering
RESEARCH LABORATORIES FOR THE ENGINEERING SCIENCES
SCHOOL OF ENGINEERING AND APPLIED SCIENCE
UNIVERSITY OF VIRGINIA
CHARLOTTESVILLE, VIRGINIA

Report No. UVA/525634/EE81/101

Copy No. 62

April 1981

1. Report No.

2. Government Accession No.

3. Recipient's Catalog No.

4. Title and Suotitle

Trellis Phase Codes for Power/Bandwidth
Efficient Satellite Communications

5. Report Date
April 1981

6. Performing Organization Code
NAS 5-25634

7. Author(s)

S. G. Wilson, J. H. Highfill
C-D. Hsu, and R. Harkness

8. Performing Organization Report No.
UVA/525634/EE81/101

9. Performing Organization Name and Address

Communications Systems Laboratory
School of Engineering and Applied Science
University of Virginia, Thornton Hall
Charlottesville, VA 22901

10. Work Unit No.

11. Contr ct or Grant No.

12. Sponsoring Agency Name and Address

NASA Lewis Research Center
21000 Brookpark Road
Cleveland, OH 44135

13. Type of Report and Period Covered
Final Report
04/10/79 - 02/22/81

14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

This report summarizes work performed in support of improved power and spectrum utilization on digital satellite channels. Specific attention is given to the class of signalling schemes known as continuous phase modulation (CPM).

The specific work described in this report addresses

(1) analytical bounds on error probability for multi-h phase codes,

(2) power and bandwidth characterization of 4-ary multi-h codes, and

(3) initial results of channel simulation to assess the impact of band-limiting filters and nonlinear amplifiers on CPM performance.

17. Key Words (Suggested by Author(s))

modulation and coding
satellite communication

18. Distribution Statement

19 Security Classif. (of this report)
Unclassified

20. Security Classif. (of this page)
Unclassified

21. No. of Pages
79

22. Price

TABLE OF CONTENTS

Page

1.0 INTRODUCTION 1

2.0 ERROR BOUNDS FOR MULTI-H PHASE CODES 2

I. INTRODUCTION 2
II. BACKGROUND 3
III. DEVELOPMENT OF ERROR BOUNDS, UNLIMITED PATH MEMORY 8
IV. UPPER BOUND FOR FINITE MEMORY 18
V. NUMERICAL EXAMPLES 22

REFERENCES 27

3.0 4-ARY MULTI-H CODES 28

REFERENCES 37

4.0 CHANNEL SIMULATION 38

4.1 SIMULATION RESULTS 41
4.2 SOFTWARE DESCRIPTION 47

REFERENCES. 57

APPENDIX A 58

1.0 INTRODUCTION

In this final report on "Trellis Phase Codes for Power/Bandwidth Efficient Satellite Communication," (NAS5-25634) we summarize work performed since the last report of July 1980. Since that time we have continued our study of multi-h and partial response FM digital modulations, and have emphasized three special aspects:

1. analytical error bounds for multi-h phase codes
2. a study of 4-ary multi-h codes
3. channel simulation for partial-response and multi-h codes to assess the effects of bandlimiting and nonlinear amplification.

These are discussed in detail in following chapters of the report.

Additional work performed under the contract has been described in two pervious semi-annual reports to NASA/Lewis Research Center, published in November 1979 and July 1980.

2.0 ERROR BOUNDS FOR MULTI-H PHASE CODES

This section is a revision of a manuscript accepted for publication in the IEEE Transaction on Information Theory. Numbering of equations and references is internally consistent.

I. INTRODUCTION

Multi-h phase codes, described in detail by Anderson and Taylor [1], and earlier conceived by Miyakawa, et al. [2], represent a class of constant envelope signal designs providing attractive gains in the power/bandwidth tradeoff, relative to MSK or QPSK. Time-variation of the transmitter frequency deviation parameter among a small set of rational numbers provides delayed remergers in the phase trellis. This in turn provides increased minimum distance between signals which may be exploited by a maximum likelihood sequence estimator to enhance detection efficiency. Analysis thus far of these signals has concentrated on minimum distance to yield asymptotic behavior, and some simulation results are available, [3,4].

These signals may be represented in terms of a finite-state trellis, though a time-varying one due to the cyclical nature of the modulator deviation parameter. In order to determine bounds on error probability, we extend the union bound/transfer function approach earlier used for linear convolutional codes [5] to the multi-h phase code case. In closely-related work, Aulin [6] has developed similar bounds for the class of partial-response FM codes with fixed modulator deviation. As with earlier applications, the principal difficulty is in formulating the state diagram and the necessary branch gains. Once this is accomplished, numerical evaluation is straightforward.

Lower bounds on receiver performance are more easily obtained in terms of a single (or few) neighboring paths, and it will be seen that the two bounds are close for performance levels of typical interest.

The paper is organized as follows. Section II provides a brief review of the multi-h signal structure, and its state-space description. Section III develops the upper and lower bounds for the case of unlimited path memory, incorporating the concept of difference-state trellises and the associated transfer functions. The effects of finite decoder path memory are addressed in Section IV. The paper concludes with several illustrative examples comparing the analytical bounds with simulation data.

II. BACKGROUND

We assume the transmitted signal is of the constant-envelope form

$$s(t) = (2P)^{\frac{1}{2}} \cos (\omega_c t + \theta_o + \phi(t,\bar{a})) \qquad (1)$$

where P is the power of the symbol, θ_o is an arbitrary phase angle, and $\phi(t,\bar{a})$ is the (causal) phase modulation induced by a semi-infinite sequence

$$\bar{a} = (a_1, a_2, .. a_n ....)$$

The symbols a_n are selected independently each T seconds with equiprobability from the set {-(M-1), ..., -3, -1, 1, 3, ... (M-1)} so that the signalling is M-ary.

Over the n^{th} signalling interval, $(n-1)\ T \leq t \leq nT$, the phase term in multi-h coding is a continuous waveform given by

$$\phi(t,\bar{a}) = \pi \sum_{j=1}^{n-1} a_j\, h_j + \pi \int_{(n-1)T}^{t} a_n\, h_n\, g(\tau - (n-1)T)d\tau \qquad (2)$$

The first term in (2) gives the accumulated phase due to symbols through a_{n-1}, while the second term provides the time-varying phase increment over the n^{th} interval. Also in (2) we have defined $g(\tau)$ to be a frequency pulse for spectrum-shaping purposes. We assume that $g(\tau)$ is non-zero for $0 \leq \tau \leq T$, and that

$$\int_0^T g(\tau)d\tau = 1 \qquad (3)$$

Under this normalization h_n plays the role of a modulation index, and h_n/T is proportional to the average frequency deviation over the symbol interval. We remark this description is more general than that of [1], in that M-ary modulation, as well as nonlinear phase trajectories, are allowed.

In standard digital FM transmission, h_n is some constant h. For MSK designs, h = 1/2, and for binary CPFSK, h = 0.715 maximizes the detection efficiency. However, in multi-h coding, h_n is cyclically chosen from a set H_k of K rational numbers:

$$H_K = \{ \frac{p_1}{q}, \frac{p_2}{q}, \ldots \frac{p_K}{q} \} \triangleq \{h^{(1)}, h^{(2)}, \ldots . h^{(K)} \} \qquad (4)$$

That is

$$h_1 = h^{(1)},\ h_2 = h^{(2)},\ h_K = h^{(K)}$$

and thereafter

$$h_{n+jK} = h_n$$

The set of possible phase trajectories associated with data sequences of length n is a phase tree of M^n trajectories. By virtue of the rational-index assumption, it will be seen that the number of distinct phases, modulo - 2π, attained at the end of symbol intervals is exactly q. (A complication is that there may be a different q values for different times, but this does not influence our development). Thus, the tree collapses to a phase trellis, describing the allowable phase transitions, and the allowable signal trajectories.

By defining the signal state at time n as

$$S_n = \pi \sum_{j=1}^{n-1} h_j \, a_j, \text{ mod } 2\pi \qquad (5)$$

we obtain a finite-state description of the signal, whereby the state evolves among q states (phases) according to

$$S_{n+1} = S_n + \pi \, h_n \, a_n, \text{ mod } 2\pi \qquad (6)$$

This description is essential for maximum likelihood decoding via the Viterbi algorithm.

As a unifying example throughout, we shall consider a simple binary code defined by $H_2 = \{2/4, 3/4\}$. The state trellis for this example is shown in Figure 1, where it may be seen that the allowable state transitions are time-varying, depending on which deviation is currently in effect. The trellis shows state transitions, and is not meant to indicate actual phase variation, although in the usual case the phase segments are piece-wise linear. The trellis is structurally very regular, its appearance complicated only by the mod 2π phase definition.

For this example, it may be seen that two data sequences which differ in the first position produce state sequences which are unmerged

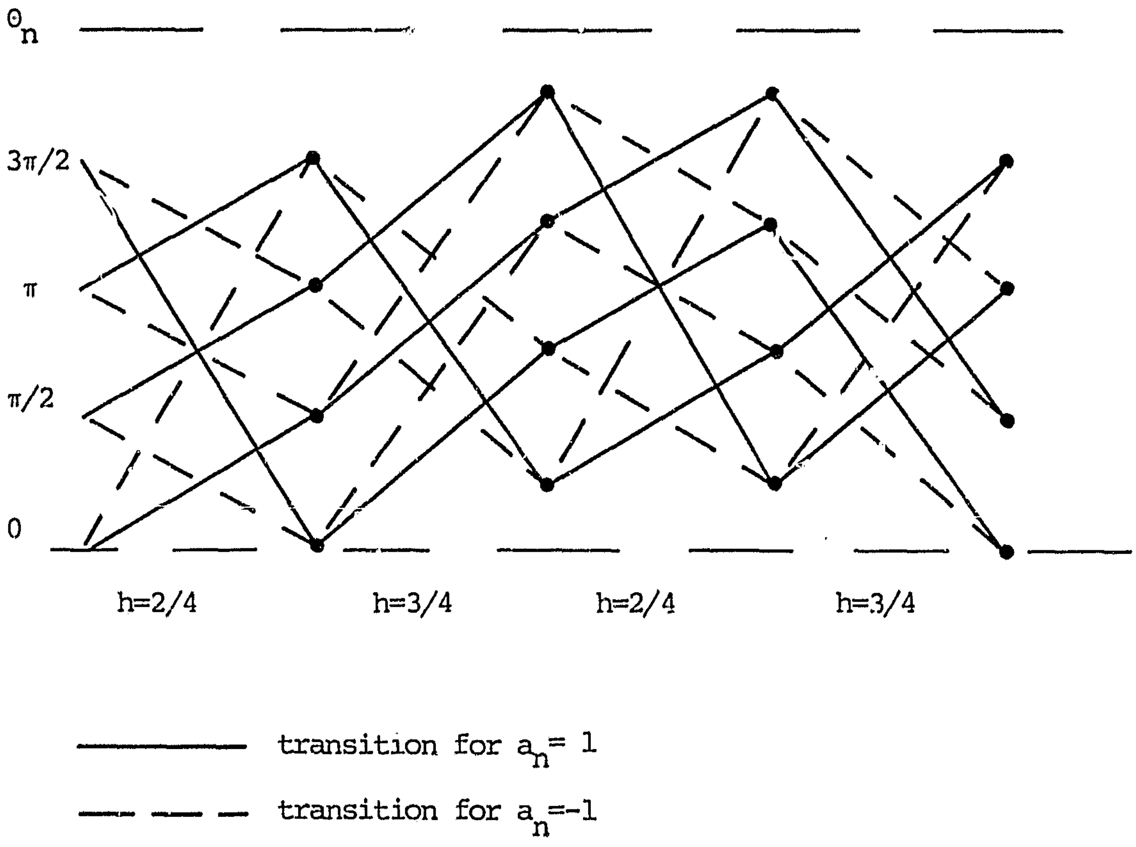


Figure 1. State Trellis for {2/4,3/4} Code

for at least K + 1 = 3 bits in the trellis. Generally speaking long remerger lengths are desirable for energy efficiency, and it may be shown that there exist sets H_K which achieve an unmerged span of K+1 levels provided $q \geq M^K$, [1], [4].

Assuming coherent reception of the multi-h signal in white Gaussian noise, the optimal sequence detector is comprised of a bank of correlators coupled with a q-state Viterbi algorithm decoder. The correlators in effect generate branch metrics for the trellis search. The receiver locates the data sequence producing the largest waveform correlation, ignoring for the moment questions of quantization and path memory. An alternate view is that the receiver does minimum distance decoding, with distance measured in the L_2 norm. Distance between waveforms is key concept in the development, and is given by

$$d_{12_N}^2 = \frac{1}{2E} \int_o^{NT} (s_1(t) - s_2(t))^2 dt \qquad (7)$$

The normalization is selected so that the probability of error in a two-signal test is

$$P_e = Q\left(\frac{d_{12}^2 E}{N_o}\right)^{1/2} \qquad (8)$$

where E = PT is the energy per symbol, and N_o is the one-sided noise spectral density. With this definition, antipodal signals of length T have a normalized distance of 2.

Quantities of special importance are the minimum distance over an interval of length N, defined as

$$d_{min_N}^2 = \min_i \min_j d_{ij_N}^2 \qquad (9)$$

and its limiting value in N, d^2_{free}. These will determine the detection efficiency at large E/N_o.

III. DEVELOPMENT OF ERROR BOUNDS, UNLIMITED PATH MEMORY

In this section we proceed to develop upper and lower bounds on the probability that a decoding error event begins at some level n, denoted P_e, and on the probability of symbol error, denoted P_s. The receiver memory is assumed arbitrarily large. The development utilizes standard transfer function bounding, e.g., [5], modified to incorporate difference state concepts [6] and the time-varying nature of the trellis.

An error event begins at time n if the decoder selects sequence $\bar{b}$ rather than the transmitted sequence $\bar{a}$, where $\bar{a}$ and $\bar{b}$ first differ at time n. The probability that two sequences are confused depends only on their phase difference trajectory (as well as E/N_o). Since phase is in turn proportional to the data sequence, we need only consider possible data difference sequences, rather than consider all possible sequence pairs ($\bar{a}$, $\bar{b}$). This fact has been utilized by Aulin [6] for the case of partial-response FM trellis codes.

For multi-h codes, the difference-state for a pair of sequences is given by the accumulated phase difference, mod 2π. That is

$$\Delta_n = \sum_{j=1}^{n-1} \pi h_j \gamma_j \quad , \text{mod-}2\pi \qquad (10)$$

where $\gamma_j = a_j - b_j$. The difference-state trellis shows the possible evolution of difference-state among the q possible values for (10). Since γ_j is in a set of size 2M-1, there are 2M-1 branches entering and leaving each node of the difference-state trellis. Figure 2 illustrates the difference-state trellis for the {2/4, 3/4} binary code described earlier.

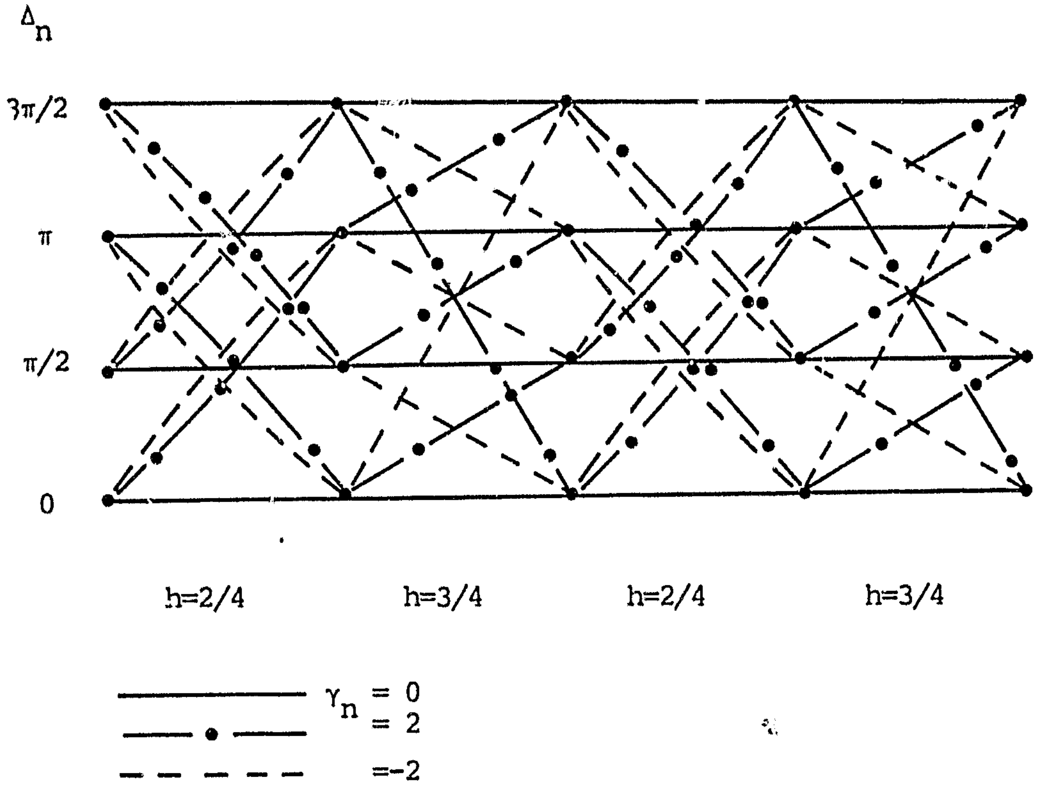


Figure 2. Difference-State Trellis for {2/4,3/4} Code

Error free decoding corresponds to residence in the $\Delta_n = 0$ state, i.e. the selected path agrees with the transmitted path. An error event begins at time n if the selected path splits from the zero difference-state at the n^{th} level. The probability of this event, P_e, is upper bounded by the sum of probabilities that the decoder selects a specific difference sequence which leaves the zero difference-state at time n and remerges later in the trellis.

Actually this bound is unduly pessimistic since transmitted sequences are not in one-to-one correspondence with difference sequences. For example consider M = 2. The allowed values for γ_n are 2, 0, -2. $\gamma_n = 2$ can be formed only by $(a_n, b_n) = (1,-1)$, whereas $\gamma_n = 0$ can correspond to $(a_n, b_n) = (1,1)$ or $(-1,-1)$. To properly average with respect to transmitted sequences, we must give lesser weight to the influence of $\gamma_n = \pm 2$ transitions. Similar effects happen with larger M, and Aulin [6] has shown a multiplying factor

$$c = \prod_n \, [M - |\gamma_n|/2] \; /M \tag{11}$$

must be applied to the probability of each difference sequence to provide the proper averaging.

Assuming for the present that deviation $h^{(1)}$ is effect at time n, we can now write the conditional event error probability as

$$P_e|_1 < \sum_{i=1}^{\infty} c_i \, Q((\frac{{}_1d_i^{\,2} \, E}{N_o})^{1/2}) \tag{12}$$

where i is an index on the set of unmerged difference sequences, ${}_1d_i^{\,2}$ is the distance corresponding to the i^{th} sequence conditioned by the fact that $h_n = h^{(1)}$, and c_i is a weighting factor obtained via (11).

Signal flow graphs and their transfer functions may be used to evaluate (12) numerically. The difference-state transition diagram is viewed as a flow graph with zero state split into input and output nodes. Paths are labelled with a gain of the form cD^xI^yL where c is given by $[M - |\gamma|/2]/M$, x is the distance accumulated in undergoing a given transition, and y is an error indicator ($y = 1$ if $\gamma \neq 0$ and $y = 0$ if $\gamma = 0$).

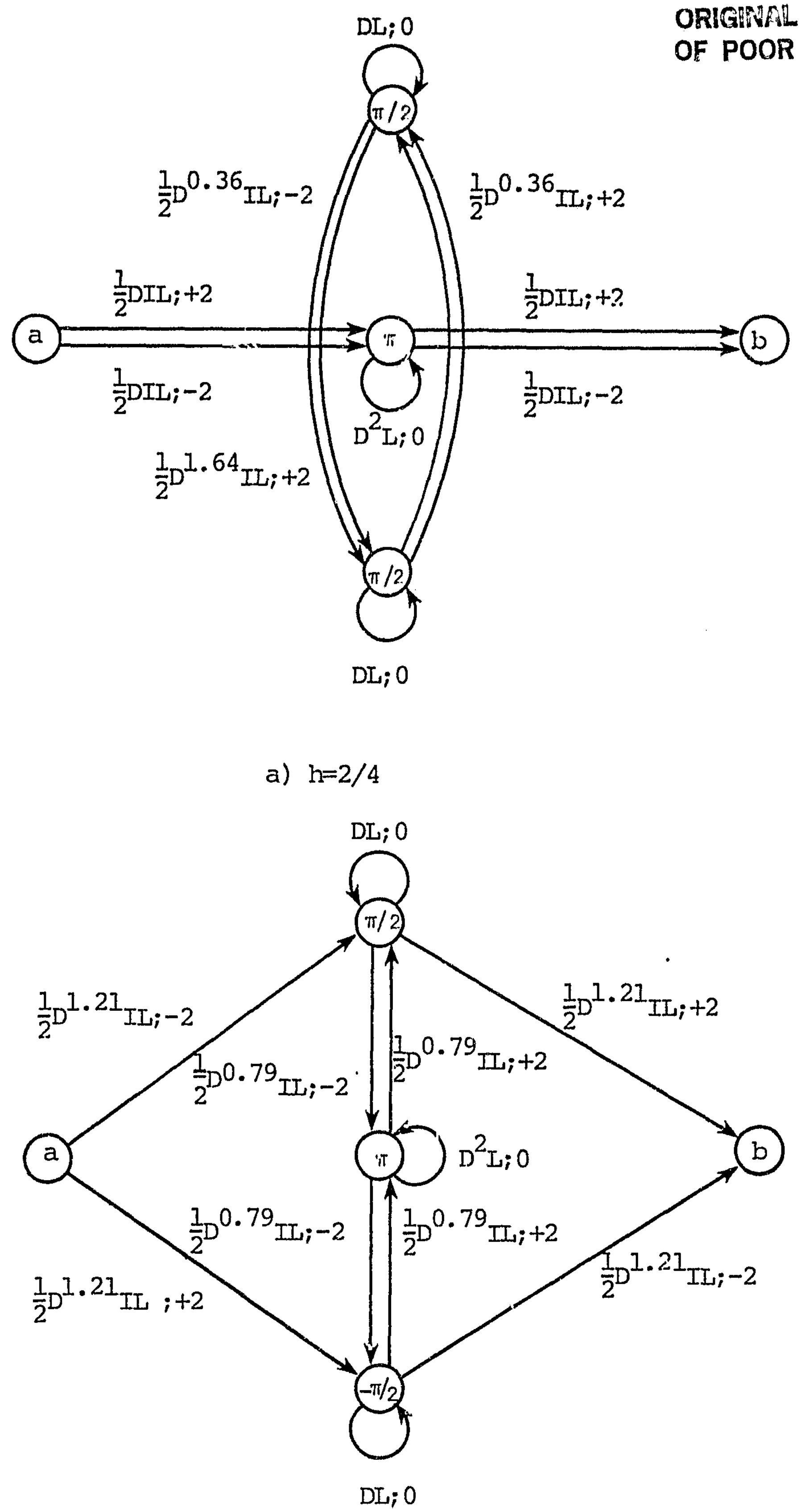
In the multi-h case, the state transition diagram is time-varying. Whereas a single transition matrix suffices in the usual case, K state transition diagrams (and matrices) are needed here. Returning to our example, we show the two split difference-state diagrams for the {2/4, 3/4} code in Figure 3. In tracing paths, one must alternate between diagrams.

To handle the time-varying case, we imagine an ensemble of transmitter/receiver pairs for which the deviation index at time $n = 1$ is randomly selected among K values equiprobably. Thereafter each system runs according to the usual multi-h format. One then sees that the error probability at any time n is an average of error probabilities conditioned on a specific choice of deviations,

$$P_e = \frac{1}{K} \sum_{j=1}^{K} P_{e|j} \qquad (13)$$

and the same holds for symbol probabilities. More practically, one may view this in terms of a time average for a single realization.

Next, we upper bound the error probability $P_{e|1}$, noting the other $P_{e|j}$ follow directly. We define, T_1 to be the matrix of path gains between difference states when deviation $h^{(1)}$ is in effect, T_2 to be the


DL; 0
π/2
$\frac{1}{2}D^{0.36}IL;-2$
$\frac{1}{2}D^{0.36}IL;+2$
$\frac{1}{2}DIL;+2$
$\frac{1}{2}DIL;+2$
a
π
b
$\frac{1}{2}DIL;-2$
$\frac{1}{2}DIL;-2$
$D^2L;0$
$\frac{1}{2}D^{1.64}IL;+2$
π/2
DL; 0
DL; 0
π/2
$\frac{1}{2}D^{1.21}IL;-2$
$\frac{1}{2}D^{1.21}IL;+2$
$\frac{1}{2}D^{0.79}IL;+2$
$\frac{1}{2}D^{0.79}IL;-2$
a
π
$D^2L;0$
b
$\frac{1}{2}D^{0.79}IL;-2$
$\frac{1}{2}D^{0.79}IL;+2$
$\frac{1}{2}D^{1.21}IL;-2$
$\frac{1}{2}D^{1.21}IL\ ;+2$
−π/2
DL; 0

a) h=2/4

b) h=3/4

Figure 3. Split Difference-State Transition Diagram, Binary {2/4,3/4} Linear Phase Code

$$T_1 = \begin{bmatrix} 0 & 0 & DIL & 0 & 0 \\ 0 & DL & 0 & \frac{1}{2}IL(D^{0.36}+D^{1.64}) & 0 \\ 0 & 0 & D^2L & 0 & DIL \\ 0 & \frac{1}{2}IL(D^{0.36}+D^{1.64}) & 0 & DL & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$T_2 = \begin{bmatrix} 0 & \frac{1}{2}D^{1.21}IL & 0 & \frac{1}{2}D^{1.21}IL & 0 \\ 0 & DL & \frac{1}{2}D^{.79}IL & 0 & \frac{1}{2}D^{1.21}IL \\ 0 & \frac{1}{2}D^{0.79}IL & D^2L & \frac{1}{2}D^{0.79}IL & 0 \\ 0 & 0 & \frac{1}{2}D^{0.79}IL & DL & \frac{1}{2}D^{1.21}IL \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 3 (continued)

matrix corresponding to $h^{(2)}$, and so on. Each matrix will be square with dimension q + 1, arising from q difference states plus one for splitting of the zero-difference state. We index the difference states 1, 2, ..., q, and let 1 and q + 1 be the indices for the split zero state. The jk^{th} element of T_1 represents the path gain from state j to state k, and will be a polynomial in D, I and L. Figure 3 itemizes these matrices for the example code.

In finding the transfer function of the graph, we note that T_1 represents the possible actions after a single step, T_1T_2 represents the actions after two steps, etc. After any prescribed number of steps, the upper right-hand corner element of the sum matrix

$$M_1 \triangleq T_1 + T_1T_2 + T_1T_2T_3 + \dots . \tag{14}$$

provides the path enumeration we seek, namely the set of possible path distances, their lengths, and respective number of symbol errors.

In the case of unlimited memory, we have an infinite sum of matrix products in (14). Noting the periodic nature of the h_n, (14) may be regrouped as

$$\begin{aligned} M_1 &= [\sum_{j=0}^{\infty} (T_1T_2 \dots T_K)^j] \, [T_1 + T_1T_2 + \dots \quad T_1T_2 \dots T_K] \\ &= [I - T_1T_2 \dots T_K]^{-1} \, [T_1 + T_1T_2 + \dots T_1T_2 \dots T_K] \end{aligned} \tag{15}$$

We call the upper-right corner element of M_1 the transfer function of the graph, when $h^{(1)}$ is the initial deviation, and denote it $G_1(D,I,L)$. It may be written in general as

$$G_1(D,I,L) = \sum_{i=1}^{\infty} c_i \, D^{d_i^2} \, I^{v_i} L^{\ell_i} \tag{16}$$

where c_i is a positive constant, d_i^2 is a distance associated with an error event of length ℓ_i, v_i is the corresponding number of symbol errors. There may be many error events of a given length having differing d_i^2 and v_i.

Recalling the upper bound to event error probability of (12) we see that if the series expansion (16) could be obtained, the bound is obtained. Such is not possible except for simple examples, so we seek an efficient numerical procedure.

By the inequality [5]

$$Q\left((a+b)^{1/2}\right) < Q(a^{1/2})e^{-b/2} \qquad a,\ b > 0 \tag{17}$$

we may write (12) as

$$P_{e|1} < Q\left(\left(\frac{{}_1d^2_{\min} E}{N_o}\right)^{1/2}\right) \exp\left[\frac{{}_1d^2_{\min} E}{2N_o}\right] \sum_{i=1}^{\infty} c_i \exp\left(-\frac{d_i^2 E}{2N_o}\right) \tag{18}$$

But the summation in (18) is precisely $G_1(D,I,L)$ with $D = e^{-E/2N_o}$, $I = 1$, $L = 1$. Thus, conditioned on $h^{(1)}$ being n^{th} deviation, we have

$$P_{e|1} < Q\left(\left(\frac{{}_1d^2_{\min} E}{N_o}\right)^{1/2}\right) \exp\left(\frac{{}_1d^2_{\min} E}{2N_o}\right) \left. G_1(D,I,L) \right|_{\substack{D = e^{-E/2N_o} \\ I = L = 1}}$$

$$< \left. G_1(D,I,L) \right|_{\substack{D = e^{-E/2N_o} \\ I = L = 1}} \Big/ \left(\frac{2\pi \, {}_1d^2_{\min} E}{N_o}\right)^{1/2} \tag{19}$$

The latter follows by the inequality

$$Q(x) < e^{-x^2/2}/(2\pi)^{1/2}x \tag{20}$$

which weakens the bound slightly for small signal-to-noise ratio.

Furthermore, the conditional upper bound on symbol error probability is

$$P_{s|1} < \sum_i c_i \, v_i \, Q\left(\left(\frac{{}_1d_i^{\,2} E}{N_o}\right)^{1/2}\right) \tag{21}$$

where v_i is the number of symbol errors associated with the ith error event. Using steps as above, this becomes

$$P_{s|1} < \frac{\partial G_1(D,I,L)}{\partial I}\Bigg|_{\substack{D = e^{-E/2N_o} \\ I = L = 1}} \Bigg/ \left(2\pi \, \frac{{}_1d^2_{min} \, E}{N_o}\right)^{1/2} \tag{22}$$

The desired derivative may be numerically evaluated using finite differences.

To complete the analysis we simply must average these bounds with respect to the K possible values for $h^{(i)}$, as in (13), and the final upper bounds become

$$P_e < \frac{1}{K} \sum_{i=1}^{K} \left(2\pi \, \frac{{}_id^2_{min} \, E}{N_o}\right)^{-1/2} \quad G_i(D,I,L)\Bigg|_{\substack{D = e^{-E/2N_o} \\ I = L = 1}} \tag{23a}$$

$$P_s < \frac{1}{K} \sum_{i} \left(2\pi \, \frac{{}_id^2_{min} \, E}{N_o}\right)^{-1/2} \quad \frac{\partial G_i(D,I,L)}{\partial I}\Bigg|_{\substack{D = e^{-E/2N_o} \\ I = L = 1}} \tag{23b}$$

Lower Bounds

Although upper bounds on performance are generally of most interest, and in fact it is known that the upper bounds derived here are asymptotically tight for large E/N_o, lower bounds may be of interest and are easily calculated. One principal use is in assessing the region for which the upper bound is tight.

A lower bound on error event probability is provided by computing the probability that <u>any</u> error sequence diverging at level n has higher metric (correlation) than the transmitted sequence, and we can make the lower bound tightest by choosing that neighbor sequence having the minimum distance. As before, there are K equiprobable selections for the n^{th} deviation, and a simple averaging of conditional lower bounds provides the desired lower bound. The minimum distance depends on the initial deviation. This minimum distance may generally be found by listing short remerger events of length L defined by

$$\sum_{n=1}^{L} h_n \gamma_n = 0, \bmod 2\pi \tag{24}$$

and' evaluating the distance from the difference state transition diagrams. In complicated cases, d^2_{min} may be found using dynamic programming on the difference-state trellis. (This is the same d_{min} needed for the upper bound).

These lower bounds to error event and symbol error probabilities become

$$P_e > \frac{1}{K} \sum_{i=1}^{K} c_i \; Q \left(\left(\frac{{}_i d^2_{min} \, E}{N_o} \right)^{1/2} \right) \tag{25a}$$

$$P_s > \frac{1}{K} \sum_{i=1}^{K} v_i \; c_i \; Q \left(\left(\frac{{}_i d^2_{min} \, E}{N_o} \right)^{1/2} \right) \tag{25b}$$

where v_i is the number of symbol errors on the various minimum distance paths. The bounds of (25) use only the minimum distance event for each possible starting deviation, and one finds the bound is weak at low signal-to-noise ratios, particularly so when several error events have comparable distances.

IV. UPPER BOUND FOR FINITE MEMORY

The infinite path memory assumption of the previous section is of course not realizable in practice, although one expects that if the decoder delay is suitably large, then ideal performance is closely approached. Since we would like to operate with minimal delay and memory, questions arise about the effect of a given delay on performance.

We assume the decoder operates as follows. The Viterbi algorithm performs trellis computations as usual, but maintains survivor paths in memory for only N symbols. Thus at level n + N of the trellis, the path with the currently best metric is located, and the oldest symbol on its path map, corresponding to $\hat{a}_n$ is released by the decoder. Trellis pruning is not performed if survivor paths disagree in the oldest position. Relative to the case of pruning of such paths, P_s actually is smaller, and the analysis is simplified.

We classify error events into two classes: those made by an ideal ML decoder operating with infinite memory; and those due to truncation wherein an unmerged path has highest metric at the current time, and the oldest symbol in its history is released by the decoder. Following Hemmati and Costello [7], we sum the error probabilities for each type as an upper bound on overall error probability. This allows some double-counting of error events.

Consider again the difference-state trellis, where the all-zero sequence corresponds to error-free operation. An error event due to truncation can occur at time n whenever a path splits from the zero difference-state at or before time n and remains unmerged at time n + N. With the aid of Figure 4, we illustrate error events of both types.

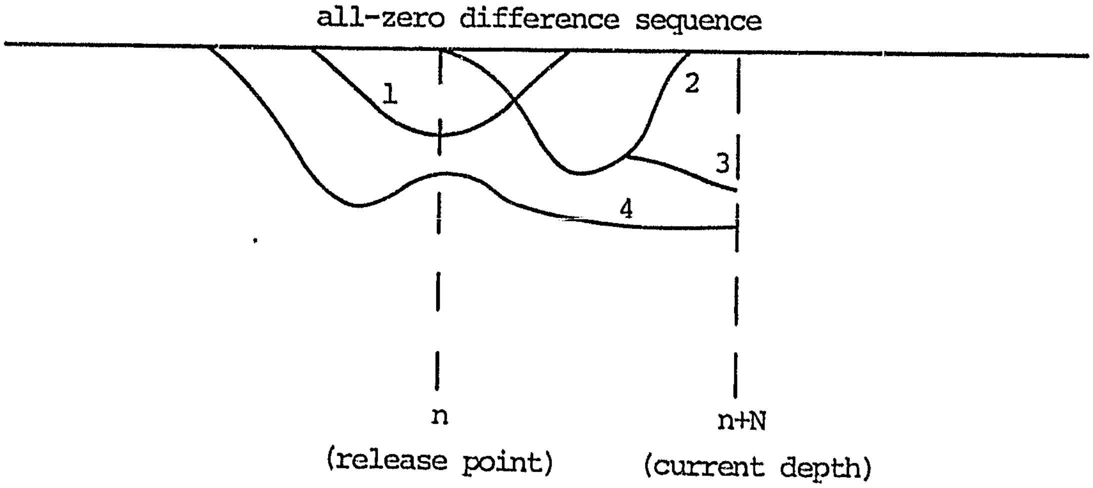


Figure 4. Schematic Illustrations of Error Event Types for Decoding with Delay N Symbols.

Events 1 and 2 are of the type which could produce symbol errors at time n in ML decoding, and were included in the analysis of Section III. Event 3 diverges at time n and remains unmerged at time n + N, so it leads to a truncation error at time n. Finally, path 4 is a sequence which diverged earlier, remains unmerged at n + N, and may produce a symbol error at time n.

We assume for notational convenience that the decoder truncation depth N is a multiple of the cycle length K, i.e. N = rK. Since multi-h codes have apparently little theoretical or practical benefit for K > 4, this represents no real instrumentation constraint.

As before we analyze performance for a specific deviation condition, then average with respect to the K such choices. We first let $h^{(1)}$ be in effect at the time of interest, n. We wish to enumerate all difference sequences which split from the zero difference-state at time n or earlier, and which remain unmerged at time n + N. Consider the matrix $(T_1T_2 \cdot\cdot\, T_K)^r$. This enumerates paths of length N = rK, and the elements in the top row, except the upper-right corner element, constitute the unmerged paths which split from the zero state at time n. Similarly, $T_K(T_1T_2 \cdot\cdot\, T_K)^r$ describes paths of length N + 1 diverging at time n - 1. Thus we sum matrix products of order N or larger, and define this sum to be $\tilde{M}_1$:

$$\begin{aligned}\tilde{M}_1 &= [\, I + T_K + T_{K-1}T_K + \dots + T_1T_2 \cdot\cdot\, T_K + \dots] \; [T_1T_2 \cdot\cdot T_K]^r \\ &= [\, I + T_K + T_{K-1}T_K + \dots + T_2T_3 \cdot\cdot T_K] \; [\, I - T_1T_2 \cdot\cdot T_K]^{-1} [T_1T_2 \cdot\cdot T_K]^r \end{aligned} \qquad (26)$$

Finally we let $\tilde{G}_1(D,I,L)$ be the polynomial sum of all top-row entries in $\tilde{M}_1$, excepting the corner element, and term this the transfer function for truncation error. Following the development of Section III, we have that, conditioned upon $h^{(1)}$ being in effect at time n, the symbol error probability due to truncation is upper-bounded as

$$P_{s_{trunc}|1} < \tilde{G}_1(D,I,L) \;/\; (2\pi\,{}_1d_{min}\,E^{2/N_o})^{1/2}\Bigg|_{\substack{D = e^{-E/2N_o} \\ I = L = 1}} \qquad (27)$$

where ${}_1d^2_{min}$ refers to the minimum distance among all unmerged difference sequences of length N or larger, with $h^{(1)}$ in effect at time n. Note the above is a symbol error probability and does not require differentiation as in the infinite memory case. This reflects the fact that only a single error is released when a truncation error event occurs.

It remains to formulate the bound when the other K-1 deviations are in effect at time n. The matrices $\tilde{M}_2$ through $\tilde{M}_K$ are similar in form to that of (26), except the subscripts are cycled in straight-forward manner. Then similar bounds to that of (27) are found, each using a different d^2_{min}. Adding all such truncation probabilities and dividing by K provides an upper bound on symbol error probability due to truncation, $P_{s_{trunc}}$. As earlier argued the final upper-bound is obtained by combining the bound for infinte memory with $P_{s_{trunc}}$:

$$P_{s_{total}} < P_{s_{ML}} + P_{s_{trunc}} \qquad (23)$$

where $P_{s_{ML}}$ is given by (25b).

V. NUMERICAL EXAMPLES

The preceding development has been applied to the earlier example, as well as to two other codes: the binary {4/8, 5/8, 6/8} code and a 4-ary {3/16, 4/16} code. The free distance calculations for these designs project an asymptotic energy efficiency relative to PSK or QPSK, of 1.4 dB, 2.8 dB, and -0.8 dB respectively. The first two have spectra comparable to that of CPFSK with $h = 5/8$, and the 99% bandwidth is approximately 1.67 times the bit rate. The 4-ary code has a lesser bandwidth occupancy (the most rapid phase change is $3\pi/4$ radians per two bits, or that of binary signalling with $h = 3/8$. Its 99% bandwidth is approximately 0.7 times the bit rate, [8].

Figure 5 illustrates the bounds for the {2/4, 3/4} code, for unlimited path memory and with $N = 2$ and $N = 4$. For unlimited memory, the upper and lower bounds closely bracket the true performance for typical levels of interest, say $P_s < 10^{-3}$. Simulation results for unlimited memory actually lie close to the upper bound. All simulation results include at least 100 symbol errors.

The decision depth (the depth in the trellis beyond which all unmerged paths have distance exceeding d_{free}) is four for this code. Thus we expect $N = 4$ should be quite adequate as a practical path memory length. Indeed the upper bound is only 0.2 dB displaced from the unlimited memory bound. Simulation results again confirm the validity of the bound. With $N = 2$, d_{min} is appreciably less, and in fact the efficiency is worse than that of PSK. This case indicates the effect of a poor choice of N.

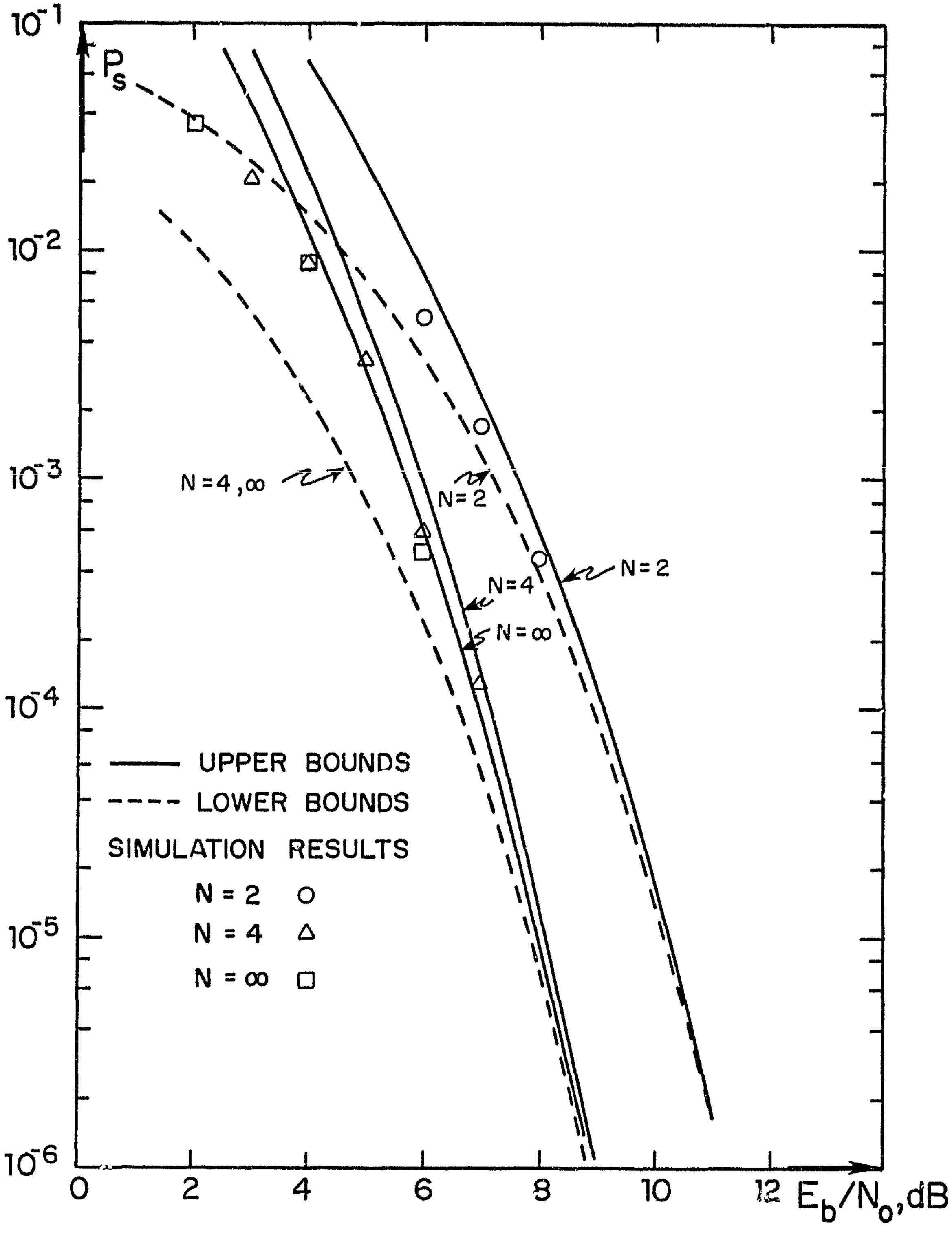


Figure 5. Upper Bounds and Lower Bounds to P_s for {2/4, 3/4} Code

Figure 6 presents upper-bound results for the K = 3 {4/8, 5/8, 6/8} code. The larger asymptotic coding gain is clearly revealed, and we also note this stronger code requires increased delay to realize its full potential. This is completely analogous to results for convolutional codes, where increasing constraint length necessitates larger decoder memory. The decision depth in this case is nine bits, and the upper bound for N = 8 is asymptotically different than the unlimited memory bound.

Finally, results for a narrow-band 4-ary design are shown in Figure 7. The decision depth in this case is $N_D = 9$ symbols. Again note that eight symbols of path memory is essentially adequate to achieve unlimited-memory performance. The asymptotic efficiency is 0.8 dB poorer than PSK/QPSK; however the spectrum is quite compact. 4-ary multi-h signalling appears to offer extra gain in the power/bandwidth/complexity trade-off, [9], [10], as is the case in other modulations such as CPFSK and partial-response FM.

ORIGINAL PAGE IS
OF POOR QUALITY

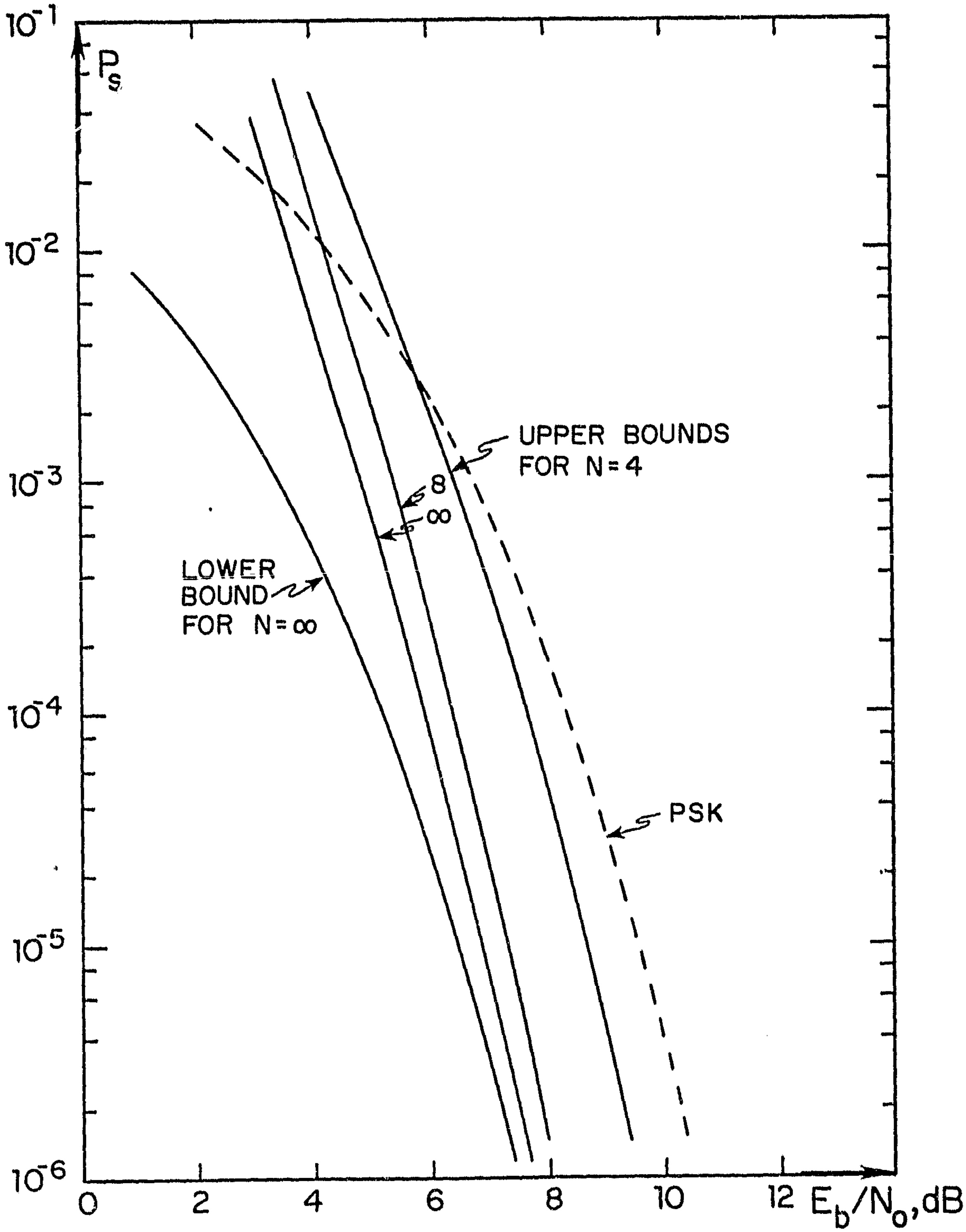


Figure 6. Error Bounds for Binary {4/8, 5/8, 6/8} Code

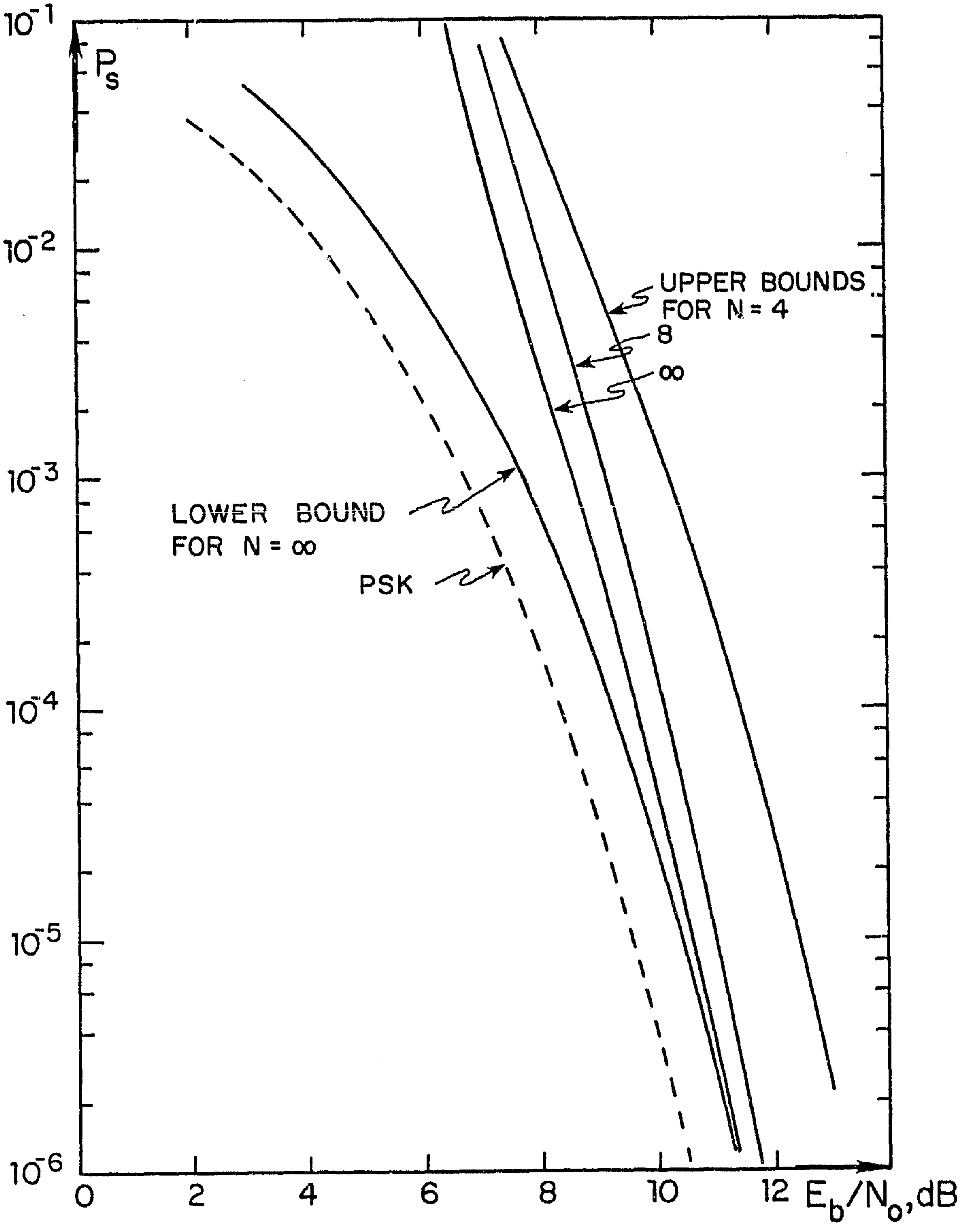


Figure 7. Error Bounds for 4-ary {3/16, 4/16} Code

REFERENCES

1. Anderson, J. B., and Taylor, D. P., "A New Class of Signal Space Codes," IEEE Transactions on Information Theory, pp. 703-712, November 1978.

2. Miyakawa, H., Harashima, H. and Tanaka, Y., "A New Digital Modulation Scheme - multi-mode CPFSK," Proceedings 3rd Int'l Conference on Digital Satellite Communications, Kyoto, Japan, 1975.

3. Anderson, J. B., "Simulated Error Performance of Multi-h Phase Codes," CRL Report 67, Communications Research Laboratory, McMaster University, Hamilton, Ontario; see also Int'l Communications Conference Record, June 1980.

4. Wilson, S. G., Highfill, J. H., Gaus, R., and Hsu, C-D., "Trellis Phase Codes for Improved Power and Bandwidth Efficiency in Satellite Communication," report to NASA/Lewis Research Center, contract NAS5-25634, July 1980.

5. Viterbi, A. J. and Omura, J., Principles of Digital Communication and Coding, McGraw-Hill, 1979.

6. Aulin, T., "Continuous Phase Modulation," Ph.D. dissertation, University of Lund, Lund, Sweden, 1979, see also National Telecommunications Conference Record, December 1980.

7. Hemmati, F. and Costello, D. J., "Truncation Error Probability in Viterbi Decoding," IEEE Transactions on Communications, pp. 530-532, May 1977.

8. Wilson, S. G. and Gaus, R., "Power Spectra of Multi-h Phase Codes," IEEE Transactions on Communications, pp. 250-256, March 1981.

9. Hsu, C-D., "Multi-h Phase Coding: Its Theory and Design," Ph.D. dissertation, University of Virginia, 1981.

10. Aulin, T. and Sundberg, C-E., "On the Minimum Euclidean Distance for a Class of Signal Space Codes," submitted to IEEE Transactions on Information Theory.

3.0 4-ARY MULTI-H CODES

By allowing non-binary signalling, we allow more freedom in the signal design and thus more flexibility in the power/bandwidth/complexity tradeoff. In more traditional designs, it is known that M-ary orthogonal signalling provides power savings over binary signalling, while M-ary PSK conserves in bandwidth. More recently, it has been found that non-binary partial response FM modulation affords substantial improvement in power-versus bandwidth performance, [1]. 4-ary and binary CPFSK are also known to be superior to binary CPFSK.

We have surveyed 4-ary multi-h phase codes with rectangular and raised-cosine frequency pulse shaping, [2]. It happens that to have the full constraint length of K + 1 symbols for a multi-h code with K deviations, that q, the common denominator of the rational deviation indexes must be greater or equal to M^K. This is also the number of states in a maximum likelihood decoder. Consequently complexity grows rather rapidly with M, and we chose to stop at 4-ary, 2-h codes, where $q = 16$.

Table 3.1 lists a selection of such codes having full constraint length, along with their free distance values. As for binary codes, the best codes are those with h's clustered near a common value, e.g. {3/16, 4/16}. Figure 3.1 illustrates the performance graphically as a function of $\bar{h}$, the mean index. Note that for small $\bar{h}$, raised-cosine codes have slightly better distance (and slightly wider main lobe in the spectrum) and that a gain of about 2 dB over 4-ary CPFSK is typical. This corresponds to the typical gains shown for binary 2-h codes over binary CPFSK.

Table 3.1 Free distance of 4-ary 2-h codes with constraint length 3.

TYPE			MEAN	LINEAR PHASE CODES		R.C. FREQUENCY CODES	
Q	P_1	P_2	INDEX	d^2_{free}	Decision Depth, T_b	d^2_{free}	Decision Depth, T_b
16	4	3	.219	✓ 1.688	18	✓ 2.103	20
16	5	4	.281	✓ 2.789	24	2.654	22
16	7	4	.344	✓ 3.454*	30	✓ 3.217*	20
17	5	3	.235	1.776	22	2.008	20
18	8	7	.417	✓ 4.527*	34	3.249*	18
18	11	8	.528	4.076	12	3.249	10
19	8	6	.368	4.262	42	2.560*	28
20	4	3	.175	1.105	18	1.387	20
20	6	5	.275	2.774	34	3.095	36
20	8	5	.325	3.399	30	✓ 3.549*	42
20	9	8	.425	✓ 4.798*	44	3.095	12
21	10	8	.429	4.286*	52	2.517*	32
22	7	6	.295	3.226	46	2.499	34
24	5	4	.187	1.323	26	1.633	30
24	6	5	.229	1.998	36	2.419	42
24	7	6	.271	2.774	46	✓ 3.217*	52
25	6	5	.220	1.853	36	2.249	42
25	10	7	.340	3.656*	34	3.726*	40
26	5	4	.173	1.136	26	1.406	30
27	8	7	.278	2.962	60	3.249	62
28	8	7	.268	2.780	62	3.285	72
28	11	9	.357	4.197	88	2.424	52
28	12	11	.411	4.646*	76	3.442*	40
29	6	5	.190	1.404	36	1.715	42
29	13	12	.431	4.838*	80	3.142*	12
30	9	8	.283	3.118	78	3.095	64
30	13	12	.417	4.877*	94	3.586*	22
31	6	5	.177	1.238	36	1.515	44
32	8	7	.234	2.190	62	2.612	74
32	10	9	.297	3.423	96	2.654	40
32	20	19	.609	4.739*	50	3.579*	36

* : codes with best free distance among codes with common q.

✓ : codes of interest (having good power/bandwidth/complexity trade-off).

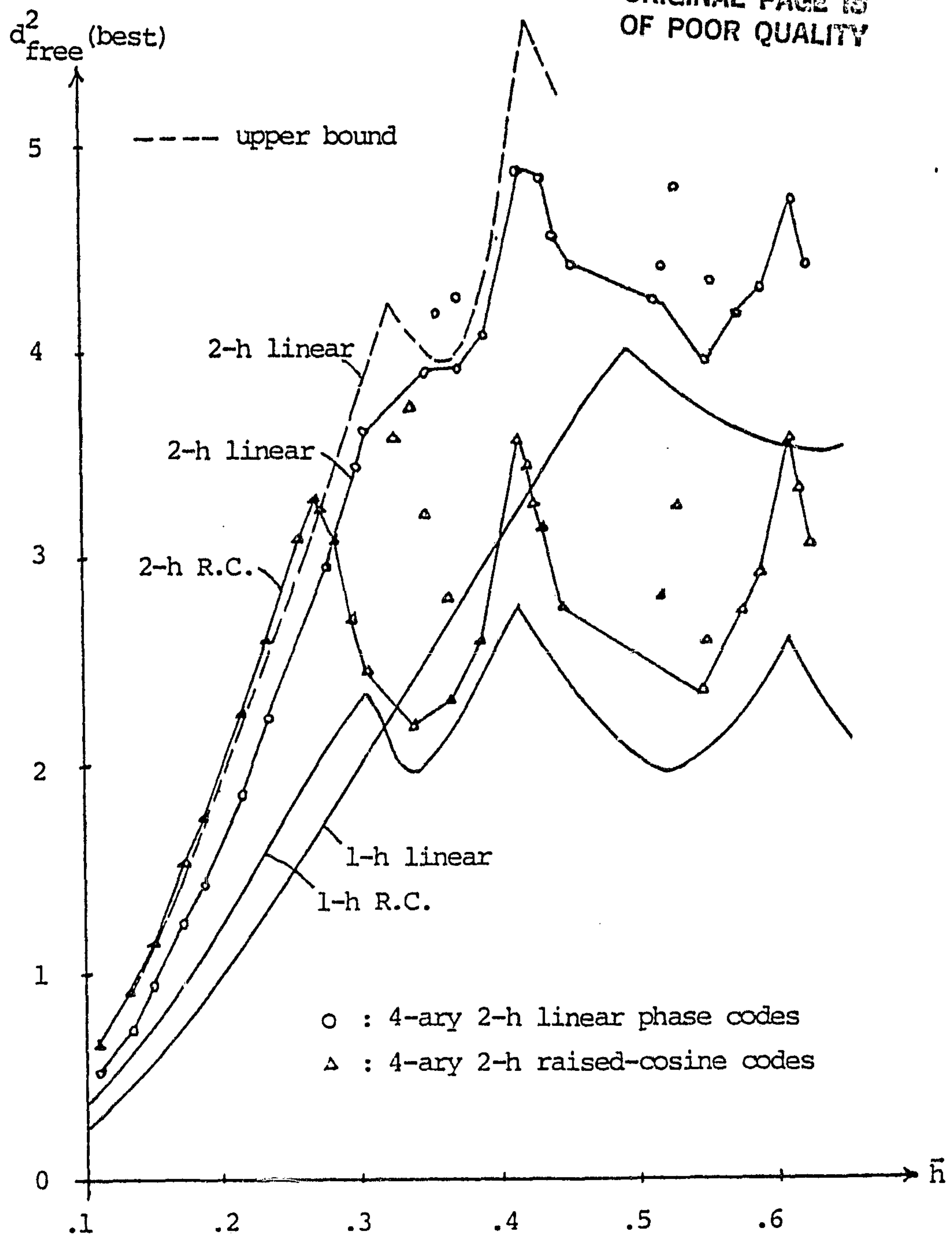


Figure 3.1 Best free distance versus $\bar{h}$. The codes are 4-ary codes with linear phase and raised-cosine pulse shaping. The unconnected marks represent codes with wide-spread indices (the difference between the numerators of the indices is greater than one).

The spectral properties of 4-ary 2-h codes may be calculated numerically using the procedure described earlier. An example is shown as Figure 3.2 for the {4/16, 5/16} case. Hsu [2] has found a quick rule of thumb for spectral behavior based on the mean index $\bar{h}$. Since the data values are {±1, ±3, ... ±M-1} and each symbol conveys log M_2 bits, we define an equivalent binary index as

$$\bar{h}_b = \frac{1}{K} \quad \frac{1}{M/2} \sum_{j=1}^{K} \sum_{i=1}^{M/2} (2i-1) \quad \frac{h_j}{\log_2 M} \qquad (3.1)$$

$$= \left(\frac{M/2}{\log_2 M} \right) \bar{h}$$

where $\bar{h}$ is the mean index, averaged over the K possible values. M-ary codes having an equivalent $\bar{h}_b$ are found to have nearly equal spectral properties, i.e. 99% power bandwidth, etc. Since $(M/2)/\log_2 M = 1$ for M = 4 and M = 2, we infer binary and 4-ary codes with equal $\bar{h}$ will have comparable bandwidth. This is illustrated in Figure 3.3, showing B_{99} and B_{90} are equal within ± 10% when codes are compared on the basis of $\bar{h}_b$.

Table 3.2 summarizes bandwidth and distance statistics for a number of multi-h codes ranging from power efficient to bandwidth efficient designs.

In summarizing the M-ary multi-h situation, we show performance in the distance-bandwidth plane, Figure 3.4. 4-ary 2-h designs show substantial gain over binary 2-h designs, particularly in the small bandwidth region, say $B_{99}\ T_b < 1$. Also, we observe the rather depressing result that 4-ary single -h codes outperform binary 2-h codes under a bandwidth constraint. Since complexity is probably no worse

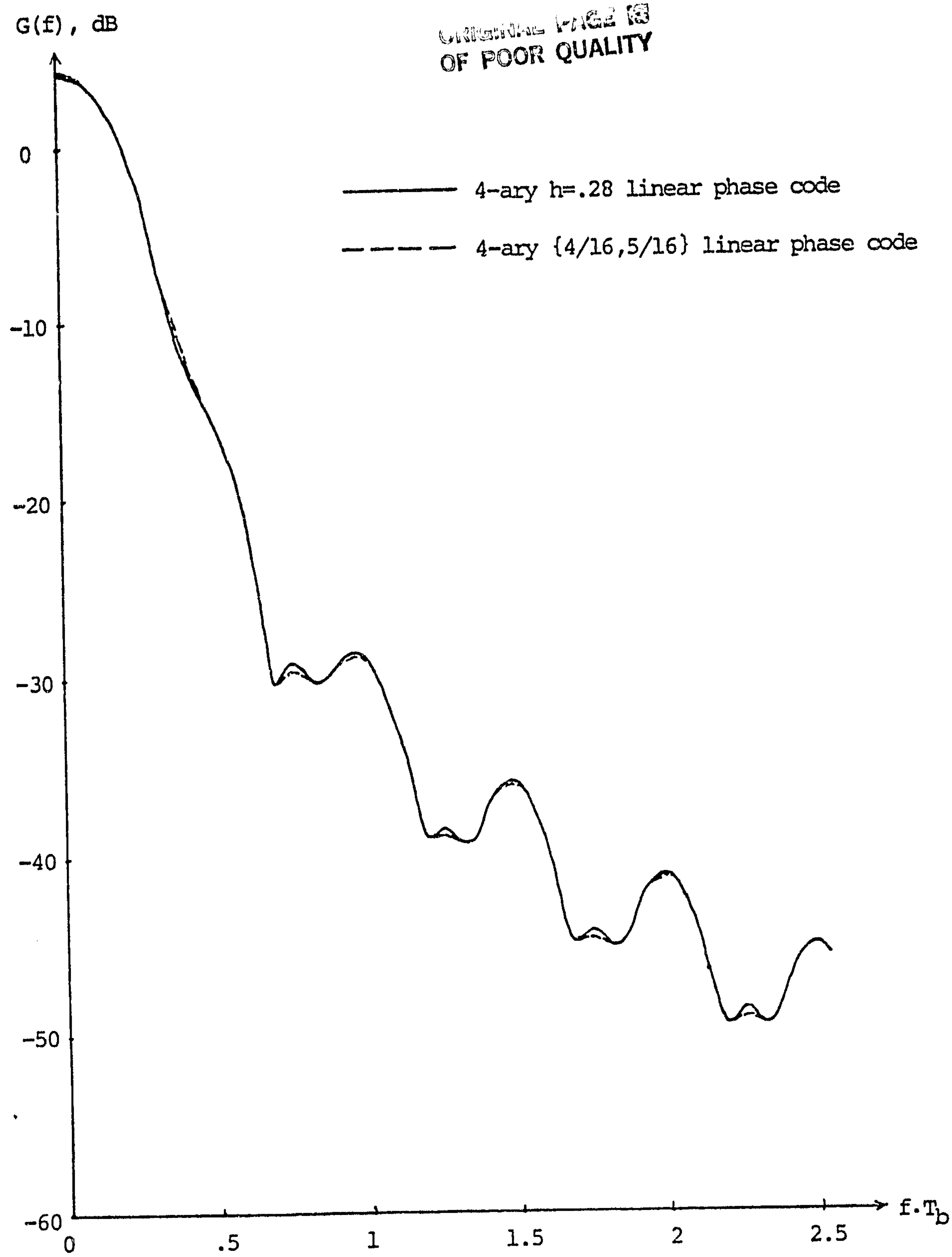


Figure 3.2 Power spectral density for 4-ary multi-h linear phase codes with the same mean index.

ORIGINAL PAGE IS
OF POOR QUALITY

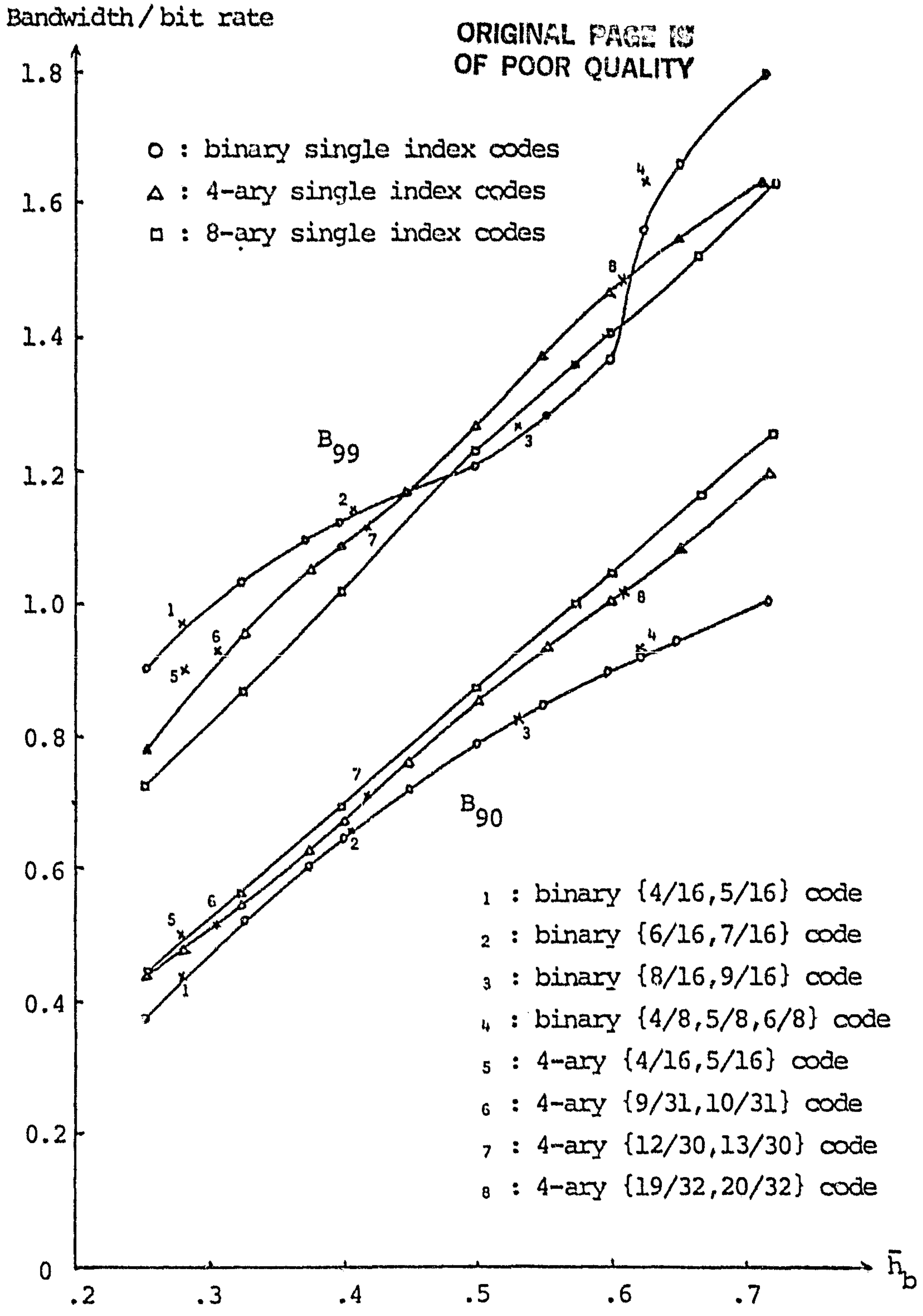


Figure 3.3 90% and 99% bandwidths versus equivalent binary mean index for various M-ary single index, linear phase codes. Also shown are some multi-h codes with comparable indices.

Table 3.2 Bandwidth performance for multi-h codes with good power/bandwidth/complexity trade-offs.

CODES		B_{99}	B_{90}	B_{rms}	mainlobe BW	35dB	50dB	coding gain, dB
ref. codes	QPSK	10.28		inf.	1.0	7.6	100.5	0
	MSK	1.20	0.79	0.50	1.5	3.2	8.2	0
power efficient codes	binary single h=5/7	1.80	1.01	0.71	1.4	4.4		0.92
	4-ary single h=1/2	1.27	0.85	0.55		3.2		3.00
	binary {4/2,3} l.p.	1.68	0.95	0.64	1.5	4.3	10.2	1.44
	binary {4/2,3} r.c.	2.60	1.10	0.78	1.6	4.4	5.2	0.90
	binary {6/3,4} l.p.	1.42	0.88	0.58	1.4	4.2	10.2	2.37
	binary {6/3,4} r.c.	2.45	1.04	0.72	1.6	4.3	5.0	2.49
	binary {8/4,5} l.p.	1.34	0.86	0.56	1.4	4.2	10.2	2.49
	binary {8/4,5} r.c.	2.39	1.01	0.69	1.6	4.2	4.9	2.70
	binary {9/4,6} l.p.	1.40	0.87	0.56		4.1		2.32
	binary {8/4,5,6} l.p.	1.66	0.93	0.63	1.4	4.3	10.2	2.77
	binary {10/5,6,7}r.c.	2.49	1.05	0.74	1.6	4.4	5.1	2.88
bandwidth efficient codes	binary {8/2,3} r.c.	1.40	0.60	0.39	1.8	2.8	4.2	-0.38
	binary {10/3,4} r.c.	1.46	0.68	0.43	1.8	2.9	4.3	0.72
	4-ary {16/3,4} l.p.	0.70	0.39	0.24		1.5		-0.74
	4-ary {16/3,4} r.c.	1.14	0.47	0.30		1.9	2.5	0.22
power and bandwidth efficient codes	4-ary {16/4,5} l.p.	0.90	0.49	0.31		2.1	5.1	1.44
	4-ary {16/4,7} l.p.	1.09	0.61	0.39		2.3		2.37
	4-ary {16/4,7} r.c.	1.60	0.71	0.49		2.7	3.5	2.06
	4-ary {20/8,9} l.p.	1.14	0.82	0.47		3.1		3.80

ORIGINAL PAGE IS OF POOR QUALITY

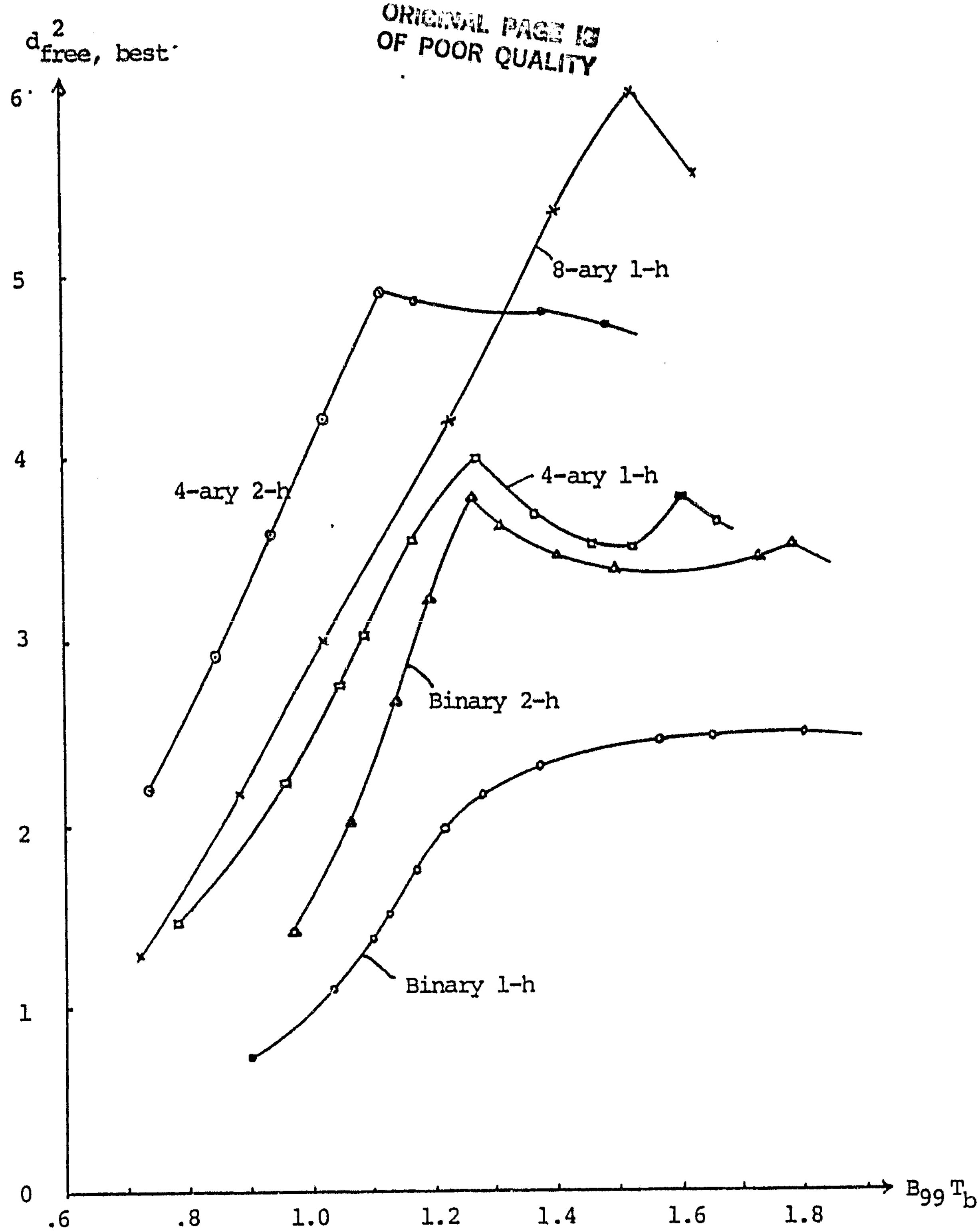


Figure 3.4 Best free distance versus 99% bandwidth for various M-ary multi-h and single index codes. The codes are of linear phase transition.

for the 4-ary single-h design, it seems questionable whether binary 2-h codes, and even binary multi-h codes in general, should be studied further. We do note that 4-ary 2-h codes outperform 8-ary CPFSK in the small bandwidth region, and that the 4-ary 2-h class thus seems a strong modulation candidate.

REFERENCES

1. Aulin, T., and Sundberg, C-E., "Continuous-Phase Modulation: Parts I and II," IEEE Transactions on Communication, March 1981.

2. Hsu, C-D., "Multi-h Phase Coding: Its Theory and Design," Ph.D. dissertation, University of Virginia, 1981.

4.0 CHANNEL SIMULATION

We have implemented a FORTRAN software simulation of a typical satellite channel for purposes of assessing various effects on performance which are induced by bandlimiting filters and/or saturating repeaters. Discrete-time simulation seems to be the expeditious approach as analytical results of sufficient generality appear formidable. The simulation package is essentially completed at this point, though no error probability production runs have been completed.

The system being addressed is shown in Figure 4.1. The software is highly-modular, so rearrangements of the functional blocks is possible; however the configuration of Figure 4.1 represents the situation with post-modulation filtering and nonlinear amplification (either at the ground station or in a satellite transponder), down-link noise being dominant, and general receiver filtering. Coherent reception with maximum likelihood decoding is also assumed. In the absence of filtering and nonlinear distortion, performance is known for a variety of partial-response FM and multi-h codes through calculations of minimum distance and error bounds. As with other more classical schemes however, the effect of channel degradation is an important engineering question. In particular it is felt by some that these newer exotic designs are less tolerant of distortion than say QPSK or MSK.

One may wonder why filtering is considered, since one of the attributes of continuous phase modulation is improved spectral shaping, with the potential for avoiding the need for post-modulation filtering. It is however likely that, even so, some filtering will be performed for one of several reasons. First, equipment with formerly-necessary

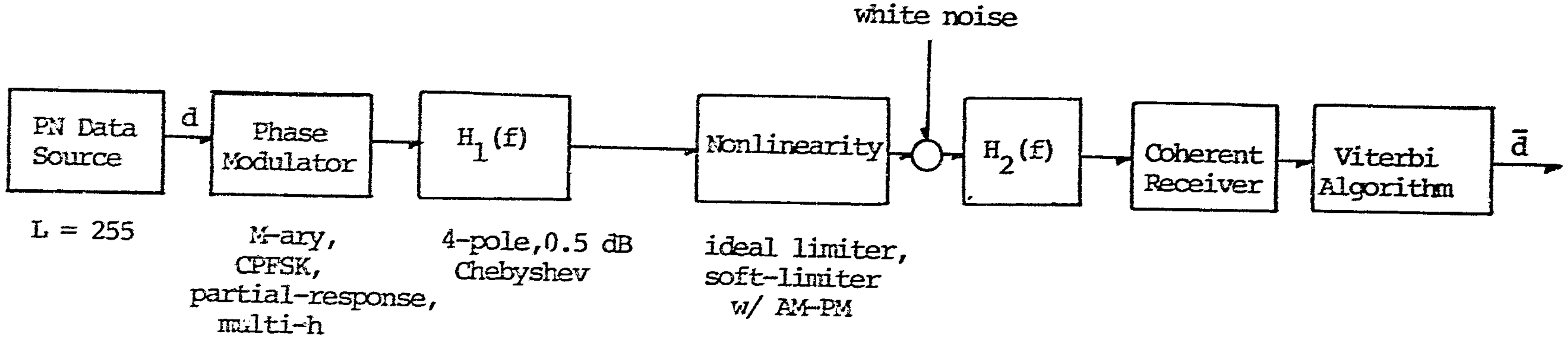


Figure 4.1 Simulation Block Diagram

channel filtering may still be in place and is to be used. Second, premodulation shaping alone is not able to attain spectral emission limits in high performance cases, e.g. trying to maximize data rate while meeting the FCC spectral mask. Post-modulation filtering to some degree is a simple expedient. Finally, aside from the above factors, the push toward high spectral efficiency will suggest moving bandwidth-efficient signals even closer in channel spacing, in essence placing even tighter specifications on spectral emission.

The simulation processes the baseband complex envelope of a bandpass signal, and operates in discrete-time with eight samples per data bit, chosen to safely avoid aliasing phenomena. The signals of interest have spectra at least 30 dB down at the Nyquist frequency, which is four times the bit rate in this case. The data source is a 255-bit maximal length shift register sequence, with an extra 0 appended to balance the sequence. With eight samples per bit, we generate 2048 complex samples as the signal sequence. These signal samples are used repeatedly with new noise sequences to generate enough trials to yield statistical significance in error probability estimates. This choice simulates much more rapidly than an approach which generates new signal samples continuously.

The length of the PN sequence must be sufficiently long however to generate all significant unique data patterns as far as the channel output is concerned. Intersymbol interference is worse for certain patterns, and all significant patterns must be generated in the correct proportion. One of the properties of maximal length sequences is that the relative frequency of different N-tuples is very close to that of a coin-flipping process. With the degree of channel filtering expected for our simulation, a L = 255 bit sequence is expected to be adequate, as the essential memory of the channel does not exceed 8 bits, and the relative frequencies are correct for up to 8-tuples.

The receiver simulation adds white Gaussian noise to the received signal and filters the sum (if desired). The noisy discrete-time process is processed by complex correlators to yield statistics for the Viterbi algorithm receiver. The system as described has no carrier phase shift, so this need not be compensated. However, there is group delay roughly proportional to 1/(composite filter bandwidth) which is on the order of one bit or more. This delay must be compensated by the correlators, and can be done once for the various configurations. Our timing quantization is $T_b/8$.

Once synchronized, the correlator outputs are passed to the ML routine having an appropriate state trellis for the modulation. After a sixteen bit delay, bit decisions are released by the decoder and compared with the actual PN sequence to compile error statistics.

Note that the receiver is optimal for AWGN environments and acts as if no channel distortions were present. Equalization could undoubtedly improve performance for severely bandlimited channels, and this could be investigated. However we feel that the initial study should be of the effect of channel distortion on performance of "conventional" receivers.

4.1 SIMULATION RESULTS

We first present results for MSK, a case for which results have recently appeared in the literature [1], [2], [3]. In Figure 4.2 we show the power spectrum at the modulator output, after filtering, and following hard limiting. The spectrum at the modulator output has side

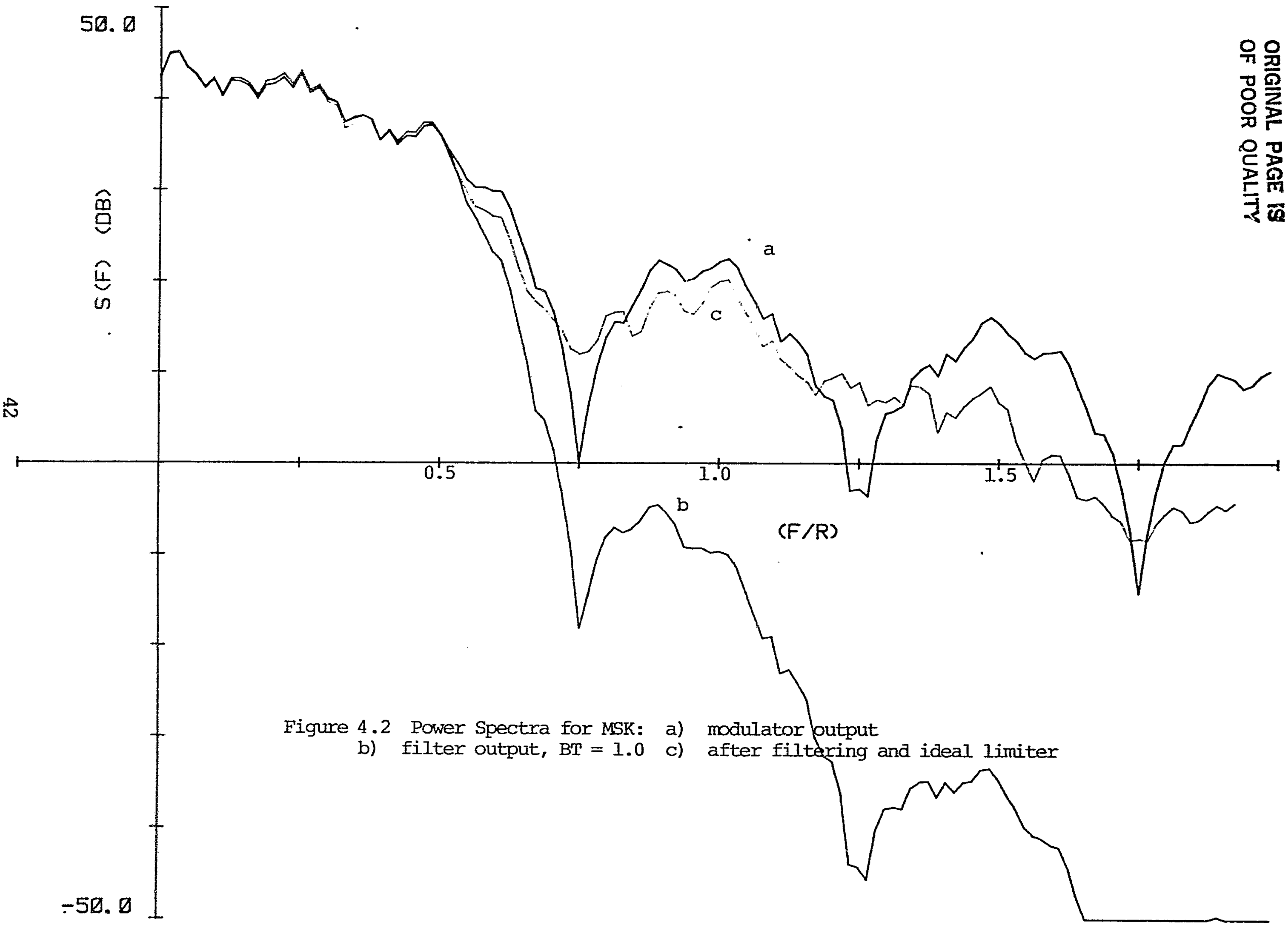


Figure 4.2 Power Spectra for MSK: a) modulator output
b) filter output, BT = 1.0 c) after filtering and ideal limiter

lobes which decay as f^{-4}, and well-defined nulls. The minor roughness in the spectrum is due to finite-averaging of a random process. The filter used was a 0.5 dB ripple, 4-pole Chebyshev characteristic with a BT product of 1.0. Note that this filtering truncates the mainlobe slightly, and obviously reduces sidelobe levels. The spectrum at this point would be comparable to QPSK with filtering at the first null in the spectrum (BT = 1). The interesting effect of ideal limiting is to regenerate spectral "sidelobes" having level almost as large as originally present. The power spectrum is smoother, and lobes have been eradicated. This is completely in accord with recent results of [1].

Figure 4.3 presents similar results on spectra for a soft-limiter modelled as an error-function characteristic for AM-AM and a 3°/dB for AM-PM. This is perhaps more representative of typical TWT characteristics. Note with 3 dB backoff, the total power, relative to that of full output, is reduced, but more importantly the spectral regeneration is less pronounced. Figure 4.4 does the same comparison with a 9 dB backoff so that the characteristic is essentially linear; however the AM-to-PM conversion effect is still present.

We also have capability for plotting amplitude and phase trajectories of the signal enroute. Figure 4.5 shows the amplitude, or envelope, of the filtered signal for filtered MSK with BT = 1. The bit sequence corresponds to 000000010111000. This plot gives an indication of the amplitude ripple produced by bandlimiting a constant-envelope signal, and in turn some notion of the severity of nonlinear amplifier distortion. Phase plots are of more interest as phase is the informa-

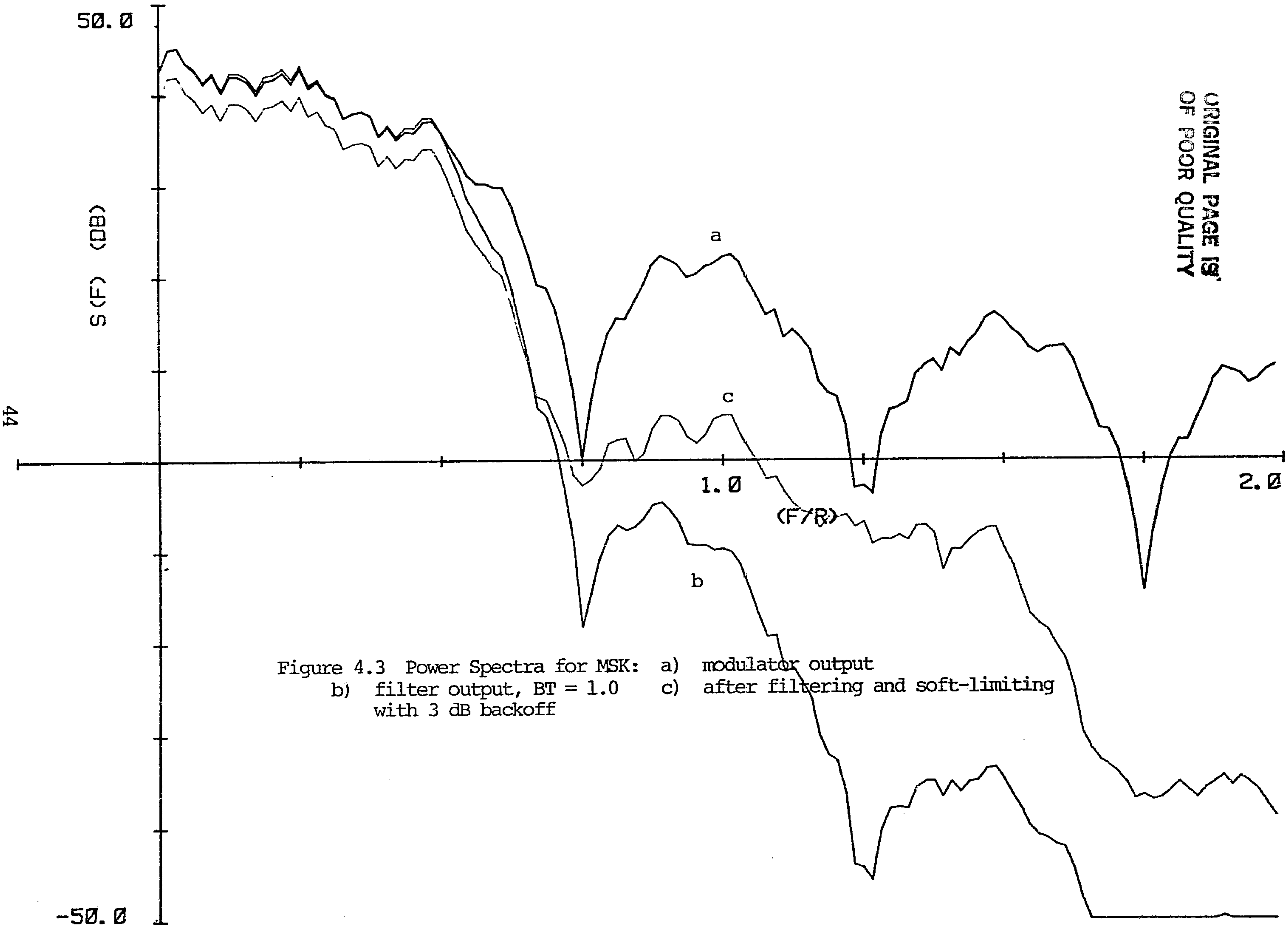


Figure 4.3 Power Spectra for MSK: a) modulator output
b) filter output, BT = 1.0 c) after filtering and soft-limiting
with 3 dB backoff

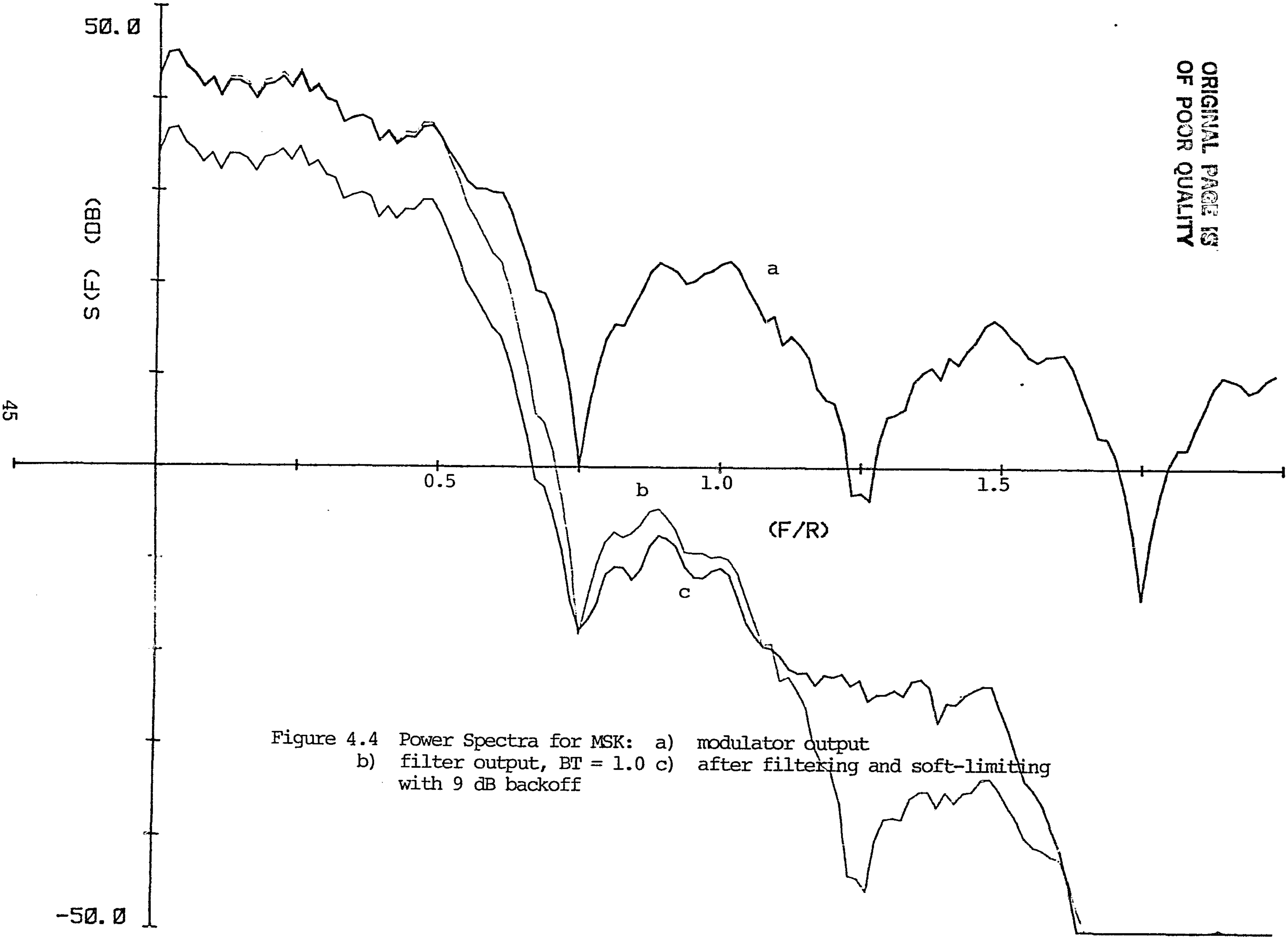


Figure 4.4 Power Spectra for MSK: a) modulator output
b) filter output, BT = 1.0 c) after filtering and soft-limiting
with 9 dB backoff

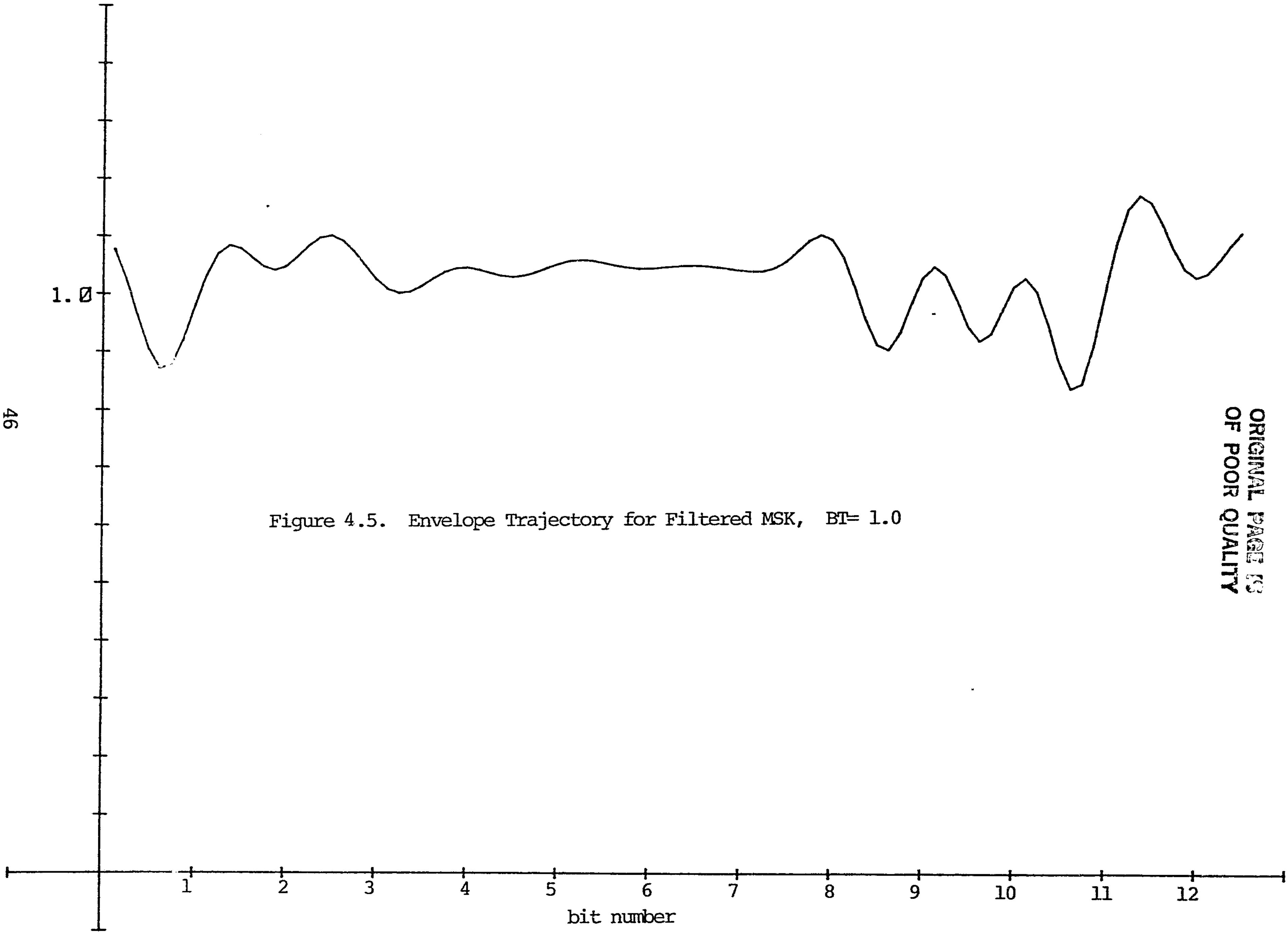


Figure 4.5. Envelope Trajectory for Filtered MSK, BT= 1.0

tion-bearing variable; however, the plots are currently difficult to interpret because of modulo-2π restrictions in the angle definition.

Spectral results for double raised-cosine (DRC) with $h = 1/4$, channel filtering with $BT = 0.5$, and hard limiting are shown in Figure 4.6. This design exhibits high bandwidth efficiency with $B_{99}\ T \cong 0.7$, and sidelobes roll-off as f^{-8}. We again note that limiting reconstitutes spectral sidelobes to a large degree. Figure 4.7 repeats the analysis for 9 dB back-off as above.

A similar set of plots is shown in Figures 4.8 and 4.9 for 4-ary multi-h coding with deviation indices 4/16 and 5/16. The coding gain is 1.4 dB and the $B_{99}\ T_b = 0.9$, 25% less than for MSK.

The sidelobe reconstitution is an interesting phenomenon which has received considerable attention for QPSK and offset QPSK. Basically, when a signal is hard-limited, the rather smooth trajectories of the filter output are "sharpened", moving a relatively larger percentage of power to higher frequencies. (Only the phase trajectory varies at the output of the limiter). However the phase trajectories remain smoother than at the modulator output, so sidelobe levels are still reduced by filtering, then limiting. We have shown that the asymptotic rate of decay at the limiter output is equal that at the filter output; however, asymptotic rates tell little of absolute levels or where the asymptotic rate pertains. We expect to study this further under the next contract (NAG3-141).

4.2 SOFTWARE DESCRIPTION

The complete listing of the software modules is found in Appendix A. Program SIMUL1 simulates a digital communication system with

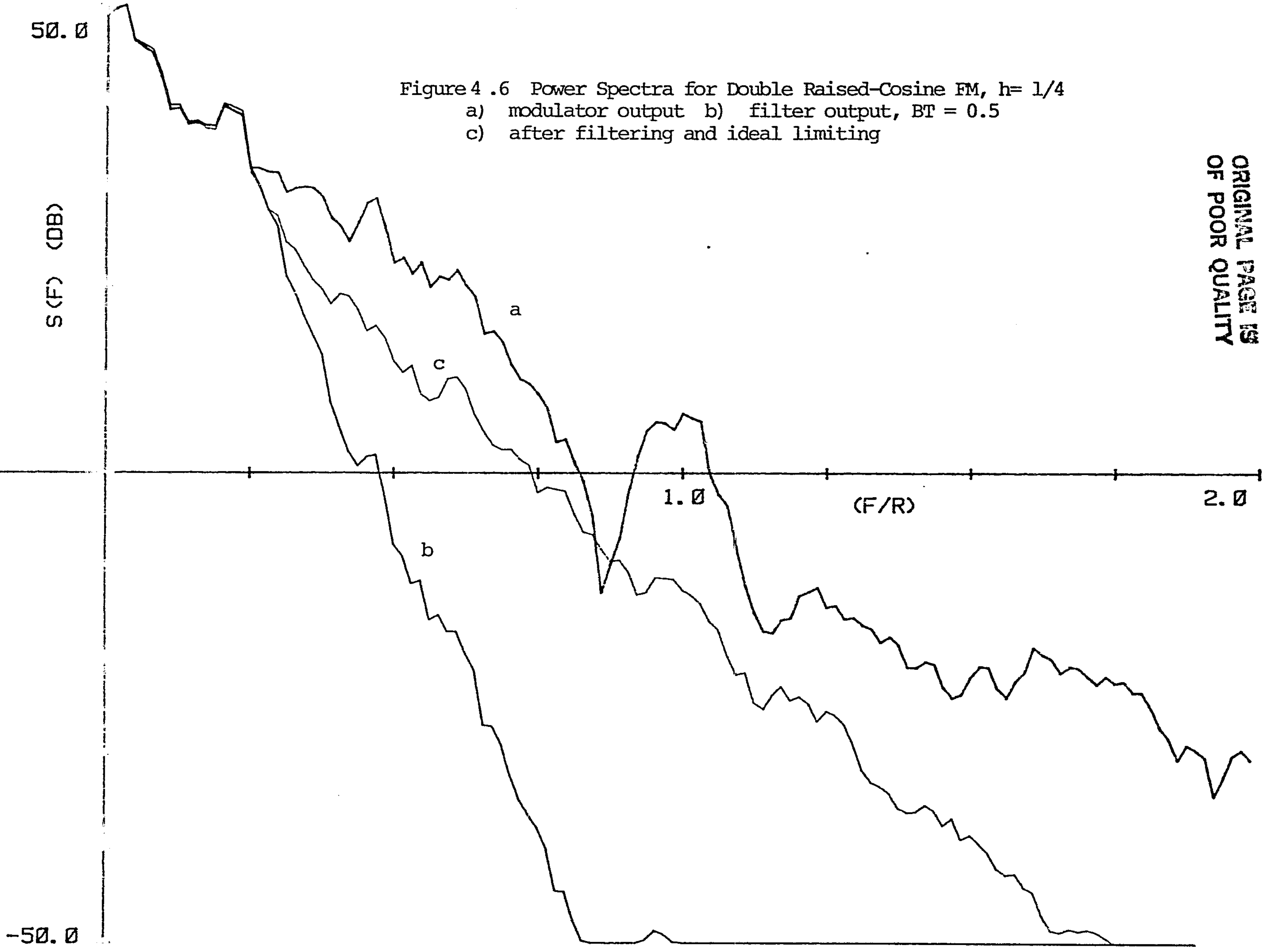


Figure 4 .6 Power Spectra for Double Raised-Cosine FM, h= 1/4
a) modulator output b) filter output, BT = 0.5
c) after filtering and ideal limiting

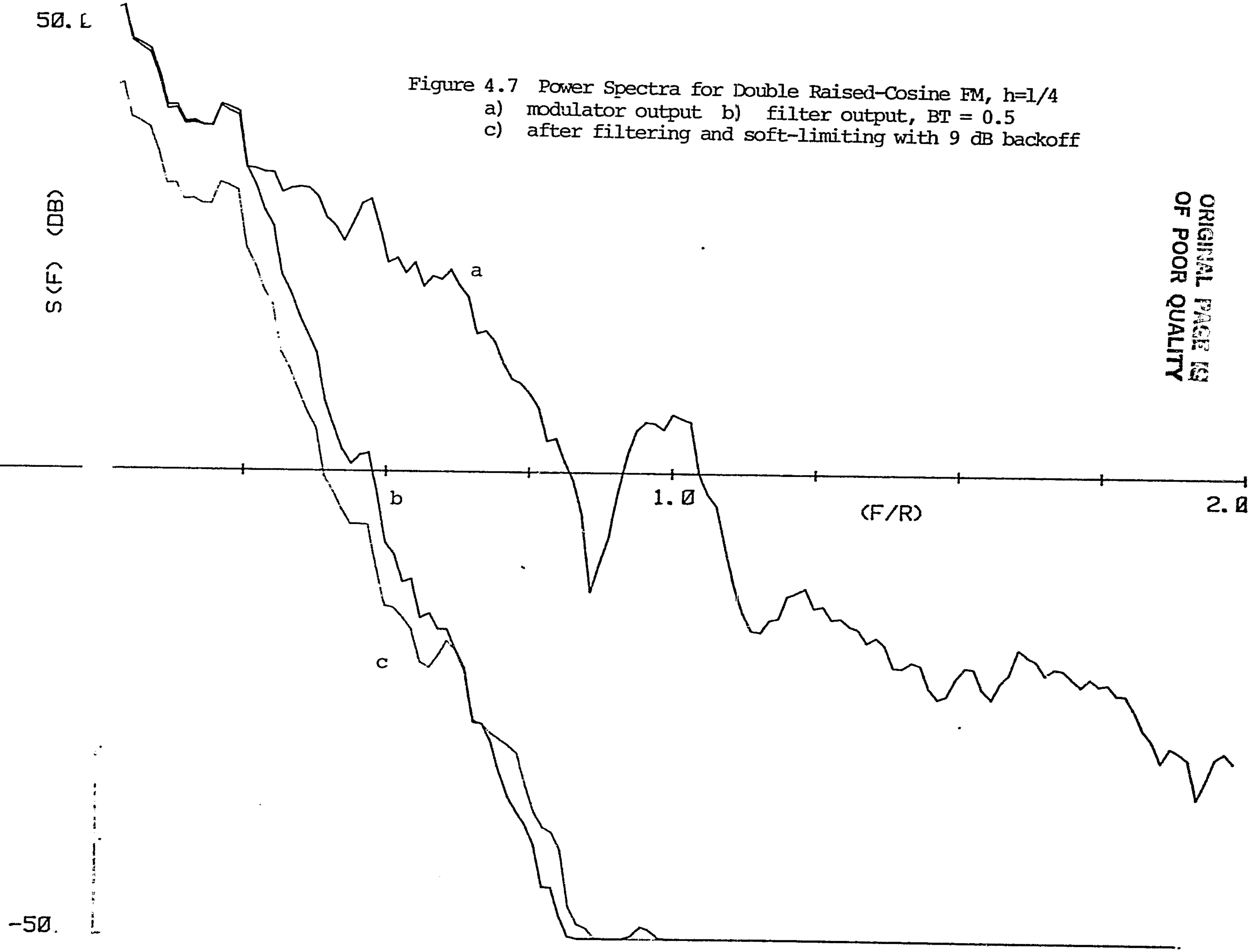


Figure 4.7 Power Spectra for Double Raised-Cosine FM, h=1/4
a) modulator output b) filter output, BT = 0.5
c) after filtering and soft-limiting with 9 dB backoff

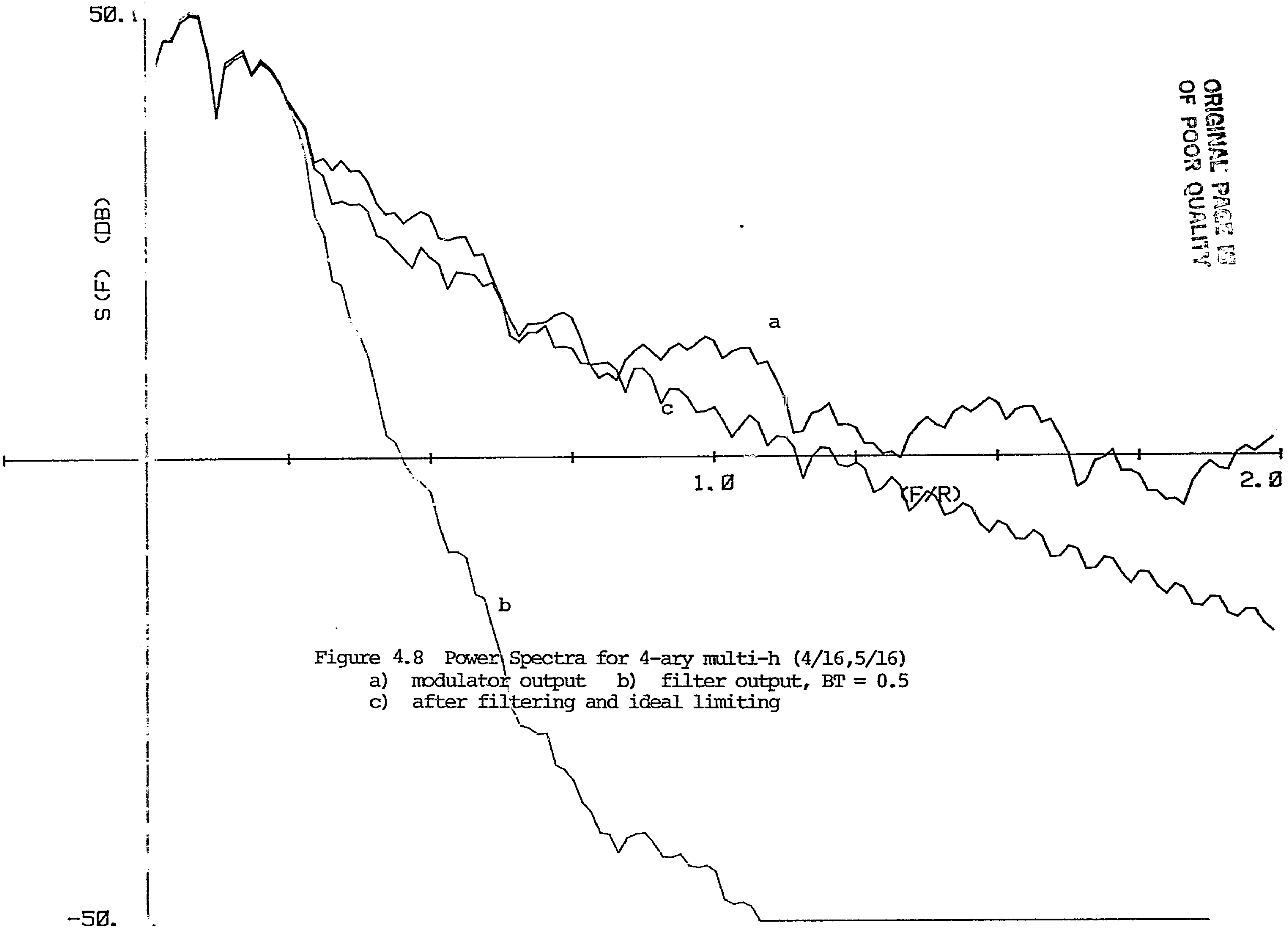


Figure 4.8 Power Spectra for 4-ary multi-h (4/16,5/16)
a) modulator output b) filter output, BT = 0.5
c) after filtering and ideal limiting

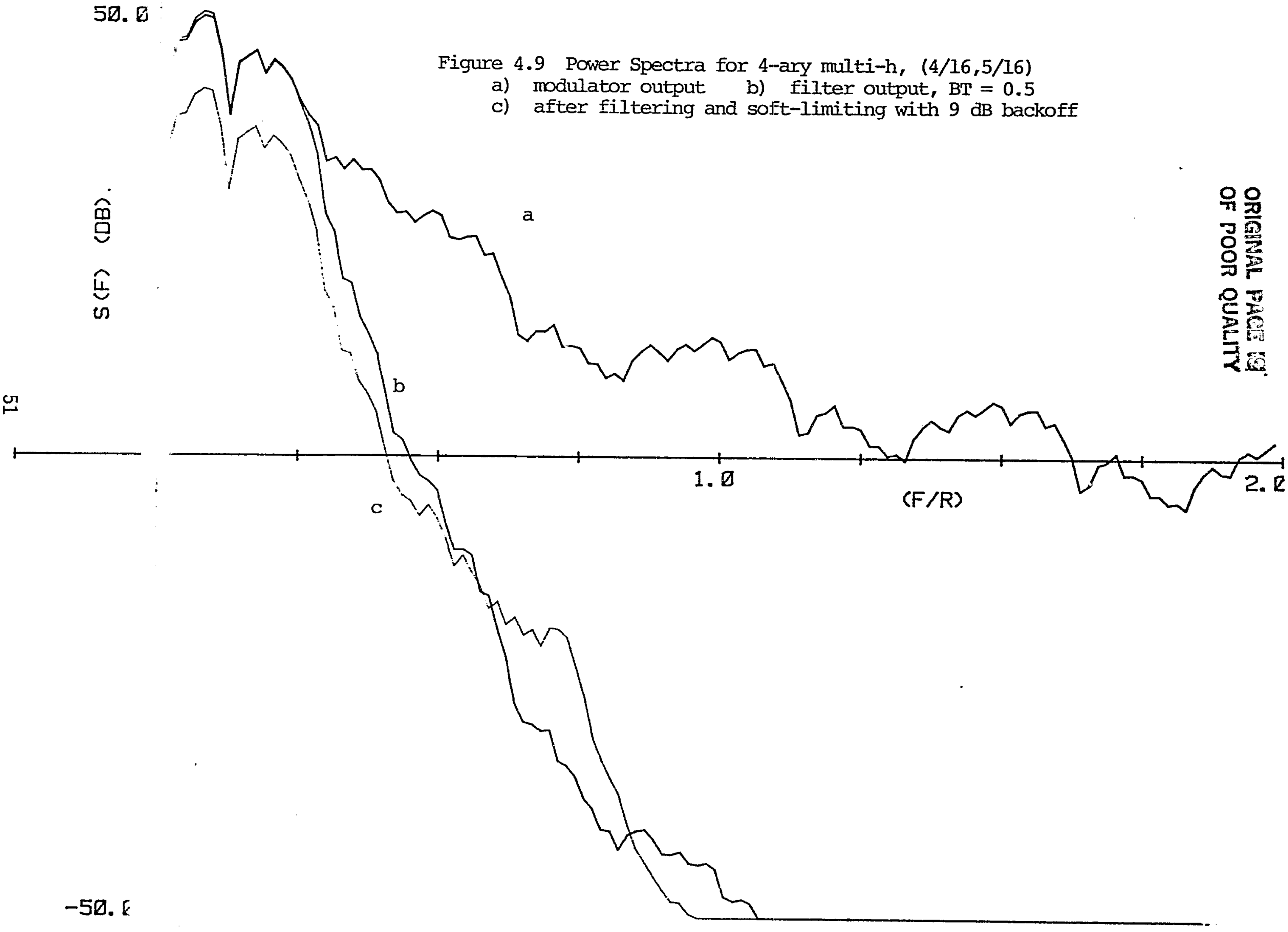


Figure 4.9 Power Spectra for 4-ary multi-h, (4/16,5/16)
a) modulator output b) filter output, BT = 0.5
c) after filtering and soft-limiting with 9 dB backoff

ORIGINAL PAGE IS
OF POOR QUALITY

characteristics similar to those of a typical satellite link. It is comprised of five basic sections; source (SORCE1, MARY), modulator (CPFSK, MULTIH, PARTLR), transmit filter (DIGBK), non-linear amplifier (TWT) and receive filter (DIGBK). Due to modularity, each section can be easily modified or totally skipped depending on the application. Furthermore, after each section the user can obtain complete documentation including envelope plot, and power spectrum phase trajectory, (PHASP, ENVELOP, FFTBK). When SIMUL1 is finished, 2048 samples of phase, represented in complex form are written to a file for use withe the receiver program. The over-all flow diagram is shown in Figure 4.10. A description of each section follows.

SORCE1 and MARY

Subroutines SORCE1 and MARY comprise the data source for the simulation. SORCE1 generates a length 256 psuedo-random sequence of 1's and -1's and stores them in an array for future reference. The sequence is formed using a 8-bit shift register with feedback. Subroutine MARY converts the binary sequence generated by SORCE1 into a M-ary sequence of length 256/K, where $K = \log_2 M$ and K is specified by the user. The M-ary symbol values are contained in the set $\{-(M-1),\ldots,-3,-1,1,3,\ldots(M-1)\}$ and the mapping is arbitrary.

CPFSK, MULTIH AND PARTLR

Subroutines CPFSK, MULTIH and PARTLR form the modulator section of the simulation. In each case 2048 samples of the phase (8 samples per bit), represented in complex form are returned in a 2048X2 array.

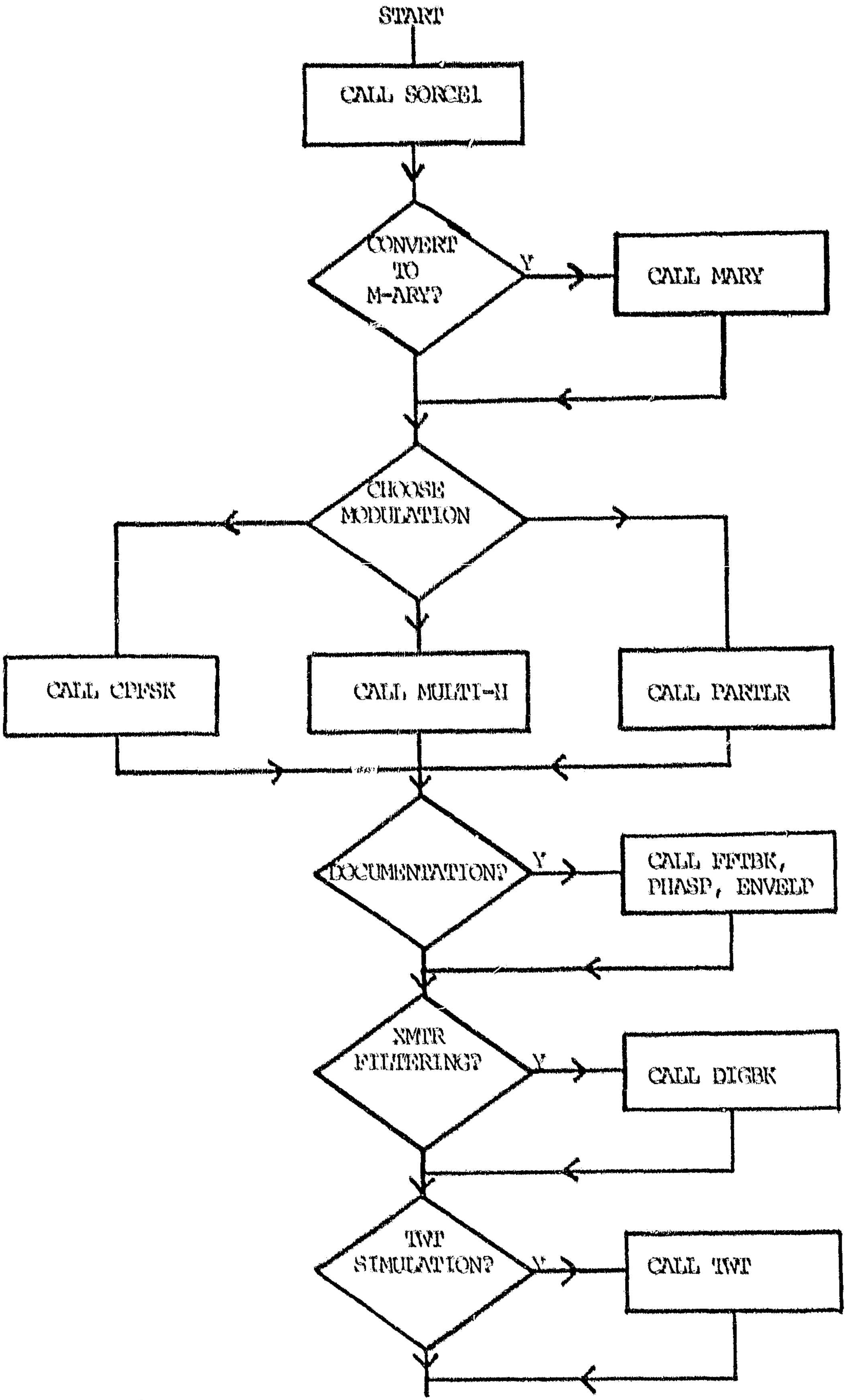


Figure 4.10 Software Flow Diagram for Simulation Package

CPFSK simulates a continuous-phase FSK modulator with deviation parameter h specified by the user. The psuedo-random M-ary sequence generated by the source, in conjunction with h, determines whether the phase should advance or retard and by how much. The phase varies linearly over 1 symbol time with accumulated phase $m = d_n h\Pi/M$. For the special case h = .5 the modulation becomes MSK.

MULTIH simulates a multi-h phase code modulator with $h_1, h_2 \ldots h_K$ specified by the user. MULTIH is similar to CPFSK except that h is now time varying.

PARLTR simulates a partial response FM code modulator with the response time K, the deviation parameter h and the pulse shape $g(\tau)$ specified by the user. The pulse shape $g(\tau)$ can be either rectangular or raised-cosine and lasts a total of K symbol durations. The composite phase at any one sample point is thus a function of the phase induced by the present symbol d_n and the phase induced by K-1 symbols of the past. For the special case K = 2 and g(τ` rectangular (binary source), the modulation is known as duobinary FM.

DIGBK

Subroutine DIGBK simulates a low pass, 4-pole Chebychev digital filter with bandwidth BW specified by the user. The filter coefficients are pre-calculated by a digital filter design program employing the bilinear transformation and then stored on disk for future use. Subroutine DIGFIL within DIGBK reads these coefficients off the disk and filters the time series data via a recursive difference equation technique. The time series data is run through the filter twice to establish steady state conditions.

TWT

Subroutine TWT simulates a non-linear amplifier with characteristics similar to those of a Travelling Wave Tube amplifier used in may satellite transponders. Output distortion is a combination of two effects, AM-AM conversion and AM-PM conversion.

With AM-AM distortion, the output envelope is a non-linear function of the input envelope. The non-linearity is either a hard limiter in which case the output envelope is set to unity regardless of the shape of the input envelope, or a soft limiter described by the equation $y = \mathrm{erf}(x/\sigma)$. In the latter case σ controls the degree of limiting; i.e. as σ increases the amplifier becomes more linear and as σ decreases the amplifier saturates and behaves more like a hard limiter. $\mathrm{Erf}(x/\sigma)$ is computed using a 5-term power series expansion with maximum error less than 7.5×10^{-8}.

In AM-PM conversion, fluctuations in the input envelope cause phase distortion in the output waveform. This distortion is modeled by a 3° per dB characteristic, i.e.

$$\theta = -(3.0/57.3)\ 20 \log(A)$$

where θ is the resulting change in output phase and A is the magnitude of the input waveform.

FFTBK, PHASP and ENVELP

Subroutines FFTBK, PHASP and ENVELOP comprise the documentation section of the simulation. FFTBK computes and plots an estimate of the power spectral density, S(f), while PHASP computes and plots a phase trajectory (modulo 2Π) and ENVELOP computes and plots the envelope or magnitude of the complex time series data.

Within subroutine FFTBK the time series data is first pre-multiplied by a Hanning window of the form

$$H(n) = .54 + .46 \cos \left(\frac{2\Pi n}{2048}\right)$$

to improve accuracy at low values of S(f) (-50dB down). The modified time series data is transformed into the frequency domain via a 2048 point complex FFT. The magnitude of the resulting, frequency samples are then averaged with 10 adjacent samples to provide a smoother estimation of the power spectral density.

REFERENCES

1. Morais, D. H., and Feher, K., "The Effects of Filtering and Limiting on the Performance of QPSK, Offset QPSK, and MSK Systems," IEEE Trans. on Communications, December 1980.

2. Cruz, J. R. and Simpson, R. S., "Cochannel and Intersymbol Interference in Quadrature-Carrier Modulation Systems," IEEE Trans. on Communications, March 1981.

3. Prabhu, V. K., "MSK and Offset QPSK Modulation with Bandlimiting Filters," IEEE Trans. on Aerospace and Electronic Systems, January 1981.

ORIGINAL PAGE IS
OF POOR QUALITY

APPENDIX A

```
      PROGRAM SIMUL1
C
C     THIS PROGRAM SIMULATES A DIGITAL COMM. SYSTEM
C     INCLUDING THE FOLLOWING COMPONENTS; SOURCE(BINARY OR M-ARY)
C     MODULATOR(CPFSK,MULTI-H,PARTLR), XMTR FILTER(4 POLE CHEBYCHEV
C     TWT NONLINEARITY(HARD OR SOFT LIMITER), RCVR FILTER
C
      REAL MODID(4),H(3)
      DIMENSION A(2048,2),IBIT(256),AR(2048),AI(2048)
      EQUIVALENCE (A(1,1),AR(1)),(A(1,2),AI(1))
      BYTE ANS,YEA,TMPFIL(15),CORFIL(15)
      BYTE TFNAME(15),RFNAME(15),FNAME(15),TWTID(4)
      DATA FS/8.0/,NDATA/256/,IEXP/11/,MRY/1/
      DATA FNAME /'N','O','N','E',11*' '/
      DATA TFNAME /'N','O','N','E',11*' '/
      DATA RFNAME /'N','O','N','E',11*' '/
      DATA TWTID /'N','O','N','E'/
      DATA YEA/'Y'/,N/0/,H/3*0.0/,NT /0/
      DATA TMPFIL /'T','M','P','F','I','L',9*' '/
      DATA CORFIL /'C','O','R','F','I','L',9*' '/
      DATA MODID /4*'    '/
      CALL ASSIGN (50,TMPFIL,14,'NEW')
      ND=IFIX(FS)*NDATA
C
C     GENERATE 255 BIT PN BINARY OR M-ARY SOURCE
C
20    CALL SORCE1(NDATA,IBIT)
      WRITE(7,212)
212   FORMAT('OBINARY OR M-ARY? (0 OR 1)')
      READ(5,214) IANS
214   FORMAT(I5)
      IF(IANS.EQ.1) CALL MARY(NDATA,IBIT,MRY)
C
C     CHOOSE MODULATION TYPE
C
25    WRITE (7,198)
198   FORMAT('OCHOOSE MODULATION'/
     &' TYPE'/
     &' 1. CP-FSK'/
     &' 2. MULTI-H'/
     &' 3. PARTIAL RESPONSE'/
     &'$WHICH?')
```

```
            READ (5,200) ISW
      200   FORMAT (I6)
            IF(ISW.EQ.1) CALL CPFSK(A,ND,NDATA,FS,IBIT,H(1),N
            &,MODID,MRY)
            IF(ISW .EQ. 2) CALL MULTIH(A,ND,NDATA,FS,IBIT,H,N,MODID
            &,K,MRY)
            IF(ISW.EQ.3) CALL PARTLR(A,ND,NDATA,FS,IBIT,H(1),N
            &,MODID,NT,MRY)
      C
      C     OTHER BRANCHES MADE HERE AS THEY BECOME AVAILABLE
      C
            IF(N .NE. 0) GO TO 27
            CALL PRINT('NOT IMPLEMENTED AT THIS TIME')
            GO TO 25
      C
      C     DOCUMENTATION (PHASE TRAJECTOR,FFT,ENVELOPE PLOT)
      C
      27    WRITE(7,220)
      220   FORMAT('OPHASE TRAJECTOR PLOT?')
            READ(5,204) ANS
            IF(ANS.EQ.YEA) CALL PHASP(A,AR,AI,ND)
            CALL FFTBK(A,AR,AI,ND,IEXP)
      C
      C     XMTR FILTER AND DOCUMENTATION
      C
            CALL PRINT('DO YOU WANT CHANNEL FILTERING AT THE XMTR? (Y/N)')
            READ (5,204) ANS
      204   FORMAT (A1)
            IF(ANS .NE. YEA) GO TO 30
            CALL DIGBK(A,AR,AI,ND,TFNAME)
            WRITE(7,206)
      206   FORMAT ('ODO YOU WANT AN ENVELOPE PLOT (Y/N)')
            READ (5,204) ANS
            IF (ANS .EQ. YEA) CALL ENVELP(A,AR,AI,ND,FS)
            WRITE(7,220)
            READ(5,204) ANS
            IF(ANS.EQ.YEA) CALL PHASP(A,AR,AI,ND)
            CALL FFTBK(A,AR,AI,ND,IEXP)
      C
      C     TWT NOLIN AND DOCUMENTATION
      C
      30    CALL PRINT('DO YOU WANT A TWT SIMULATION? (Y/N)')
            READ (5,204) ANS
            IF(ANS .NE. YEA) GO TO 50
            CALL TWT(A,ND,TWTID)
            WRITE(7,206)
            READ(5,204) ANS
            IF(ANS .EQ. YEA) CALL ENVELP(A,AR,AI,ND)
            CALL FFTBK(A,AR,AI,ND,IEXP)
      C
      C     RCVR FILTER AND DOCUMENTAION
```

```
C
50      CALL PRINT('WOULD YOU LIKE ADDITIONAL CHANNEL FILTERING (Y/N)')
        READ(5,204) ANS
        IF (ANS .NE. YEA) GO TO 60
        CALL DIGBK(A,AR,AI,ND,RFNAME)
        WRITE(7,206)
        READ(5,204) ANS
        IF (ANS .EQ. YEA) CALL ENVELP(A,AR,AI,ND)
        WRITE(7,220)
        READ(5,204) ANS
        IF(ANS.EQ.YEA) CALL PHASP(A,AR,AI,ND)
        CALL FFTBK(A,AR,AI,ND,IEXP)
C
C       WRITE TIME SERIES DATA TO A FILE
C
        CALL CLOSE(50)
60      CALL PRINT('DO YOU WANT TO WRITE SAMPLES TO A FILE? (Y/N)')
        READ(5,204) ANS
        IF(ANS .NE. YEA) GO TO 70
        CALL PRINT('INPUT FILENAME')
        IERR=0
        CALL GETSTR(5,FNAME,14,IERR)
        CALL ASSIGN(10,FNAME,14,'NEW')
        WRITE(10) FNAME,MODID,NT,H,FS,TFNAME,TWTID,RFNAME,K,IBIT,A
        CALL CLOSE(10)
70      WRITE(7,208) FNAME,(MODID(J),J=1,2),NT,(MODID(J),J=3,4)
        WRITE(7,209) H(1),H(2),H(3),FS,TFNAME,TWTID,RFNAME
208     FORMAT('OSUMMARY OF SIMULATION'/
        &'OFILENAME: ',15A1/
        &' MODULATION TYPE: ',2A4,3X,I2,2A4)
209     FORMAT(' H(1)=',F8.3,3X,'H(2)=',F8.3,3X,'H(3)=',F8.3/
        &' FS=',F8.3/
        &' XMTR FLTR: ',15A1/
        &' TWT       : ',4A1/
        &' RCVR FLTR: ',15A1)
        WRITE(7,210)
210     FORMAT ('OSIMULATION IS COMPLETE')
        STOP
        END
*
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
         SUBROUTINE SORCE1(NDATA,IBIT)
C
C        THIS PROGRAM SIMULATES A 255 BIT PN GENERATOR
C        NDATA=# OF SOURCE BITS, BIT=DATA, (1=1,0= -1)
C
         DIMENSION IBIT(NDATA)
         LOGICAL D(8),F(3)
         DATA IU/32757/,IX/7057/
         DATA D/7*.FALSE.,.TRUE./
         DO 50 I=1,NDATA
         F(1)=(.NOT.D(8).AND.D(7)) .OR. (D(8).AND.(.NOT.D(7)))
         F(2)=(.NOT.F(1).AND.D(3)) .OR. (F(1).AND.(.NOT.D(3)))
         F(3)=(.NOT.F(2).AND.D(2)) .OR. (F(2).AND.(.NOT.D(2)))
C
C        PERFORM THE RIGHT SHIFT
         DO 30 J=1,7
         D(9-J)=D(8-J)
30       CONTINUE
         D(1)=F(3)
         IF (D(8).EQ..FALSE.) IBIT(I) = -1
         IF (D(8).EQ..TRUE.) IBIT(I) = 1
50       CONTINUE .
         RETURN
         END
*
```

```
      SUBROUTINE MARY(NDATA,IBIT,M)
C
C     THIS SUBROUTINE MAPS BINARY DATA INTO M-ARY DATA
C
      DIMENSION IBIT(NDATA)
20    WRITE(7,200)
200   FORMAT('OINPUT K (M=2**K ,M AS IN M-ARY)'/
     &' 1< K >=6')
      READ (5,210) M
210   FORMAT(I5)
      IF(M.LE.1) GO TO 20
      IF(M.GE.6) GO TO 20
C
C     PERFORM MAPPING
C
      L=0
      K=0
      MINBIT=-(2**M)+1
30    DO 80 K=1,NDATA,M
      L=L+1
      J=MINBIT
      DO 60 I=1,M
      IF(IBIT(K+I-1).EQ.-1) IBIT(K+I-1)=0
      J=J+(2**(M-I+1))*IBIT(K+I-1)
60    CONTINUE
      IBIT(L)=J
80    CONTINUE
      RETURN
      END
*
```

```
      SUBROUTINE CPFSK(A,ND,NDATA,FS,IBIT,H,N,MODID,M)
C
C     THIS PROGRAM SIMULATES CPFSK AND RETURNS SAMPLED
C     VALUES IN AN ARRAY(A).  A=OUTPUT ARRAY,NDATA=# OF
C     SOURCE BITS, FS=SAMPLING FREQUENCY RELATIVE TO DATA RATE
C     IBIT = DATA,H=FREQUENCY DEVIATION
C
      DIMENSION A(ND,2),IBIT(NDATA),TODID(2)
      REAL PI,MODID(2)
      DATA TODID/'CP-F','SK  '/
      DATA PI/3.141592/,ANG/0./,K/0/
      DATA OFST/0.0/
      MODID(1)=TODID(1)
      MODID(2)=TODID(2)
C
C     DETERMINE MODULATOR PARAMETERS
C
      WRITE (7,100)
100   FORMAT ('OTHIS SIMULATES CP-FSK.'/
     $' INPUT H')
      READ (5,102) H
102   FORMAT (F6.3)
      IFS=IFIX(FS)*M
      NSYMBL=NDATA/M
C
C     PERFORM SIMULATION
C
      I=0
10    I=I+1
      J=0
20    J=J+1
      ANG=OFST+(IBIT(I)*H*PI*J)/(FS*M)
        K=K+1
      A(K,1)=COS(ANG)
      A(K,2)=SIN(ANG)
      IF(J.LT.IFS) GO TO 20
      OFST=OFST+IBIT(I)*H*PI
      IF(I.LT.NSYMBL) GO TO 10
      IF(K.EQ.ND) GO TO 50
      IFS=ND-IFS*NSYMBL
      GO TO 10
50    CONTINUE
      N=1
      RETURN
      SUBROUTINE MULTIH(A,ND,NDATA,FS,IBIT,H,N,MODID,K,M)
C
C     THIS SUBROUTINE SIMULATES A MULTI-H PHASE CODE
C     MODULATOR.  A=TIME SERIES DATA, NDATA=# OF DATA BITS,
C     FS= SAMPLING RATE, IBIT=DATA BIT ARRAY, H=DEVIATION PARAMETER,
      N SET TO 1, MODID= IDENTIFICATION, K=# OF H PARAMETERS,
      M AS IN M-ARY
C
      DIMENSION A(ND,2),H(3),IBIT(NDATA),TODID(2)
      REAL MODID(2)
      DATA TODID /'MULT','I-H '/,ANG /0.0/, PI /3.14159/
      DATA KMAX /3/, L /0/
      MODID(1)=TODID(1)
      MODID(2)=TODID(2)
      N=1
      NSYMBL=NDATA/M
      IFS=IFIX(FS)*M
C
C     INPUT PARAMETERS (K,H(K))
C
25    WRITE(7,200)
200   FORMAT('OTHIS SIMULATES A MULTI-H PHASE CODE MODULATOR'/
      &' INPUT K (# OF H PARAMETERS'/
      &' (NOTE, KMAX = 3)')
27    READ(5,202) K
202   FORMAT(I3)
      IF(K.LT.4) GO TO 30
      WRITE(7,204)
204   FORMAT('OKMAX = 3, INPUT K')
      GO TO 27
30    WRITE(7,206)
206   FORMAT('OINPUT H(1),...H(K)')
      READ(5,208) (H(I),I=1,K)
208   FORMAT(3F6.3)
C
C     PERFORM SIMULATION
C
      OFST=0.0
      I=0
50    I=I+1
      MH=MOD(I+(K-1),K)+1
      J=0
75    J=J+1
      ANG=OFST+(IBIT(I)*H(MH)*PI*J)/(FS*M)
      L=L+1
      A(L,1)=COS(ANG)
      A(L,2)=SIN(ANG)
      IF(J.LT.IFS) GO TO 75
      OFST=OFST+IBIT(I)*H(MH)*PI
      IF(I.LT.NSYMBL) GO TO 50
      IF(L.EQ.ND) GO TO 90
      IFS=ND-IFS*NSYMBL
      GO TO 50
90    RETURN
      END
*
```

```
      SUBROUTINE PARTLR(A,ND,NDATA,FS,IBIT,H,N,MODID,NTT,MRY)
C
C     THIS PROGRAM SIMULATES A PARTIAL RESPONSE FM CODE
C     MODULATOR FOR TWO TYPES OF PULSES, DUOBINARY FM
C     (N-RECTANGULAR) AND DOUBLE RAISED COSINE (NRC)
C     A=TIME SERIES DATA, ND= # OF DATA POINTS, NDATA=# OF
C     DATA BITS, FS=SAMPLING RATE, IBIT=DATA BIT ARRAY, H=
C     DEVIATION PARAMETER, N IS SET TO 1, MODID= IDEN. NTT=RES-
C     PONSE TIME, MRY AS IN M-ARY
C
      DIMENSION A(ND,2),IBIT(NDATA),TODID(6),IPREV(7)
      REAL HODID(4)
      COMMON PFS,PI,PH,NT,MM
      DATA TODID /'PART','RESP','-REC','T   ',' -RC','    '/
      DATA PI /3.14159265359/,K/0/
      DATA NTMAX /7/
      PFS=FS
      N=1
      MODID(1)=TODID(1)
      MODID(2)=TODID(2)
      MM=MRY
C
C     DETERMINE MODULATOR PARAMETERS (PULSE SHAPE,K)
C
      WRITE (7,200)
200   FORMAT('0THIS SIMULATES A PARTIAL RESPONSE FM MODULATOR'/
     &' FOR THE FOLLOWING CASES'/
     &' 1. DUOBINARY FM (N-RECTANGULAR)'/
     &' 2. DOUBLE RAISED COSINE (NRC)'/
     &' WHICH?')
30    READ(5,202) ISW
202   FORMAT (I6)
      IF(ISW.GT.2) GO TO 40
      IF(ISW.LT.1) GO TO 40
      GO TO 50
40    WRITE(7,204)
204   FORMAT('0NO SUCH CHOICE, CHOOSE AGAIN')
      GO TO 30
50    WRITE(7,208)
208   FORMAT('0INPUT H,NT'/
     &' (NT=DURATION OF FREQ. PULSE, NTMAX=7)')
      READ(5,210) H,NT
210   FORMAT(F6.3,I5)
      NTT=NT
```

```
        IF(NT.LT.NTMAX) GO TO 80
        WRITE(7,198) NTMAX
198     FORMAT(' NT MUST BE <',I5)
        GO TO 50
80      CONTINUE
        IF(ISW.EQ.2) GO TO 85
        MODID(3)=TODID(3)
        MODID(4)=TODID(4)
        GO TO 87
85      MODID(3)=MODID(5)
        MODID(4)=TODID(6)
87      CONTINUE
        PH=H
        IFS=IFIX(FS)*MM
        NSYMBL=NDATA/MM
C
C       PERFORM SIMULATION
C
        DO 90 IN=1,NT-1
        IPREV(NT+1-IN)=1
90      CONTINUE
        IPREV(1)=IBIT(1)
        D=0.0
        M=0
100     M=M+1
        J=0
110     J=J+1
        K=K+1
        L=J
        P=0.0
        DO 120 IN=1,NT
        P=P+PHASE(IPREV(IN),L,ISW)
        L=L+(8*MM)
120     CONTINUE
        P=D+P
        A(K,1)=COS(P)
        A(K,2)=SIN(P)
        IF(J.LT.IFS) GO TO 110
        D=D+PH*PI*IPREV(NT)
        DO 135 IN=1,NT-1
        IPREV(NT+1-IN)=IPREV(NT-IN)
135     CONTINUE
        IF(M.GE.NSYMBL) GO TO 140
        IPREV(1)=IBIT(M+1)
        GO TO 100
140     IF(K.EQ.ND) GO TO 150
        IFS=ND-IFS*NSYMBL
        IPREV(1)=IBIT(M+1)
        GO TO 100
150     RETURN
        RETURN
```

```
      END
      REAL FUNCTION PHASE(L,M,N)
      COMMON PFS,PI,PH,NT,MM
      IF(N.EQ.1) GO TO 50
      PHA=(PH*PI*L)/(NT*MM)
      PHAS=SIN((2.0*PI*M)/(PFS*NT*MM))
      PHASE=PHA*((M/PFS)-(((NT*MM)/(2.0*PI))*PHAS))
      RETURN
50    PHASE=(L*PH*PI*M)/(NT*PFS*MM)
      RETURN
      END
*
```

```
        SUBROUTINE ENVELP(A,AR,AI,ND,FS)
C
C       THIS PROGRAM PLOTS A GRAPH OF THE SIGNAL
C       ENVELOPE VS. TIME.  A=TIME SERIES DATA, ND=
C       # OF DATA POINTS, FS=SAMPLING RATE,  AR(K)=A(K,1)
C       AI(K)=A(K,2)
C
        DIMENSION A(ND,2),AR(ND),AI(ND)
        REAL*8 XNAME(2),YNAME(2)
        DATA XNAME /'  TIME   ',' SECONDS'/
        DATA YNAME/' SIGNAL ','ENVELOPE'/
        NS=100
        NSMAX=150
        WRITE (50)A
C
C       DETERMINE PLOT PARAMETERS
C
        WRITE (7,200)
200     FORMAT('0ACCEPT (<CR>) OR CHANGE (/<NEW VALUE><CR> THE'/
        &' NUMBER OF SAMPLES TO BE PLOTTED(NS)'/
        &' (REMEMBER, # OF PAGES=NS/51)')
        CALL INTIO('     NS',NS,NSMAX,0)
C
C       CALCULATE MAGNITUDE
C
        DO 10 J=1,NS
        AI(J)=SQRT(A(J,1)**2+A(J,2)**2)
        AR(J)=J/FS
10      CONTINUE
C
C       PLOT ENVELOPE ON CRT OR DECWRITER AND SAVE DATA
C       ON FILE FOR HP-PLOTTER
C
        PAUSE 'ROLL PAPER TO NEXT TO BOTTOM LINE THEN <CR>'
        DUMB=0.0
        CALL PLOTR(AR,AI,NS,XNAME,YNAME,0.0,1.25,1)
        CALL HPPLT(A,ND,NS)
        REWIND 50
        READ (50) A
        REWIND 50
        RETURN
        END
*
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      SUBROUTINE PHASP(A,AR,AI,ND)
C
C     THIS SUBROUTINE CALCULATES AND PLOTS PHASE TRAJECTORIES.
C     PHASE = ARCTAN(AI/AR) MODULO(2PI)
C     A=TIME SERIES DATA, ND=# OF DATA POINTS, AR(K)=A(K,1)
C     AI(K)=A(K,2).
C
      DIMENSION A(ND,2),AR(ND),AI(ND)
      REAL*8 XNAME(2),YNAME(2)
      DATA PI/3.141592654/
      DATA XNAME/'PHASE   ','(RAD)   '/
      DATA YNAME/'SAMPLE  ','INDEX   '/
      WRITE(50) A
C
C     DETERMINE PLOTTING PARAMETERS
C
      TP=2*PI
      IFP=1
      NP=150
      WRITE(7,200)
200   FORMAT('0PHASE TRAJECTORY PLOTTER'/
     &' ACCEPT(<CR>) OR CHANGE(</NEW VALUE><CR>) '/
     &' THE # OF POINTS TO BE PLOTTED(NP')
      CALL INTIO('     NP',NP,ND,0)
      WRITE(7,202)
202   FORMAT('0FIRST POINT TO BE PLOTTED(IFP)')
      CALL INTIO('    IFP',IFP,ND-NP,0)
C
C     CAALCULATE PHASE MOD(2 PI)
C
      DO 50 J=IFP,IFP+NP
      AI(J)=ATAN2(AI(J),AR(J))
      AR(J)=J-1
50    CONTINUE
C
C     PLOT PHASE TRAJECTORY ON CRT OR DECWRITER AND SAVE DATA
C     ON FILE FOR USE WITH HP-PLOTTER(SUBROUTIN HPPLT)
C
      CALL HPPLT(A,ND,NP)
      PAUSE 'ROLL PAPER'
      CALL PLOTR(AR,AI,NP,XNAME,YNAME,-4*PI,4*PI,0)
      REWIND 50
      READ(50) A
      REWIND 50
      RETURN
      END
*
```

```
      SUBROUTINE FFTBK(A,AR,AI,ND,IEXP)
C
C     THIS SUBROUTINE COMPUTES AND PLOTS A POWER SPECTRAL
C     DENSITY VIA A HANNING WINDOW FOLLOWED BY A FFT AND A
C     N-POINT SMOOTHER.  A=TIME SERIES DATA, ND=# OF DATA POINTS
C     IEXP=LOG2(ND), AR(K)=A(K,1), AI(K)=A(K,2)
C
      REAL*8 XNAME(2),YNAME(2)
      DATA XNAME/' (F/R)  ','        '/
      DATA YNAME/'  GAIN  ','  (DB)  '/
      DATA PI/3.14159265358/
      DIMENSION X(129)
      DIMENSION A(ND,2),AR(ND),AI(ND),B(2),C(2)
      BYTE ANS,YEA
      DATA YEA/'Y'/
C
      WRITE(7,202)
202   FORMAT ('0DO YOU WANT A FFT?  (Y/N)')
      READ (5,204) ANS
204   FORMAT (A1)
      IF(ANS .NE. YEA)  RETURN
      WRITE (50) A
C
C     MULTIPLY TIME SERIES DATA BY HANNING WINDOW
C
      DO 20 K=1,ND
      Y=FLOAT(K-ND/2)/FLOAT(ND)
      W=0.5+0.5*COS(2*PI*Y)
      A(K,1)=A(K,1)*W
      A(K,2)=A(K,2)*W
20    CONTINUE
      CALL FFTR(A,ND,IEXP,-1.)
C
C     DETERMINE SMOOTHING AND PLOTTING PARAMETERS
C
      KS=11
      NLP=512
      NND=ND/2
      NBP=4
      WRITE(7,206)
206   FORMAT ('0ACCEPT (<CR>) OR CHANGE (/<NEW VALUE><CR> THE'/
     &' FOLLOWING PARAMETER VALUES ASSOCIATED WITH THE PLOT'/
     &' # OF SAMPLES BETWEEN PLOT POINTS (NBP):')
```

```
        CALL INTIO('    NBP',NBP,40,0)
        CALL PUTSTR(7,' LAST SAMPLE POINT TO BE PLOTTED(NLP):',0)
        CALL INTIO('    NLP',NLP,NND,0)
        WRITE(7,208)
208     FORMAT('0SMOOTHING CONSTANT(KS)'/
        &' KS MUST BE ODD,KS=1 FOR NO SMOOTHING')
        CALL INTIO('     KS',KS,15,0)
        KS1=(KS-1)/2
        KS2=KS1+1
        NPTS=NLP/NBP
C
C       CALCULATE MAGNITUDE AND PERFORM KS-POINT SMOOTHING OPERATION
C
        DO 25 J=1,NND
25      AR(J)=A(J,1)**2+A(J,2)**2
        K=0
        DO 27 J=1,NLP,NBP
        K=K+1
        X(K)=0.
        IM=MAX0(J-KS1,1)
        DO 26 KK=IM,J+KS1
26      X(K)=X(K)+AR(KK)
        IM2=MIN0(5,J+KS2-IM)
        X(K)=X(K)/FLOAT(IM2)
        X(K)=10.*ALOG10(X(K))
        IF(X(K).LT.-50.) X(K)=-50.0
27      AR(K)=FLOAT(J-1)/256.
        DO 28 KK=1,NPTS
28      AI(KK)=X(KK)
C
C       PLOT POWER SPECTRUM ON CRT OR DECWRITER AND SAVE DATA ON
C       FILE FOR HP-PLOTTER.
C
        CALL HPPLT(A,ND,NPTS)
        PAUSE 'ROLL PAPER TO NEXT TO BOTTOM LINE THEN <CR>'
        DUMB=0.0
        CALL PLOTR(AR,AI,NPTS,XNAME,YNAME,-50.0,50.0,1)
        REWIND 50
        READ(50)A
        REWIND 50
        RETURN
        END
*
```

```
      SUBROUTINE TWT(X,ND,TWTID)
C
C     THIS SUBROUTINE SIMULATES A TRAVELLING WAVE TUBE
C     TYPE AMPLIFIER.  X= TIME SERIES DATA, ND=#OF DATA POINTS
C     TWTID=IDENTIFICATION
C     AM-AM DETERMINED BY FUNCTION F1
C     AM-PM DETERMINED BY FUNCTION F2
C
      DIMENSION X(ND,2)
      BYTE TWTID(4),DWTID(4)
      DATA DWTID /'Y','E','S',' '/
      DATA S/1.0/
      DATA PI/3.14159265359/
      COMMON S,SP
      SP=1.0/SQRT(2*PI)
      DO 5 J=1,4
      TWTID(J)=DWTID(J)
5     CONTINUE
C
C     DETERMINE LIMITER TYPE (AND) SIGMA
C
50    WRITE(7,200)
200   FORMAT('OHARD LIMITER OR SOFT LIMITER? (0 OR 1)')
      READ(5,210) ISW
210   FORMAT(4I)
      IF(ISW.EQ.1) GO TO 60
      IF(ISW.NE.0) GO TO 50
      GO TO 70
60    WRITE(7,220)
220   FORMAT('OINPUT SIGMA')
      READ(5,230) S
230   FORMAT(F6.3)
C
C     PERFORM SIMULATION
C     ISW=1 IMPLIES SOFT LIMITER
C
70    DO 100 I=1,ND
      A=SQRT(X(I,1)**2+X(I,2)**2)
      PHI=ATAN2(X(I,2),X(I,1))
90    AMP=A/S
      AO=F1(AMP,ISW)
      DPHI=F2(A,ISW)
      X(I,1)=AO*COS(PHI+DPHI)
      X(I,2)=A0*SIN(PHI+DPHI)
100   CONTINUE
150   RETURN
      END
      REAL FUNCTION F1(A,ISW)
C     IDEAL LIMITER
      DATA B1 /.319381530/, B2/-.356563782/
      DATA B3/1.781477937/,B4/-1.821255978/,B5/1.330274429/
      DATA P/.2316419/,PI/3.14159265359/
      COMMON S,SP
      IF(ISW.EQ.0) GO TO 50
      Z=SP*EXP(-(A**2)/2.0)
      T1=1.0/(1.0+P*A)
      T2=T1**2
      T3=T2*T1
      T4=T3*T1
      T5=T4*T1
      F=Z*(B1*T1+T2*B2+B3*T3+B4*T4+B5*T5)
      F1=2*(.5-F)
      RETURN
50    F1=1.0
      RETURN
      END
      FUNCTION F2(A,ISW)
      IF(ISW.EQ.1) GO TO 50
      F2=-(3.0/57.3)*(20*ALOG10(A))
      RETURN
50    F2=0.0
      RETURN
      END
*
```

```
      SUBROUTINE DIGBK(A,AR,AI,ND,FNAME)
C
C     THIS SUBROUTINE SIMULATES A VARIABLE BANDWIDTH, 4-POLE
C     CHEBYCHEV LOW PASS FILTER. (SUBROUTINE DIGFIL DOES THE
C     ACTUAL FILTERING).  A=TIME SERIES DATA, ND=# OF DATA POINTS,
C     FNAME=FILTER COEFFICIENT FILENAME
C
      BYTE ANS,YEA,FNAME(15)
      LOGICAL FIRST
      DIMENSION A(ND,2),CR(8,5),CI(8,5),AR(ND),AI(ND)
      DATA YEA /'Y'/
      WRITE (50) A
C
C     SELECT FILTER BANDWIDTH
C
      WRITE (7,200)
200   FORMAT ('OFILE      DESCRIPTION'/
     &'OTEST1    CHEBYCHEV, 4-POLE, 0.5 DB RIPPLE, FCO=1'/
     &' TEST2    CHEBYCHEV, 4-POLE, 0.5 DB RIPPLE, FCO=.5'/
     &'OINPUT FILTER COEFFICIENT FILENAME')
C
C     READ FILTER COEFFICIENTS FROM FILE
C
      CALL GETSTR(5,FNAME,14,IERR)
      CALL ASSIGN (10,FNAME,14,'RDO')
      READ(10,208) FCO,FS,RIP,NP,FSTP,HSTP,N,((CR(J,I),J=1,8),I=1,N)
208   FORMAT (/3(28X,G13.6/),28X,I3/,2(28X,G13.6/),33X,I2,//,
     &(2X,8(2X,G13.6)))
      CALL CLOSE(10)
      DO 20 J=1,8
      DO 10 I=1,N
      CI(J,I)=CR(J,I)
10    CONTINUE
20    CONTINUE
C
C     PERFORM FILTERING VIA SUBROUTINE DIGFIL.  DATA IS PASSED
C     THROUGH FILTER TWICE TO OBTAIN INITIAL CONDITIONS.
C
      FIRST=.TRUE.
30    CALL DIGFIL(2,CR,ND,AR,AR)
      CALL DIGFIL(2,CI,ND,AI,AI)
      IF(FIRST .EQ..FALSE.) GO TO 40
      FIRST=.FALSE.
      REWIND 50
      READ (50) A
      GO TO 30
40    REWIND 50
      RETURN
      END
*
```