OPTIMAL BAYESIAN ESTIMATION OF THE STATE OF A PROBABILISTICALLY MAPPED MEMORY-CONDITIONAL MARKOV PROCESS WITH APPLICATION TO MANUAL MORSE DECODING

Edison Lee Bell

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

OPTIMAL BAYESIAN ESTIMATION OF THE STATE OF A PROBABILISTICALLY MAPPED MEMORY-CONDITIONAL MARKOV PROCESS WITH APPLICATION TO MANUAL MORSE DECODING

by

Edison Lee Bell

September 1977

Thesis Advisor: S. Jauregui

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) Optimal Bayesian Estimation of the State of a Probabilistically Mapped Memory-Conditional Markov Process with Application to Manual Morse Decoding	5. TYPE OF REPORT & PERIOD COVERED Doctor of Engineering Thesis; September 1977
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Edison Lee Bell	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	12. REPORT DATE September 1977
	13. NUMBER OF PAGES 198
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Manual Morse Decoding

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This dissertation investigates the problem of automatic transcription of the hand-keyed Morse signal. A unified model for this signal process transmitted over a noisy channel is shown to be a system in which the state of the Morse process evolves as a memory-conditioned probabilistic mapping of a conditional Markov process, with the state of this process playing the role of a parameter vector of the channel model.

(20. ABSTRACT Continued)

The decoding problem is then posed as finding an optimal estimate of the state of the Morse process, given a sequence of measurements of the detected signal. The Bayesian solution to this nonlinear estimation problem is obtained explicitly for the parameter-conditional linear-gaussian channel, and the resulting optimal decoder is shown to consist of a denumerable but exponentially expanding set of linear Kalman filters operating on a dynamically evolving trellis. Decoder performance is obtained by computer simulation, for the case of random letter message texts. For nonrandom texts, further research is indicated to specify linguistic and format-dependent models consistent with the model structure developed herein.

Optimal Bayesian Estimation of the State of a Probabilistically Mapped Memory-Conditional Markov Process with Application to Manual Morse Decoding

by

Edison Lee Bell
Lieutenant, United States Navy
B.E.E., Georgia Institute of Technology, 1969
M.S.E.E., Naval Postgraduate School, 1974
Elec. Engr., Naval Postgraduate School, 1975

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF ENGINEERING

from the
NAVAL POSTGRADUATE SCHOOL
September 1977

ABSTRACT

This dissertation investigates the problem of automatic transcription of the hand-keyed Morse signal. A unified model for this signal process transmitted over a noisy channel is shown to be a system in which the state of the Morse process evolves as a memory-conditioned probabilistic mapping of a conditional Markov process, with the state of this process playing the role of a parameter vector of the channel model. The decoding problem is then posed as finding an optimal estimate of the state of the Morse process, given a sequence of measurements of the detected signal. The Bayesian solution to this nonlinear estimation problem is obtained explicitly for the parameter-conditional linear-gaussian channel, and the resulting optimal decoder is shown to consist of a denumerable but exponentially expanding set of linear Kalman filters operating on a dynamically evolving trellis. Decoder performance is obtained by computer simulation, for the case of random letter message texts. For nonrandom texts, further research is indicated to specify linguistic and format-dependent models consistent with the model structure developed herein.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

I wish to express my deep appreciation to Dr. Stephen Jauregui for his continual support and patience during the preparation of this dissertation. I am also grateful to all the members of my doctoral committee for their guidance and suggestions in the development and expression of the ideas presented in this work.

I. INTRODUCTION

The problem of automatically transcribing the hand-keyed manual morse (HKM) signal with an acceptable error rate, without exact knowledge of the sender's keying characteristics and transmitted signal parameters, has, in general, remained unsolved. The easier companion problem of automatically transcribing a Morse signal sent by a keyboard (KAM), and whose transmitted frequency is known, has largely been solved, and a number of Morse decoders are commercially available for this task. These decoders also can be used on the HKM signal, but with considerable loss in performance except in cases of very good keying quality.

The difficulty of automatically transcribing the HKM signal (problems in frequency acquisition and detection aside) is often not recognized by the uninitiated. This difficulty is analogous to that of designing an automatic speech recognition device. While the analogy cannot be taken too far, certain parallels are evident. The HKM signal, being a human-generated process, has all the characteristics of individuality associated with such a process. No two senders of Morse send in exactly the same way, just as no two speakers speak in exactly the same way. Yet a trained Morse operator can understand what is being sent, just as a person who understands the language of a speaker can understand (almost) anyone who speaks that language, whatever the individual characteristics of his speech. A

Morse transcription machine for HKM which bases its decisions solely on the local Morse symbols (dot, dash, element space, character space, word space, pause) can, with some imagination, be likened to a situation in which a person who does not know English attempts to translate a spoken English phrase by isolating the syllables of the words. Clearly the Morse transcription task is not quite so difficult as this analogy since there are only six "syllables" in Morse; yet the analogy is illustrative of the difficulty of transcribing the HKM process.

On the other hand, the KAM signal can be likened to a teletype signal with a well-defined structure. Thus it is sufficient to decode such a signal on the basis of the baud structure, since there is a one-to-one mapping from the code words to the text. This non-singular mapping accounts for the relative ease of decoding a demodulated KAM signal.

The above analogy has tacitly assumed that the Morse waveform was perfectly demodulated. In the real world of imperfect demodulation, it is clear than an HKM transcription machine which uses only local information, can provide no error-correction capability to correct incorrectly demodulated Morse symbols. Thus as a result of a single incorrect demodulation decision, an entire letter (two letters if the symbol was a character space) is transcribed incorrectly. Demodulation, therefore, must be considered as an integral part of the HKM processor, and this processor must have some

knowledge of the Morse "language" in order to provide error-correction capability.

This paper reports the results of an investigation into the problem of automatically transcribing the HKM process. The problem is attacked from the point-of-view of optimal estimation and modern information theory. Theoretical results are derived which can be directly applied to the design of an optimal HKM transcriber. It is shown that such an optimal transcriber is unrealizable in the practical sense, but that a suboptimal transcriber which can be made arbitrarily close to optimal is realizable. Lower bounds on the theoretical error-rate performance of an ideal transcriber are obtained as a function of signal-to-noise ratio, keying characteristics, and HKM model complexity. The performance of the suboptimal transcriber is obtained by computer simulation and compared to the theoretical results for the optimal transcriber. Finally, the suboptimal transcriber is tested against a limited set of field data in order to validate the simulations.

The report is organized into two parts: theoretical and application. In the theoretical section, a unified model structure for the HKM process is derived which may account for code symbol dependencies, variation in data rate, operator sending anomalies, source letter context, message format, and linguistic dependencies. A channel model is constructed to account for transmitter, propagation, and receiver effects. The resulting modeled system is shown to be a system in which the state of the HKM process evolves as a memory-conditioned probabilistic mapping of a conditional Markov process, with

the state of this process playing the role of a parameter vector of the channel and measurement models. The joint demodulation, decoding, and translation problem is then posed as finding an optimal estimate of the discrete state of the HKM signal process, given a sequence of noisy measurements of the detected signal. The Bayesian solution to this nonlinear estimation problem is obtained explicitly for the case of parameter-conditional linear-gaussian channel and measurement models, and the resulting optimal Morse transcription machine is shown to consist of a denumerable but exponentially expanding set of linear Kalman filters operating on a trellis defined by the discrete state values of the parameter vector. Because of the exponential growth, the optimal estimator is unrealizable, and a realizable suboptimal solution which adaptively restricts the growth of the trellis is obtained.

The application section shows how a specific model of the HKM process results from the general model constructed in the theoretical section. It is shown in principle how the generality of the model readily provides for any level of complexity in modeling an actual Morse message, i.e. from a very simple model of local Morse symbols up to and including a complex model of syntactic and semantic rules for the Morse "language." It is shown theoretically how context may be used to provide error-correction capability and reduce the lower-bound on output letter-error rate. Simulation results are obtained which confirm the expected improved performance for increasingly complex modeling of the Morse message.

II. PROBLEM DESCRIPTION

The statement of the problem is actually very simple: Obtain a processor which will transcribe hand-keyed manual Morse as well as a human operator. The simplicity of the statement, however, belies the complexity of describing a "hand-keyed manual Morse" signal and the difficulty of quantifying the phrase "as well as a human operator."

A. THE HAND-KEYED MANUAL MORSE (HKM) SIGNAL PROCESS

As used throughout this report, the term HKM signal refers to International Morse Code and its derivatives sent manually by key, mechanical bug, or electronic bug. The baseband HKM process is the output voltage level of the keyer and is represented by the logic levels 0 and 1, corresponding to the states "key up" and "key down." The six symbols of the code are: dot, dash, element-space, character-space, word-space, and pause. The term element (or baud) refers to the standard time unit of the code; its actual duration in seconds will of course vary with sending speed. Standard Morse code consists of the symbol durations shown in Table I.

The standard word (including word-space) in Morse communication is 50 elements in length. Thus the standard element duration in seconds for a given sending speed is 6/5 times the reciprocal of the speed in words-per-minute. The instantaneous data rate for an HKM signal is defined to be 6/5 times the reciprocal of the duration of the symbol (in

TABLE I

Standard Morse Symbols

Name	Symbol	Duration (in elements)
Dot	.	1
Dash	-	3
Element-space	^	1
Character-space	∿	3
Word-space	W	7
Pause	P	14

seconds) divided by the standard duration in elements; e.g., the instantaneous data rate for a dash of duration 60 msec is (6/5)/(1/.020) = 60 wpm.

An HKM signal differs from the standard Morse signal in that the instantaneous data rate is a random variable, resulting in symbol durations which are random. The element duration is defined to be the mean value of the dot duration; this mean value is also a random variable. The HKM signal may exhibit a large variation in both element duration and instantaneous data rate. The modeling of these random variables is discussed in section VI.A. The distributions of element duration and instantaneous data rate are unique to a particular sending operator, and in most cases depend on the type of traffic being sent, and on the intended recipient of the signal as well.

B. THE HKM SIGNAL CHANNEL

The HKM signal process is usually transmitted at HF by a transmitter whose final amplifier is on-off keyed (OOK) by the keyer, although in some cases, the oscillator itself is on-off keyed. Because of the effect of transients in the transmitter, the signal is usually chirped to some extent, the magnitude of the chirp being indicative of the quality of the transmitter design and state of maintenance. For well-designed, properly maintained transmitters, the chirp is on the order of tens of Hertz. Poorly designed or improperly maintained transmitters may exhibit as much as 300Hz chirp, as well as random drift of the nominal carrier frequency. Thus in most cases, signal detection must be accomplished by using an envelope detector since the phase of the signal is not known.

In addition to the signal uncertainties caused by the transmitter itself, the signal is also corrupted by both additive and multiplicative noise in the form of atmospherics, interference, and fading, which at HF is nonstationary. Thus demodulation of the OOK Signal must be accomplished in the face of frequency, phase, and amplitude uncertainty, along with incomplete knowledge of the noise statistics.

C. OPERATOR PERFORMANCE

The ultimate goal of the Morse transcriber is to provide output copy with an error rate approaching that which a typical human operator provides. The human operator rapidly

adapts to changing signal and channel parameters and can provide reliable copy of a highly variable HKM signal in the presence of numerous other Morse and non-Morse signals. The operator is obviously aided by an understanding of the context of the message, the format, and the Morse "language."

The available data on operator performance is summarized in Figures 1 and 2. Figure 1 is a plot of error rate vs. SNR for an actual communications link in the LF band reported by Watt et. al. [1], while Figure 2 shows the performance obtained in a laboratory experiment [2]. Both tests were conducted using random five-letter code groups as the test message. Table II, from Lane [3], shows the number of dB which must be added or subtracted from the abscissa of the performance curve to obtain the performance for different speeds of transmission. Clearly the laboratory tests show a better performance capability for the human operator than that obtained for the actual communication link, with a difference of about 2-3 dB for equal error rates. Such an observation indicates that one must design the automated transcriber using the laboratory performance measurements in order to obtain the required performance under field conditions for the same SNR.

The error rates discussed above were obtained using a text consisting of independent letters (5-letter code groups). For a text which has more structure than random letters, whether through linguistic content, known message format,

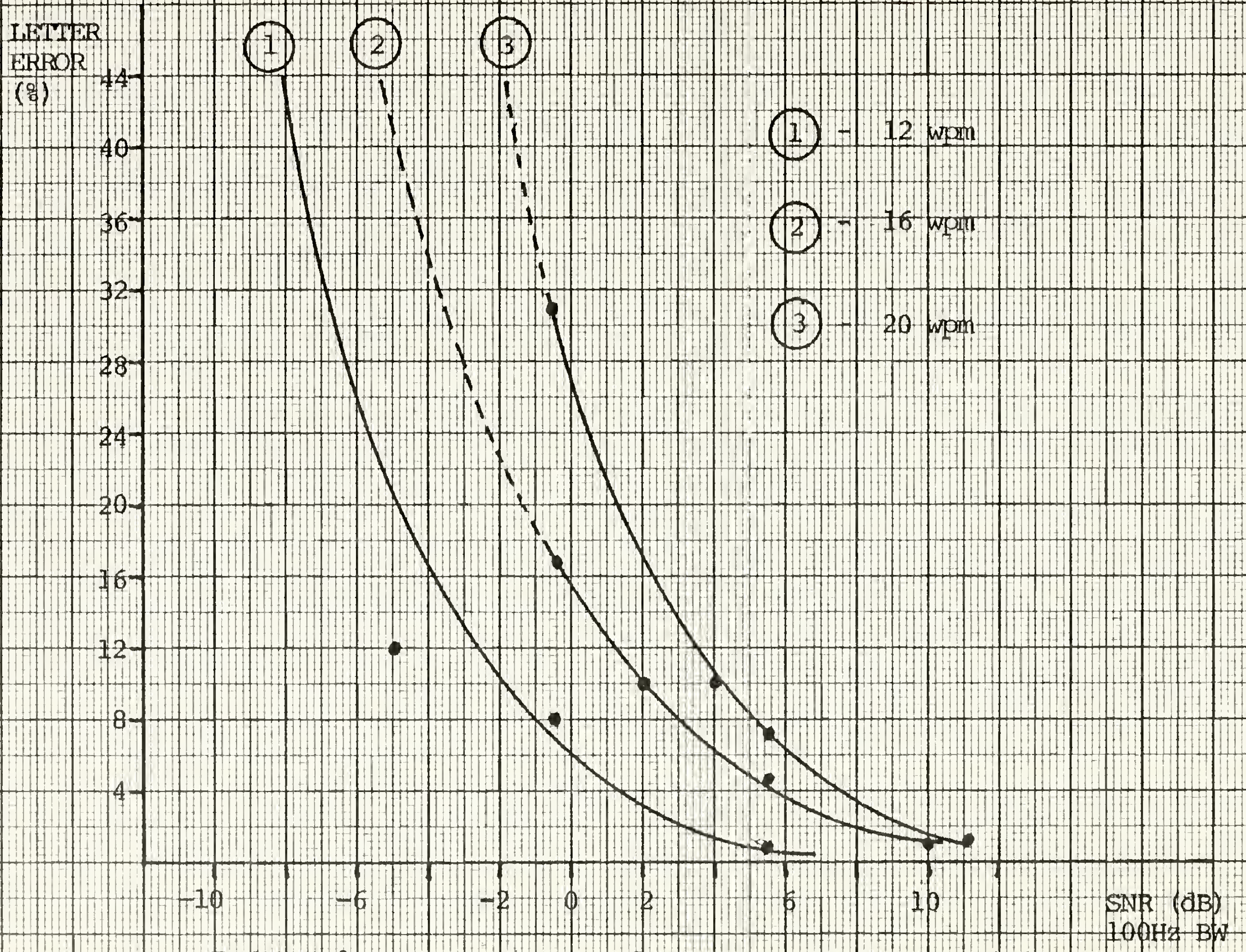

FIGURE 1. Operator Performance; LF Comm. Link, 5-Letter Code Groups (From Watt [1])

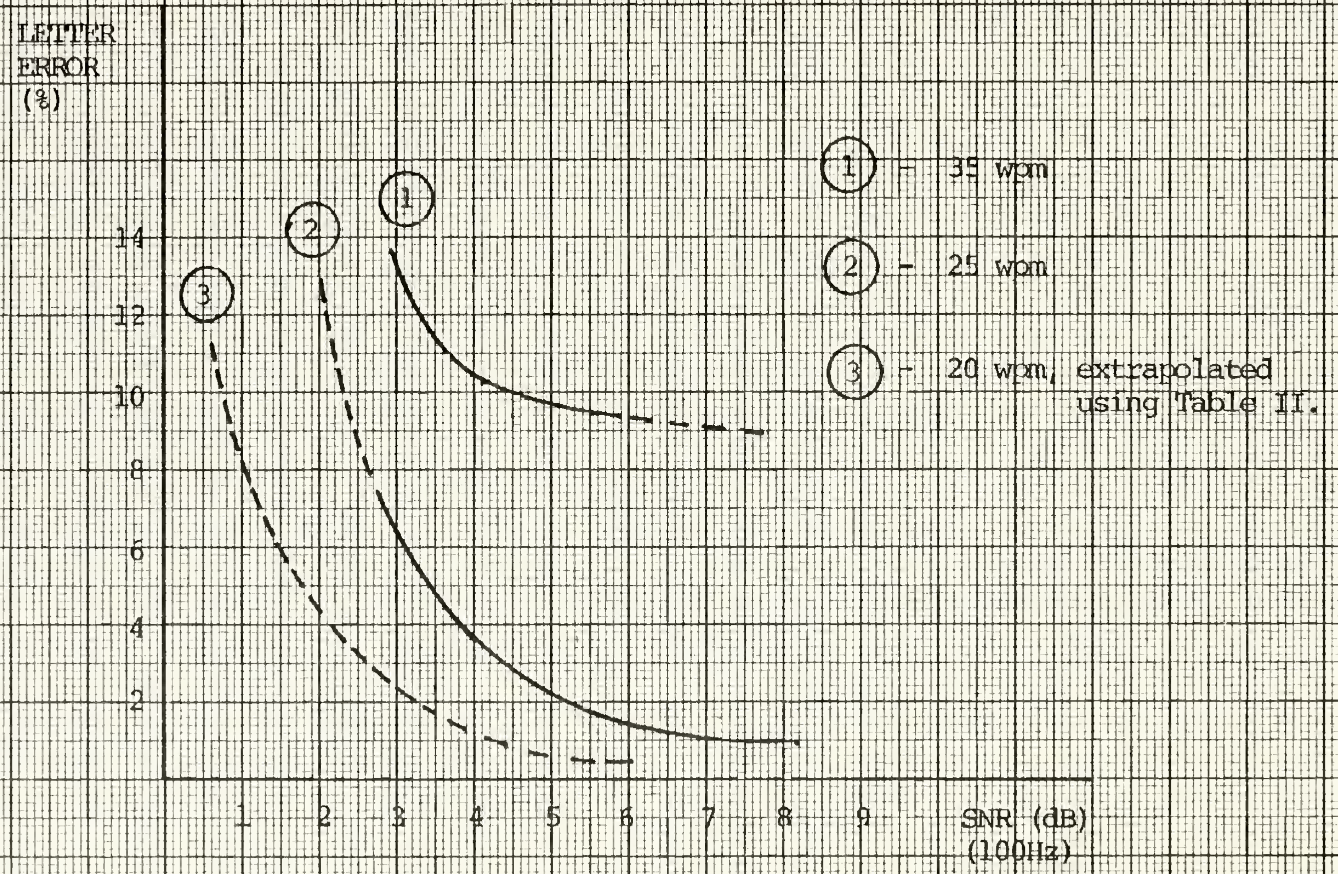

FIGURE 2. Operator Performance; Lab Results
5-Letter Code Groups, [2]

TABLE II

OPERATOR PERFORMANCE ADJUSTMENT FACTOR
FOR SENDING SPEEDS
(FROM LANE [3])

RATE (wpm)	FACTOR (dB)
10	-5.0
12	-3.6
14	-2.3
15	-1.8
16	-1.4
18	-0.6
20	0
25	1.6
30	2.6

or increased semantic content, the human operator will take advantage of the structure to effectively reduce his average error rate. His error rate, however, for those portions of a message which exhibit uncertainty equivalent to independent letters, will remain at that for independent letters. Thus although his error rate for those portions of a message which have a high information content will not decrease, the transcribed message will be much more "readable," and the more costly errors will be much easier to locate in his output copy. As an example of "readability", consider the two messages shown below, each with a 10% error rate, including spacing errors. The first message is of low information content and is readable, although with some difficulty; the second is a message with higher information content. (These

two messages were generated by using a random number generator to obtain the errors, which may not correspond to typical morse substituions.)

Message 1:

THIS IS AN RX A9P LE OF EN G LI SH TE XT
WITH AN ERROR RATE OF 10 PERCENK. THC
ERRORS INCLUDE SPA CING BETWEEN LE TTERS
AS WELL AS THE WP1D SPACE. MS CAN3 E
SEEN, THIS TEXT IS ON TH E THRESHOLDO F
ACC EPTABILRTY AN D REQUIRA 2 S1AE
DIFW8C U LTX TO R EAD.

Message 2:

BM GEZRGE P BURDELL TO JOXN BUUYEL
L123 EASW S T BEW YORK BT
PSE C ALL NAMP HO NE NO 555 1233 AND
TELL SIM WILL NOW DRR IVE KENNE DY
AVTAN 17 38 12 JU LFLT NO 63
WILL DEPANT FOX WAMH AT 231 9 12 JUL.

The obvious point of this exercise is that average letter error rate alone is not a definitive measure by which the efficiency of a transcriber (either human or machine) can be judged, except for messages consisting of random letters. Secondly, it is clear that an automatic transcriber which does not use the message context and structure (linguistics, semantics, format) to decode the received message will not

be capable of producing a transcript as readable as the human operator except for random letter texts.

III. LOWER BOUNDS ON ERROR RATE

In this section, information theoretic concepts are applied to the problem of decoding and translation of the Morse signal. Lower bounds on the performance of a transcription machine are obtained as a function of signal-to-noise ratio, keying quality, and decoder complexity. A channel model appropriate for studying the performance in this context is derived and its capacity determined. Source code models for the Morse code are also obtained, and together with the channel model, are used to derive a lower bound on decoded letter error rate. Although the average letter error rate, as argued in the previous section, is not a sufficient criterion for measuring the utility of a transcription machine in specific cases, it nevertheless provides a great deal of insight into the problem of determining how complex a decoder must be in order to approach the performance of a human operator. In order to obtain some intuitive appreciation of the Morse code as a source code, estimates of the entropy of a Morse-coded source are first determined under various assumptions about the source and the code.

A. ESTIMATION OF MORSE-CODE ENTROPY

The source entropy for a symbol-by-symbol decoder is obtained by considering the source to be an ensemble of Morse symbols each sent independently with probability equal to the expected relative frequency of occurrence of that

symbol. A decoder which is designed according to a model of the source as a Markov chain results in a source entropy calculated on the basis of that same Markov model. Thus various levels of model complexity result in corresponding levels of source entropy, as seen by the decoder. For independent symbol sequences the source entropy for an alphabet of size M is given by [4]:

$$H = - \sum_{i=1}^{M} p(i) \log p(i)$$

$p(i)$ = relative frequency of occurrence of symbol i.

For Markov sources the entropy is given by [4,p.68]:

$$H(u) = - \sum_{i=1}^{J} q(i) H(u|s=i)$$

where $q(i)$ = limiting probability of the state $s = i$;

$$H(u/s=i) = - \sum_{k=1}^{K} P_j(a_k) \log P_j(a_k)$$

$$P_j(a_k) = \Pr[u_\ell = a_k | s_\ell = j],$$

i.e. the probability that source letter a_k is produced when the Markov process is in state j at time ℓ.

1. Independent Symbols

Consider first the case of a source modeled by independent occurrences of the Morse symbols. In this case the entropy is

$$H = -P_{dot}\log P_{dot} - P_{dash}\log P_{dash} - P_{esp}\log P_{esp} - P_{csp}\log P_{csp}\,.$$

The relative frequencies of the symbols in random Morse are:

$$P_{dot} = .26, \quad P_{dash} = .24, \quad P_{esp} = .36, \quad P_{csp} = .14;$$

and the entropy is:

$$H = .26\log(.26) - .24\log(.24) - .36\log(.36) - .14\log(.14)$$

$$= 1.927 \text{ bits/Morse symbol}$$

Since there are 1.76 bauds per Morse symbol, on the average, the entropy in bits per channel digit is $H = 1.927/1.76 = 1.09$ bits.

2. First-Order Markov Process on a Symbol Basis

The independent symbol model of Morse is actually only of passing interest since even the crudest of Morse models recognizes the fact that in Morse code a mark symbol (dot or dash) must always be followed by a space symbol (esp or csp), and vice versa.

A first-order Markov model has the following approximate transistion matrix and limiting probabilities:

$$\begin{array}{l} \\ \text{dot} \\ \text{dash} \\ \text{esp} \\ \text{csp} \end{array} \begin{array}{c} \begin{array}{ccccc} \text{dot} & \text{dash} & \text{esp} & \text{csp} & q(i) \end{array} \\ \begin{bmatrix} 0 & 0 & .7 & .3 & .26 \\ 0 & 0 & .7 & .3 & .24 \\ .55 & .45 & 0 & 0 & .36 \\ .5 & .5 & 0 & 0 & .14 \end{bmatrix} \end{array}$$

Using the formulas given above for finding the entropy of a Markov source,

$$H(u|s=1) = -.7\log(.7) - .3\log(.3) = .8813$$

$$H(u|s=2) = -.7\log(.7) - .3\log(.3) = .8813$$

$$H(u|s=3) = .55\log(.55) - .45\log(.45) = .9929$$

$$H(u|s=4) = -.5\log(.5) - .5\log(.5) = 1.0$$

$$\begin{aligned} H(u) &= (.26)(.8813) + (.24)(.8813) + (.36)(.9929) + (.14)(1.0) \\ &= .938 \text{ bits/Morse symbol} \\ &= .533 \text{ bits/channel digit} \end{aligned}$$

3. <u>Second-Order Markov Process On A Symbol Basis</u>

A second-order Markov process of the Morse Code has the approximate transition Matrix and limiting state probabilities as follows:

$$\begin{array}{ccc}
 & \begin{array}{cccccccc} \cdot\wedge & \cdot\sim & -\wedge & -\sim & \wedge\cdot & \sim\cdot & \wedge- & \sim- \end{array} & q(i) \\
\begin{array}{c} \cdot\wedge \\ \cdot\sim \\ -\wedge \\ -\sim \\ \wedge\cdot \\ \sim\cdot \\ \wedge- \\ \sim- \end{array} &
\begin{bmatrix}
0 & 0 & 0 & 0 & .55 & 0 & .45 & 0 \\
0 & 0 & 0 & 0 & 0 & .5 & 0 & .5 \\
0 & 0 & 0 & 0 & .55 & 0 & .45 & 0 \\
0 & 0 & 0 & 0 & 0 & .5 & 0 & .5 \\
.7 & .3 & 0 & 0 & 0 & 0 & 0 & 0 \\
.97 & .03 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & .6 & .4 & 0 & 0 & 0 & 0 \\
0 & 0 & .97 & .03 & 0 & 0 & 0 & 0
\end{bmatrix} &
\begin{array}{c} .187 \\ .073 \\ .173 \\ .067 \\ .187 \\ .073 \\ .173 \\ .067 \end{array}
\end{array}$$

Again, using the formulas for the entropy of a Markov source, the entropy of the source for this model is found to be

$$\begin{aligned} H &= .858 \text{ bits/Morse symbol} \\ &= .488 \text{ bits/channel digit} \end{aligned}$$

4. Independent Letters

The entropy of a source which produces equally likely independent letters from an alphabet of size 36 (26 alphabet letters, 10 numerals) is

$$H = -\log\ (.02776) = 5.17 \text{ bits/ltr}$$

The average number of Morse symbols per letter is 7.27, resulting in an average entropy for the Morse symbols:

$$\begin{aligned} H_{avg} = 5.17/7.27 &= .711 \text{ bits/Morse symbol} \\ &= .404 \text{ bits/channel digit} \end{aligned}$$

5. <u>English Text</u> [5]

For a model of an English text source, producing equally independent letters, the entropy is 4.76 bits/letter. Using the proper relative frequencies for the occurrence of each letter, the entropy is reduced to 4.03. A first-order model of English has entropy 3.32, and a second order model reduces the entropy to 3.1. A model which produces equally likely words of text has an entropy of 2.14. Thus if a decoder which properly uses context, linguistics, and message structure can be designed, then the entropy of the Morse symbol for English text can be as low as 2.14/7.27

$$= .294 \text{ bits/symbol}$$

$$= .167 \text{ bits/channel digit}$$

In summary, then, it can be seen that there is considerable merit in using for design purposes a model of the encoded source based on independent or Markov letters, rather than a model based on a probabilistic description of a sequence of Morse symbols. (The various entropies are tabulated in Table III.) Given an optimal demodulator, a decoder which fully exploits the letter structure of the encoded source, then, can be expected to perform as well as the human operator for a source of independent letters. As discussed previously, however, any Morse message of significant interest does not consist of independent letters, and the human operator easily exploits the decrease in

TABLE III

ENTROPY OF MORSE CODE SYMBOLS
AND CHANNEL BITS

MODEL	MORSE SYMBOL	CHANNEL BIT
INDEP SYMBOLS	1.927	1.09
FIRST-ORDER MARKOV SYMBOLS	.938	.533
SECOND-ORDER MARKOV SYMBOLS	.858	.488
INDEP SOURCE LTRS	.711	.404
ENGLISH TEXT EQUI-PROB LTRS	.655	.372
ENGLISH TEXT FIRST-ORDER MARKOV LTRS	.457	.260
ENGLISH TEXT EQUI-PROB WORDS	.294	.167

source entropy by knowing the context, linguistics, semantics, and format of the message. Conversely, any decoder which does not exploit this decrease in source entropy can never match the capability of the human operator, although it may perform well enough in some cases to be of value.

B. IDEALIZED HKM CHANNEL MODEL

Since the objective here is to obtain lower bounds on error rate, and not an estimate of actual performance, it is appropriate to consider an idealization of the HKM process, the detection process, and optimum demodulation in the presence of white gaussian noise. As such, the output of the detector would be input to a matched filter whose integration time is equal to the element duration of the Morse code being received. Exact knowledge of the baud length is assumed in order that the matched filter can remain in synchronism with the incoming signal. Obviously no decoder for HKM can ever have such information with certainty, thus this idealization represents the best possible demodulator which can never be achieved in practice. Secondly, the error crossover probabilities (dot vs. dash; element-space vs. character space) are idealized to be discrete probabilities rather than considering duration densities for these symbols; the word-space is included as a source letter and the pause symbol is ignored for this analysis. Under these simplifying assumptions, the channel can be modeled as a discrete symmetric channel, as shown in Figure 3.

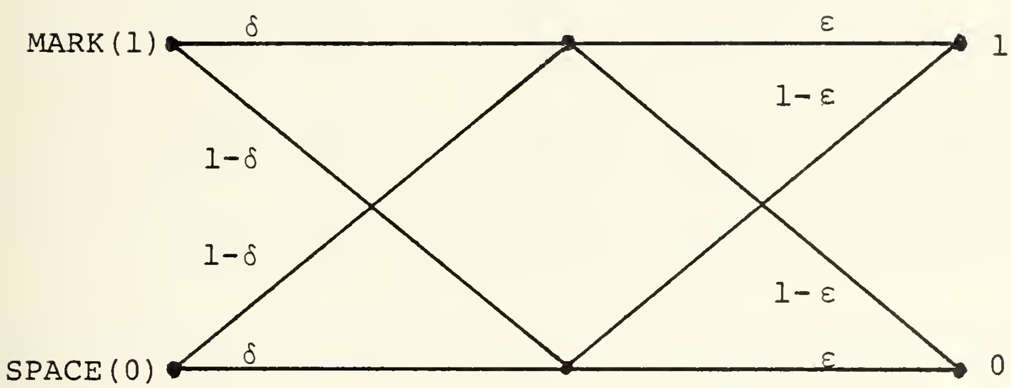

Figure 3. Idealized HKM Channel Model

In this model, the crossover probability δ is related to the Morse symbol crossover probability by defining δ to be the probability which yields the same average letter error rate as the symbol crossover probability on the basis of an average encoded letter. Since the average letter of Morse code consists of 7 symbols and 12 channel bits, δ is defined by the relationship

$$\bar{E}_s \triangleq (1 - \delta)^{12} = (1 - P_{es})^7$$

where $\bar{E}_s$ is the average sending letter error rate and P_{es} is the corresponding symbol error crossover probability. It will be convenient to make the following definitions on the keying quality of a HKM signal:

GOOD: $\bar{E}_s = .01$ ($P_{es} = .00143$, $\delta = .000837$)

FAIR: $\bar{E}_s = .1$ ($P_{es} = .0149$, $\delta = .00874$)

POOR: $\bar{E}_s = .25$ ($P_{es} = .0403$, $\delta = .0237$)

that is, a good sending operator sends the Morse symbols such that the resulting code stream consists of encoded letters in which 1% contain at least one incorrect Morse symbol; a fair operator sends with a 10% error rate; and a poor operator sends with a 25% error rate.

The crossover probability ε is just $1 - P_d$, where P_d is the probability that the matched-filter demodulator announces the correct mark/space decision. This probability is obtained as a function of SNR by computing E_b/N_o, where E_b = signal energy during an element duration and N_o = one-sided noise spectral density. The error probability ε is then obtained from the performance curve for the probability of error using either coherent or envelope detection, as appropriate, followed by a matched filter [6].

The channel shown in Figure 3 may be converted to the equivalent binary symmetric channel shown in Figure 4 by

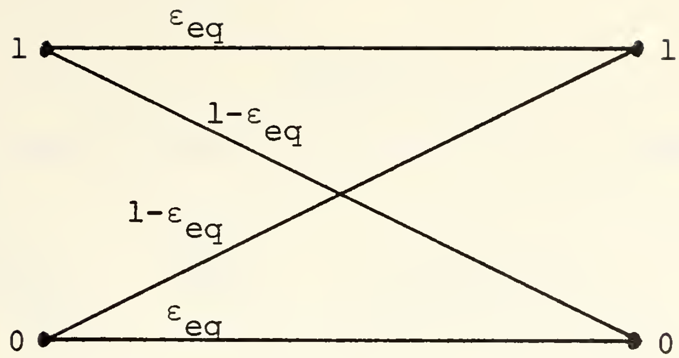

Figure 4. Equivalent HKM BSC

defining the equivalent crossover probability, ε_{eq}:

$$\varepsilon_{eq} \triangleq p(1/0) \equiv p(0/1) = \varepsilon + \delta - 2\delta\varepsilon$$

Clearly if $\delta = 0$ (perfect keying), then $\varepsilon_{eq} = \varepsilon$, and if

$\varepsilon = 0$ (perfect demodulation), then $\varepsilon_{eq} = \delta$.

Since this channel is symmetric, capacity is achieved by assigning equiprobable input binary symbols, and is given by

$$C = 1 + \varepsilon_{eq} \log \varepsilon_{eq} + (1 - \varepsilon_{eq}) \log (1 - \varepsilon_{eq}).$$

Table IV gives the channel capacity as a function of signal speed and SNR for the KAM signal using envelope detection.

C. CALCULATION OF LOWER BOUNDS FOR LETTER-ERROR PROBABILITY

A lower bound average letter error rate is easily obtained by using the Straight-line Bound for a binary symmetric channel [4, p. 163]. To use this bound, it is necessary to know the number of codewords in the code, and the length

TABLE IV

HKM Channel Capacity as Function of Speed and SNR

Speed (wpm)	SNR (dB) (100Hz)	E/No (dB)	$1-P_d$ (Envelope Det)	C
50				
	12	15.8	2×10^{-5}	~1.0
	9	12.8	2.5×10^{-3}	.975
	6	9.8	2.7×10^{-2}	.821
	3	6.8	1.1×10^{-1}	.500
	0	3.8	2.3×10^{-1}	.222
30				
	12	18	$< 10^{-5}$	~1.0
	9	15	1.3×10^{-4}	.998
	6	12	6×10^{-3}	.947
	3	9	4.5×10^{-2}	.735
	0	6	1.3×10^{-1}	.443
20				
	12	19.8	$< 10^{-5}$	~1.0
	9	16.8	$< 10^{-5}$	~1.0
	6	13.8	7×10^{-4}	.992
	3	10.8	1.6×10^{-2}	.882
	0	7.7	8×10^{-2}	.598

(in binary digits) of the codewords. Additionally this bound only applies to stationary block codes, requiring construction of an equivalent stationary block code for Morse, which in reality is a code which produces variable length word sequences. Given an equivalent block code the appropriate relationship for the probability of codeword error, P_e, is given by:

$$P_e > \left[\binom{N}{k} - \frac{1}{M}\sum_{m=1}^{M} A_{k,m}\right] \varepsilon_{eq}^{k}(1-\varepsilon_{eq})^{N-k}$$

$$+ \sum_{n=k+1}^{N} \binom{N}{n} \varepsilon_{eq}^{n} (1-\varepsilon_{eq})^{N-n} ,$$

where

N = codeword length

M = no. of codewords

$$A_{n,m} = \begin{cases} \binom{N}{n}; & 0 \leq n \leq k-1 \\ 0\ ; & k+1 \leq n \leq N \end{cases}$$

and k is chosen so that

$$M \sum_{n=0}^{k-1} \binom{N}{n} + \sum_{m=1}^{M} A_{k,m} = 2^N; \qquad 0 < \sum_{m=1}^{M} A_{k,m} \leq M \binom{N}{k} .$$

This result for P_e is for a block code with M codewords, each of length N bits transmitted over a BSC with error probability ε_{eq}. The problem then is to construct a block code which is equivalent, in some sense, to the variable-length-codeword Morse code, then to determine the number of codewords and the length of the codewords for this equivalent code. Clearly the complexity of this equivalent block code will depend on how one chooses to model the human Morse-encoding process for the design of the decoder, i.e., encoding

symbol-by-symbol; symbol pairs, triplets, etc., letter-by-letter, letter pairs, 3-letter words, 5-letter words, etc. Additionally the codewords must be chosen so that the resulting encoded sequences are stationary in order to state that the statistical expectation represented by P_e is the same as the expected letter error rate (expectation over time). This stationarity can be ensured by requiring the encoded sequence to begin at a random point within a source letter [7]. Such a requirement is equivalent to stating that the decoder is not synchronized with the encoder on a letter basis; that is, the decoder has no a-priori knowledge of the beginning and ending of a letter of the variable-length word sequence produced by the Morse code.

Consider first the construction of an equivalent block code for Morse which is assumed to be encoded as a symbol pair. Table V shows the variable-length Morse codewords for this code. An equivalent set of equal length block codewords, on the basis of equal average codeword length, is shown in Table VI. It is to be noted that some codewords cannot follow other codewords in an encoded sequence. For example, the sequence 101011 cannot be followed by any codeword except those beginning with 10 since the sequence 11 and the sequence 1111 are not allowable Morse sequences.

In principle, the same procedure can be followed to obtain the set of codewords for any desired codeword length.

TABLE V

Variable-Length Codewords For Symbol Pairs

Morse Symbol	Channel Code
•^	10
−^	1110
•∿	1000
−∿	111000
^•	01
^−	0111
∿•	0001
∿−	000111

Average No. of Channel Bits Per Morse Codeword: 4

TABLE VI

Equivalent Four-Bit Channel Mode For Symbol Pairs

0000	1000
0001	1010
0010	1011
0011	1100
0100	1101
0101	1110
0111	

No. of Codewords: 13

For sequence lengths greater than about 12, however, the sheer number of possibilities makes this procedure intractable. For obtaining codeword sets for an encoder which encodes combinations of more than one source letter at a

time, then, another procedure is used. Although this procedure does not obtain all the codewords in the equivalent block code set, it obtains almost all of them and thus represents a lower bound on the actual number of codewords.

The average Morse code sequence is 7.27 symbols in length. For a Morse code, however, the sequence length in Morse symbols must be an even number (it must begin with a mark and end with a character space). By choosing an average of 8 symbols/character for the equivalent block code, and by requiring that the 8th symbol be a character-space, then, it can be seen that it is impossible to produce a sequence of a Morse symbols which does not represent some character. It is also obvious that not all characters are represented by this code. Now, of the four symbols, only two are allowed in any one position of the sequence (since space follows mark invariably and vice versa) thus the possible number of <u>synchronous</u> Morse sequences on this basis is $2^7 = 128$, and the minimum length of the codewords in binary digits is $8 \times 1.76 \cong 14$. To obtain the full set of <u>nonsynchronous</u> codewords, each codeword is shifted one bit at a time and a one or zero appended, if allowable, until no new codewords are produced. To illustrate, consider the synchronous codeword 10111011101000. By right shifting and appending a zero and one respectively, the two additional codewords 01011101110100 and 11011101110100 are obtained. On the next shift, note that the sequence 0110 is not legal,

so only three additional codewords are obtained: 1010..., 0010..., and 1110.... In general, those codewords beginning with a dot (10) produce eleven additional codewords, and the codewords beginning with a dash (1110) produce eight additional codewords. If M_s = number of synchronous codewords, then $M_s/2$ = no. of codewords beginning with a dot (dash), so the total number of nonsynchronous codewords is given by

$$M = 19\ M_s/2 + M_s = 10.5\ M_s$$

Table VII gives the number of binary codewords (M) and the codeword length (N) for the encoding procedure of interest. For $N \leq 12$, M and N are exact, as computed by the first procedure discussed above. For $N > 12$, M and N are lower bounds obtained by the second procedure. Using these values of M and N, the lower bound on P_e as a function of ε_{eq} is obtained. This value for P_e is the error rate over a code of M codewords, and for the case of single character encoding, is the same as the average letter error rate. For other cases of source alphabet models, however, P_e does not represent the letter error rate, since letters consist of more or fewer than one codeword depending on the length of the codeword. To determine the letter error rate, $\bar{E}_\ell$, consider the following arguments.

TABLE VII

Equivalent Block Codeword Set Size And Length For Morse Code

Encoder	M	N
Symbol Pair	13	4
3-symbol	33	6
Single letters (exact)	395	12
Single letters (bound)	1,344	14
Double Letters	139,264	28
3-letter words	22,020,096	42

Case 1: Letters consisting of two or more codewords.

For this case, the distribution of codeword error events per letter is binomial with parameter P_e. Let m be the number of codewords per letter. Then the probability of exactly k error events per letter is given by $\binom{m}{k} P_e^{\ k} (1 - P_e)^{m-k}$, and the probability of at least one error event per letter (i.e. the probability of a letter error) is given by $\bar{E}_\ell = 1 - (1 - P_e)^m$.

Case 2: Codewords consisting of n letters.

In this case, $\bar{E}_\ell$ is lower bounded by assuming that a codeword error event causes a single letter error within the codeword; then $\bar{E}_\ell = P_e/n$.

Figures 5-7 show plots of the lower bound on average letter error rate, $\bar{E}_\ell$, as a function of SNR and keying quality for several levels of assumption about the Morse encoding process.

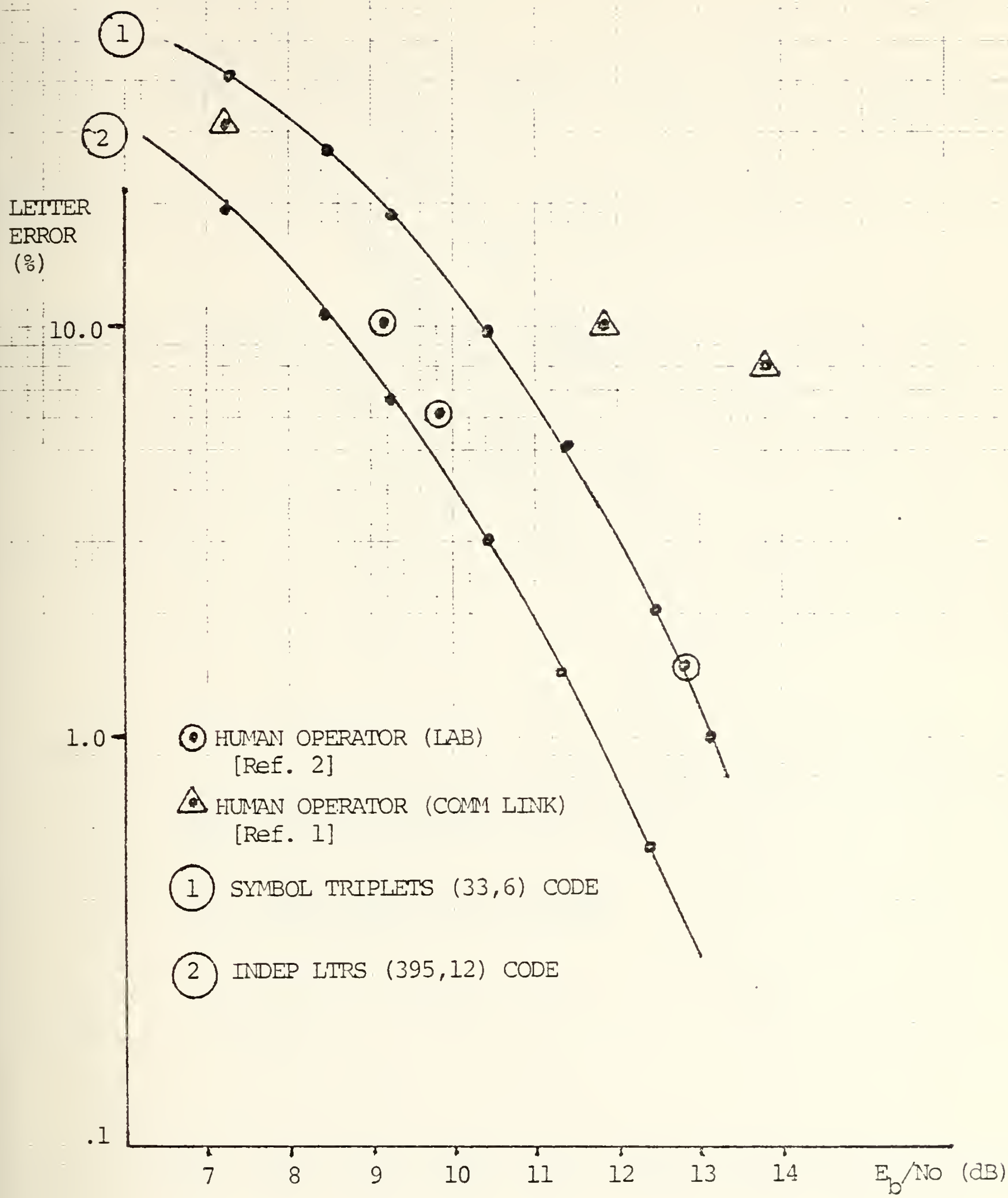

FIGURE 5. Lower Bounds on Letter Error Rate for Morse Code - KAM Signal, Coherent Detection

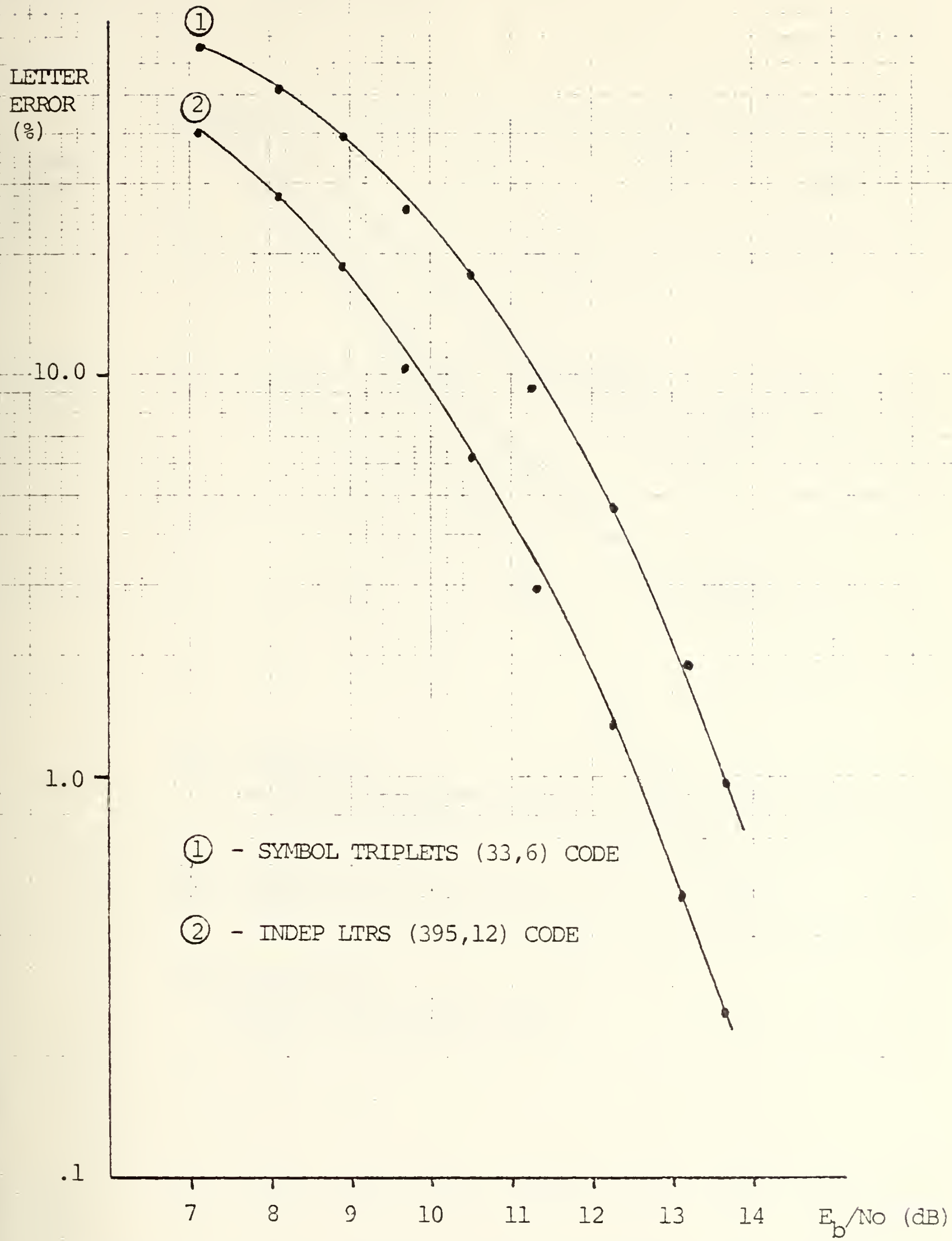

FIGURE 6. Lower Bounds on Letter Error Rate for Morse Code - KAM Signal, Envelope Detection

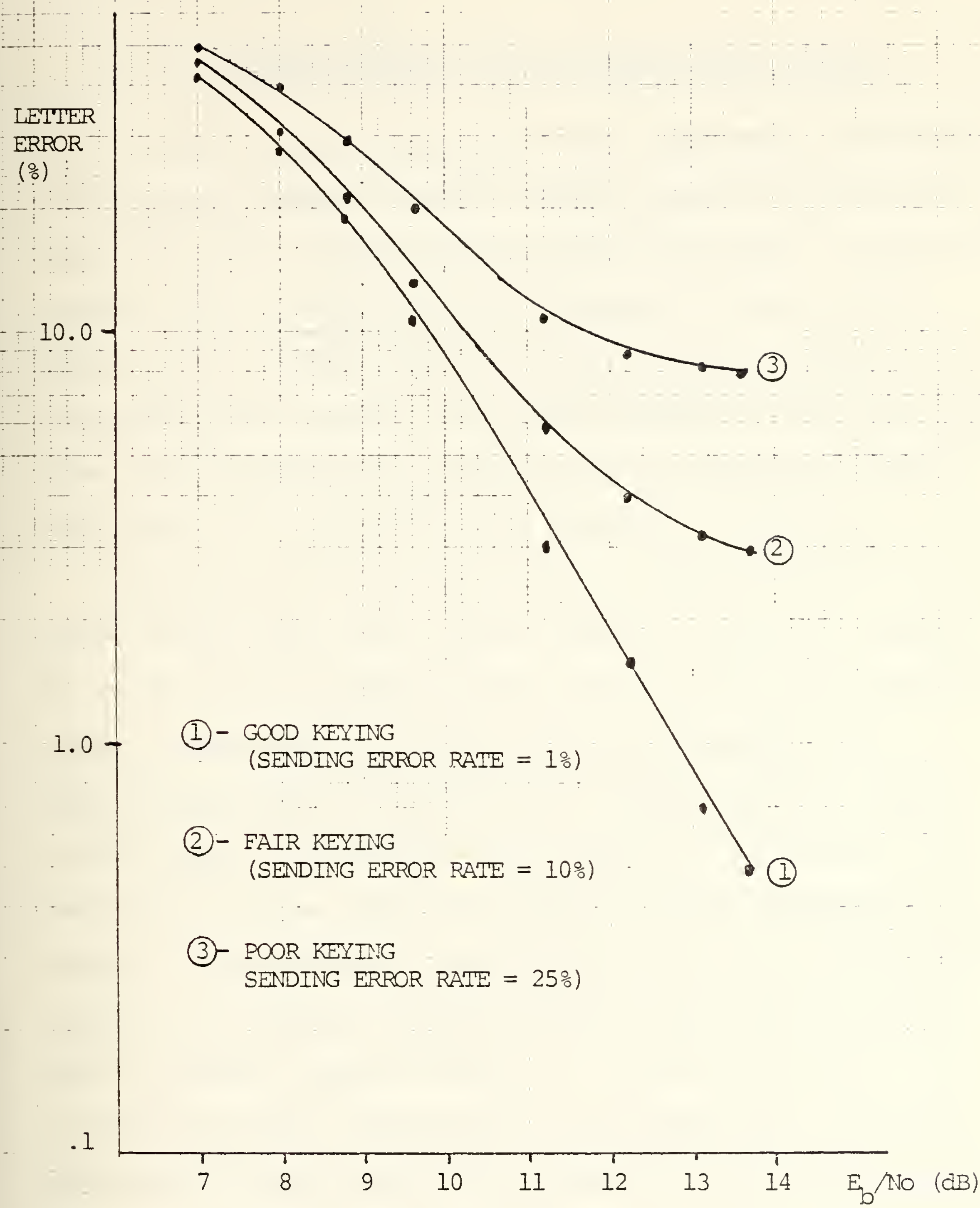

FIGURE 7. Lower Bounds on Letter Error Rate for Hand-Keyed Morse, Envelope Detection, Random Letter Source

IV. A GENERAL MODEL FOR THE HKM SIGNAL PROCESS

In this section, a general model structure which accounts for message context, sender operator errors, variation in date rate, and variability of element duration is constructed. Further it is shown that various special cases of this model result in processes for which optimum estimation algorithms and decoders have been treated in the literature, some from the point of view of optimal estimation theory and others from an information theoretic viewpoint.

Fundamentally the model that is constructed is a sliding block coder (SBC) with infinite memory. However, instead of encoding the letters of the text into the Morse symbols either noiselessly or with a fidelity criterion, the encoding process is considered as a probabilistic mapping of the output of the SBC. The complexity of the SBC is determined by the degree to which the Morse message is desired to be modeled, from the simplest case of independent symbols to a highly complex syntatic and semantic model. While specific complex models of a Morse message are not developed in this investigation, the structure for implementation of such models is provided by the general model. Thus the structure proposed represents a unified approach to modeling the Morse message from the simplest case to the most complex.

A. BASEBAND HKM SIGNAL PROCESS

The desired representation of the discrete-time baseband HKM process is a sequence of 1's and 0's whose pattern of occurrence closely resembles that of a human operator sending a Morse text. By considering intuitively how a sending operator may encode the letters of the text, the random variables which influence the human encoding procedure can be recognized. Figure 8 is useful for visualizing this process.

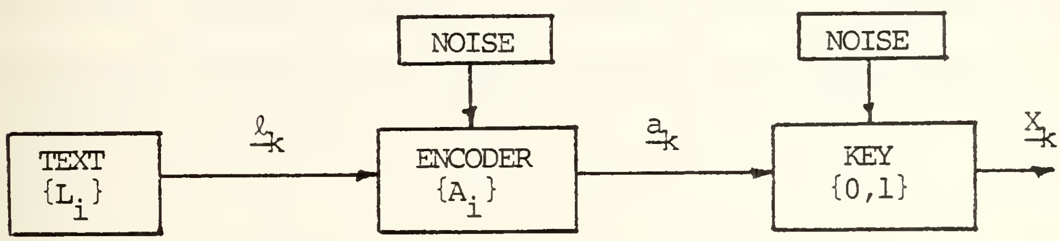

Figure 8. Morse Encoding Process

At some time k, one or more letters of the text, $\underline{\ell}_k$, are encoded into a sequence of code words $\underline{a}_k$, consisting of the Morse symbols. The human operator, however, does not always send the proper Morse sequence for a given sequence of letters; typical mistakes are insertions and deletions of one or more symbols (particularly dots), and substitutions of one symbol for another (particularly word-spaces for

character-spaces, and character-spaces for element-spaces). Additionally the speed at which he is sending may vary over a period of time, depending on his alertness, proficiency, fatigue and the importance of the traffic being sent.

The key converts these symbols into the 0,1 logic levels of duration consistent with the particular Morse symbol being sent. The length of time that the key is in a 0 or 1 state, however, while determined principally by the Morse symbol being sent, is a random variable since the human operator cannot always produce repeatable, precise durations. The variability of the durations for each symbol, again, is dependent on the operator's proficiency, alertness, and individual sending habits. Consideration of these random influences leads to the model which is now developed.

Let

$x_k \in \{K_i;\ i = 1,2\}$, the set of keystates;

$a_k \in \{A_i;\ i = 1,2,\ldots 6\}$, the set of code symbols;

$\ell_k \in \{L_i;\ i = 1,2,\ldots N\}$, the set of source letters.

Further, define the following finite state memory functions:

(1) $\beta_k = f_\beta(x_k, \beta_{k-1})$, the memory associated with keying;

(2) $\alpha_k = f_\alpha(a_k, \alpha_{k-1})$, the memory associated with encoding;

(3) $\lambda_k = f_\lambda(\ell_k, \lambda_{k-1})$, the memory associated with the source,

where

$\beta_k \; \varepsilon \; \{B_i;\; i = 1,2,\ldots\}$, the set of key memory states;

$\alpha_k \; \varepsilon \; \{A_i;\; i = 1,2,\ldots\}$, the set of encoder memory states;

$\lambda_k \; \varepsilon \; \{M_i;\; i = 1,2,\ldots\}$, the set of source (message) states.

Then the state of the process at time k is specified by the vector:

$$\begin{bmatrix} \underline{s}_k \\ \underline{\sigma}_k \end{bmatrix} \triangleq [x_k, a_k, \ell_k, \beta_k, \alpha_k, \lambda_k]^T,$$

where

$$\underline{s}_k \triangleq [x_k, a_k, \ell_k]^T, \qquad \underline{\sigma}_k \triangleq [\beta_k, \alpha_k, \lambda_k]^T.$$

For example, if f_β counts the number of samples since the last keystate transition, f_α counts the number of symbols

sent since the last letter transition and f_λ records the previous letter, then a specification of the state vector gives the current key state, code symbol, and letter being sent, along with the amount of time the key has been in its current state, which symbol of the Morse code sequence for the letter is being sent, and the previous letter.

To introduce the randomness associated with sending errors and variation in data rate, let a random control vector be defined which selects the Morse code sequence for the letter being transmitted, controls the instantaneous data rate, and the average speed of sending:

$$\underline{u}_k \in \{\underline{U}_i;\ i = 1,2,\ldots M\}, \quad \text{the set of control vectors.}$$

The complete state vector is now given by

$$\begin{bmatrix} \underline{s}_k \\ \underline{u}_k \\ \underline{\sigma}_k \end{bmatrix} = [x_k\ a_k\ \ell_k\ \underline{u}_k^T\ \beta_k\ \alpha_k\ \lambda_k]^T$$

The probabilistic evolution of the states of the process will be fully specified when the following transition probabilities are determined:

$$\Pr[\underline{s}_k = S_i,\ \underline{u}_k = \underline{U}_j,\ \underline{\sigma}_k = \underline{\Sigma}_m | \underline{s}_{k-1} = \underline{S}_n,\ \underline{u}_{k-1} = U_p,\ \underline{\sigma}_{k-1} = \underline{\Sigma}_q]$$

where

$\{S_i;\ i = 1,2,\ldots R\}$ is the set of all state values,

and

$\{\Sigma_i;\ i = 1,2,\ldots Q\}$ is the set of all memory states.

This state transition probability matrix is now derived in terms of the components of the vector $\underline{s}_k$.

Let the evolution of the keystate, which is dependent only on its present and past inputs and its past outputs be described by the transition probabilities:

$$(4)\quad p(x_k | a_k\ \alpha_{k-1}\ \beta_{k-1}) \triangleq Pr[x_k = K_i | a_k = A_j,\ \alpha_{k-1} = A_m,\ \beta_{k-1} = B_k]$$

Similarly the evolution of the encoded letters a_k from the decoder is dependent on the present and past inputs to the encoder and on its past outputs, but it is also dependent on the history of the keystate, since the code symbol being keyed cannot be changed until the current symbol has completed keying. The transition probabilities describing the encoder function then are given by:

$$(5)\quad p(a_k | u_k\ \ell_k\ \lambda_{k-1}\ \alpha_{k-1}\ \beta_{k-1}) \triangleq Pr[a_k = A_i | u_k = U_j,$$
$$\ell_k = L_m,\ \lambda_{k-1} = M_n,\ \alpha_{k-1} = A_p,\ \beta_{k-1} = B_q].$$

The evolution of letters from the source is dependent on the history of the message text, but it is also dependent on the history of the encoding process, since the letter being encoded cannot be changed until the current letter has completed the encoding procedure. The transition probabilities for the source then are:

$$(6)\quad p(\ell_k | \lambda_{k-1}\ \alpha_{k-1}) \triangleq Pr[\ell_k = L_i | \lambda_{k-1} = M_j,\ \alpha_{k-1} = A_m].$$

The control vector u_k is modeled as a conditional Markov chain, conditioned on $\alpha_{k-1}, \beta_{k-1}, \lambda_{k-1}$, accounting for the dependence of operator sending peculiarities and data rate on message context, message duration, traffic type, etc. The transition probabilities for this model are:

$$(7)\quad p(\underline{u}_k | \underline{u}_{k-1}\ \alpha_{k-1}\ \beta_{k-1}\ \lambda_{k-1}) \triangleq Pr[\underline{u}_k = \underline{U}_i | \underline{u}_{k-1} = \underline{U}_j,\ \alpha_{k-1} = A_m,\ \beta_{k-1} = B_n,\ \lambda_{k-1} = M_p]$$

In terms of the abbreviated notation defined by expressions (4) through (7) above, the state transition matrix is given in terms of the components of the state vector s_k by:

$$p(\underline{s}_k\ \underline{u}_k\ \underline{\sigma}_k | \underline{s}_{k-1}\ \underline{u}_{k-1}\ \underline{\sigma}_{k-1}) \equiv p(x_k\ \beta_k\ a_k\ \alpha_k\ \ell_k\ \lambda_k\ \underline{u}_k | x_{k-1}\ \beta_{k-1}\ \alpha_{k-1}\ \ell_{k-1}\ \lambda_{k-1}\ \underline{u}_{k-1}).$$

Invoking the independence of appropriate variables argued in writing expressions (4) - (7), this expression reduces by the chain rule to:

$$(8) \quad p(\underline{s}_k\ \underline{u}_k\ \underline{\sigma}_k | \underline{\sigma}_{k-1}\ \underline{u}_{k-1}) = p(x_k | a_k\ \beta_{k-1}\ \alpha_{k-1}) \cdot p(\beta_k | x_k\ \beta_{k-1})$$

$$\cdot\ p(a_k | \ell_k\ \underline{u}_k\ \alpha_{k-1}\ \lambda_{k-1}\ \beta_{k-1}) \cdot p(\alpha_k | a_k\ \alpha_{k-1})$$

$$\cdot\ p(\ell_k | \lambda_{k-1}\ \alpha_{k-1}) \cdot p(\lambda_k | \ell_k\ \lambda_{k-1})$$

$$\cdot\ p(\underline{u}_k | \underline{u}_{k-1}\ \alpha_{k-1}\ \beta_{k-1}\ \lambda_{k-1}).$$

Now the expressions for the transition probabilities of β_k, α_k, λ_k are given by the following due to definitions (1) - (3):

$$p(\beta_k | x_k\ \beta_{k-1}) = \begin{cases} 1, & \text{if } B_i = f_\beta(K_j, B_n) \\ 0, & \text{otherwise} \end{cases}$$

$$p(\alpha_k | a_k\ \alpha_{k-1}) = \begin{cases} 1, & \text{if } A_i = f_\alpha(A_j, A_n) \\ 0, & \text{otherwise} \end{cases}$$

$$p(\lambda_k | \ell_k\ \lambda_{k-1}) = \begin{cases} 1, & \text{if } M_i = f_\lambda(L_j, M_n) \\ 0, & \text{otherwise} \end{cases}$$

Thus the transition probability (8) is zero for unallowable transitions, where the set of allowable transitions is given by (1) - (3). The expressions for the state transition probabilities (8), then, may be written as

(9a) $$p(\underline{s}_k\ \underline{u}_k | \underline{u}_{k-1}\ \underline{\sigma}_{k-1}) =$$

$$p(x_k | a_k\ \beta_{k-1}\ \alpha_{k-1}) \cdot p(a_k | \ell_k\ \underline{u}_k\ \alpha_{k-1}\ \lambda_{k-1}\ \beta_{k-1})$$

$$\cdot\ p(\ell_k | \lambda_{k-1}\ \beta_{k-1}) \cdot p(\underline{u}_k | \underline{u}_{k-1}\ \alpha_{k-1}\ \beta_{k-1}\ \lambda_{k-1})$$

where the set of allowable transitions is given by

(9b) $$\underline{f}_{\underline{\sigma}}(\underline{s}_k, \underline{\sigma}_{k-1}) \triangleq [f_\beta(x_k, \beta_{k-1})\ f_\alpha(a_k, \alpha_{k-1})\ f_\lambda(\ell_k, \lambda_{k-1})]^T.$$

Expression (9), then is the desired description of the probabilistic evolution of the state of the HKM process, given in terms of the source (message) statistics, Morse encoding procedure, keying characteristics and data rate statistics.

This model for the HKM process accounts for many effects which go into the generation of the key output logic levels. The extent to which the model accurately represents a Morse code stream is determined by the complexity of the memory functions f_λ, f_α, f_β and by the proper assignment of the conditional transition probabilities.

For example, if the f_λ function is sufficiently complex and clever, the entire past context of a message may be accounted for in assignment of the letter transition probabilities. In the simplest case, the assumption is made that $f_\lambda \equiv 0$, and uniform probabilities are assigned to the letter transitions. The next level of complexity is to assume that $f_\lambda = \ell_{k-1}$, allowing a Markov model for the letter transition probabilities. Considerably more complex is a model which recognizes that certain sequences of letters are always followed by a known sequence in certain formatted messages. The most sophisticated model for this function is one which models the structure of the Morse code message as a natural language, requiring construction of syntatic and grammar-like rules which are used to parse the message into meaningful sequences of letters and words. Such a model would obviously require a highly complex f_λ.

At the next level, that of encoding the letters into the mark/space durations consistent with the dot/dash/space Morse sequence for the letter, any level of sophistication and cleverness for the f_α function may be used, together with the model for the vector control variable $\underline{u}$. It is at this point that operator inconsistencies such as deletion, substitution and insertion of Morse elements can be accounted for. Additionally, by proper construction of the f_α function, one may also account for variations in weight (average dot/elem-space ratio), sending speed, and known conditional

relationships between the ratios of current to predecessor element durations. In the simplest case, the assumption is made that the operator always encodes perfectly and that his element durations are consistent. This simple case would apply to machine-sent Morse code and corresponds to the situation where $\underline{u}$ = constant, and $f_\alpha = a_{k-1}$.

At the key, the durations a_k are converted into the 0,1 logic levels of duration roughly equal to that produced by the encoder. The human, however, cannot always produce these durations consistently; thus, the time duration in a particular state will be random, with mean value roughly equal to the durations produced by the encoding process, and with a variance inversely proportional to his proficiency and concentration. There are, for example, certain conditional relationships which have been found to be true for almost every operator; in particular, inter-element dots are more consistently produced than beginning or ending dots.

At this point, also, the effect of the type of key used by the operator may be accounted for. Hand-keys, mechanical bugs, and electronic bugs all produce different duration statistics for the same operator with the same message.

The purpose of this research is not to derive sophisticated models for the f-functions, but to derive a result which shows in general, whatever model is used, how the concepts of context, message formatting, operator encoding anomalies, and operator "fist" modeling may be included in a unified framework to produce at the receiver an optimal

estimate of the transmitted text. The extent to which the output translated text is an accurate reproduction of the transmitted message is clearly a function of the sophistication and accuracy of the model used.

The results of this development of the model are summarized in the following simple theorem.

Theorem

Let S_k be an n-dimensional discrete-valued random vector with finite state-space: $\{S_i;\ i = 1,2,\ldots N\}$.
Let U_k be an m-dimensional discrete-valued random vector with finite state-space: $\{U_i;\ i = 1,2,\ldots M\}$.
Let Σ_k be an r-dimensional discrete-valued random vector with finite state-space: $\{\Lambda_i;\ i = 1,2,\ldots R\}$.

Define the function f_σ: $S_k \times \Sigma_k \rightarrow \Sigma_k$ such that $\sigma_k = f_\sigma(s_k, \sigma_{k-1})$, where s_k, σ_k are realizations of the random processes S_k, Σ_k, respectively.

Let the probabilistic evolution of the U_k process be described by the following conditional Markov process:

$$p(u_k | u_{k-1}\ \sigma_{k-1}) \triangleq \Pr[u_k = U_j | u_{k-1} = U_m,\ \sigma_{k-1} = \Lambda_\ell]$$

$$\text{all } j,\ m,\ \ell.$$

Let the probabilistic evolution of the S_k-process be described by the following conditional probabilistic mapping of the U_k-Markov process:

$$p(s_k | u_k \; u_{k-1} \; \sigma_{k-1}) \triangleq \Pr[s_k = S_i | u_k = U_j, \; u_{k-1} = U_\ell,$$

$$\sigma_{k-1} = \Lambda_n], \quad \text{all } i, j, \ell, n.$$

Then, the output state s_k of the HKM process described by equation (9) results from a probabilistic mapping of the Markov control vector u_k, conditioned on the entire past history of the output state.

Proof:

First, it is clear that the function f_σ records the past history of the output state s_k, since

$$\sigma_k = f_\sigma(s_k, \sigma_{k-1}) \equiv f_\sigma(s_k, f_\sigma(s_{k-1}, \sigma_{k-2}))$$

$$\equiv f_\sigma(s_k, f_\sigma(s_{k-1}, f_\sigma(s_{k-2}, \; \ldots \; f_\sigma(s_1, \sigma_0)) \ldots).$$

Second, expression (9a) reduces by the chain rule to:

$$P(s_k \; u_k | u_{k-1} \; \sigma_{k-1}) = p(s_k | u_k \; u_{k-1} \; \sigma_{k-1}) \cdot p(u_k | u_{k-1} \; \sigma_{k-1}).$$

Corresponding the terms on the right-hand side with the $\mathcal{S}_k$, $\mathcal{U}_k$ processes described above, and expression (9b) with the f_σ function, the theorem is proved.

Corollary

Let the function f_σ be invertible in the sense that $s_k = f_{\sigma s}^{-1}(\sigma_k, \sigma_{k-1})$ is uniquely defined.

Then the output state σ_k of the HKM process is a sliding block encoding of the sequence $s_0, s_1, s_2 \ldots s_k$, where the evolution of the S_k process is described by the conditional mapping:

$$p(s_k | u_{k-1}\ \sigma_{k-1}) \triangleq \Pr[s_k = S_i | u_{k-1} = U_j,\ \sigma_{k-1} = \Lambda_m]$$

and the U_k process is described by:

$$p(u_k | u_{k-1}\ \sigma_{k-1}\ \sigma_k) \triangleq \Pr[u_k = U_i | u_{k=1} = U_j, \sigma_{k-1} = \Lambda_m,$$

$$\sigma_k = \Lambda_n].$$

Proof: From the main theorem, the state σ_k is describeable as:

$$\sigma_k = f_\sigma(s_k, f_\sigma(s_{k-1}, f_\sigma(s_{k-2}, \ \ldots\ f_\sigma(s_1,\ _0))\ldots),$$

which can be expressed in terms of a new function f_σ' as

$$\sigma_k = f_\sigma'(s_k, s_{k-1}, s_{k-2}, \ldots s_1, \sigma_0).$$

Now, defining $\sigma_0 \equiv s_0$, which is consistent with (9b) since σ_{-1} is arbitrary, then f'_σ represents a sliding block encoding of the sequence $\{s_i\}$, $i = 0,1,\ldots k$.

Now (9a) can be expressed as:

$$p(s_k\, u_k | u_{k-1}\, \sigma_{k-1}) = p(u_k | u_{k-1}\, \sigma_{k-1}\, s_k) \cdot p(s_k | u_{k-1}\, \sigma_{k-1})$$

and by the corollary hypothesis on the invertibility of f_σ,

$$= p(u_k | u_{k-1}\, \sigma_{k-1}\, f_{\sigma s}^{-1}(\sigma_k, \sigma_{k-1})) \cdot p(s_k | u_{k-1}\, \sigma_{k-1}).$$

But u_k is already conditioned on σ_{k-1}, so the additional conditioning provided by $s_k = f_{\sigma s}^{-1}(\sigma_k, \sigma_{k-1})$ is exactly that provided by σ_k, thus (9a) is reduced to:

$$p(s_k\, u_k | u_{k-1}\, \sigma_{k-1}) \equiv p(u_k | u_{k-1}\, \sigma_{k-1}\, \sigma_k) \cdot p(s_k | u_{k-1}\, \sigma_{k-1}),$$

which are the two processes hypothesized, proving the corollary.

Comments: The theorem and corollary are interesting primarily from a theoretical viewpoint. The main theorem actually does no more than place the intuitively developed model for the HKM process on a solid probabilistic foundation. In Section V, where an optimal estimator for the state of the process is derived through Bayesian techniques, the form of the model presented in the main theorem is that which is used. However, after the estimation algorithm has

been derived, it is shown that the optimal estimator has a trellis structure, which is not surprising in view of the corollary result showing an SBC interpretation. The block diagram shown in Figure 9 is useful for visualizing the evolution of the output state, s_k.

B. BASEBAND HKM CHANNEL MODEL

Although the channel model for the HKM process described in Section III was useful for obtaining lower bounds an error-rate performance, it is of little use in actually describing the physical processes which affect the reliable transmission of a Morse message. Consider the following simplified model of the communication channel for Morse transmitted at HF. The keyer turns the transmitter on and off according to the HKM source. When keyed, the transmitted RF signal has amplitude C(t) at a carrier frequency ω. The HF propagation channel introduces both additive noise (N(t)) in the form of atmospherics and interference, and multiplicative noise (B(t)) in the form of fading and multipath propagation effects. At the receiver, the carrier is removed after being band-pass filtered and gain-controlled. After low-pass filtering and sampling, the baseband signal is given by $z_k = x_k c_k b_k + n_k$, where c_k is the sampled, gain-controlled received signal amplitude; b_k is the sampled, gain-controlled, low-pass filtered effective multiplicative noise component; and n_k is the low-pass filtered version of the additive noise.

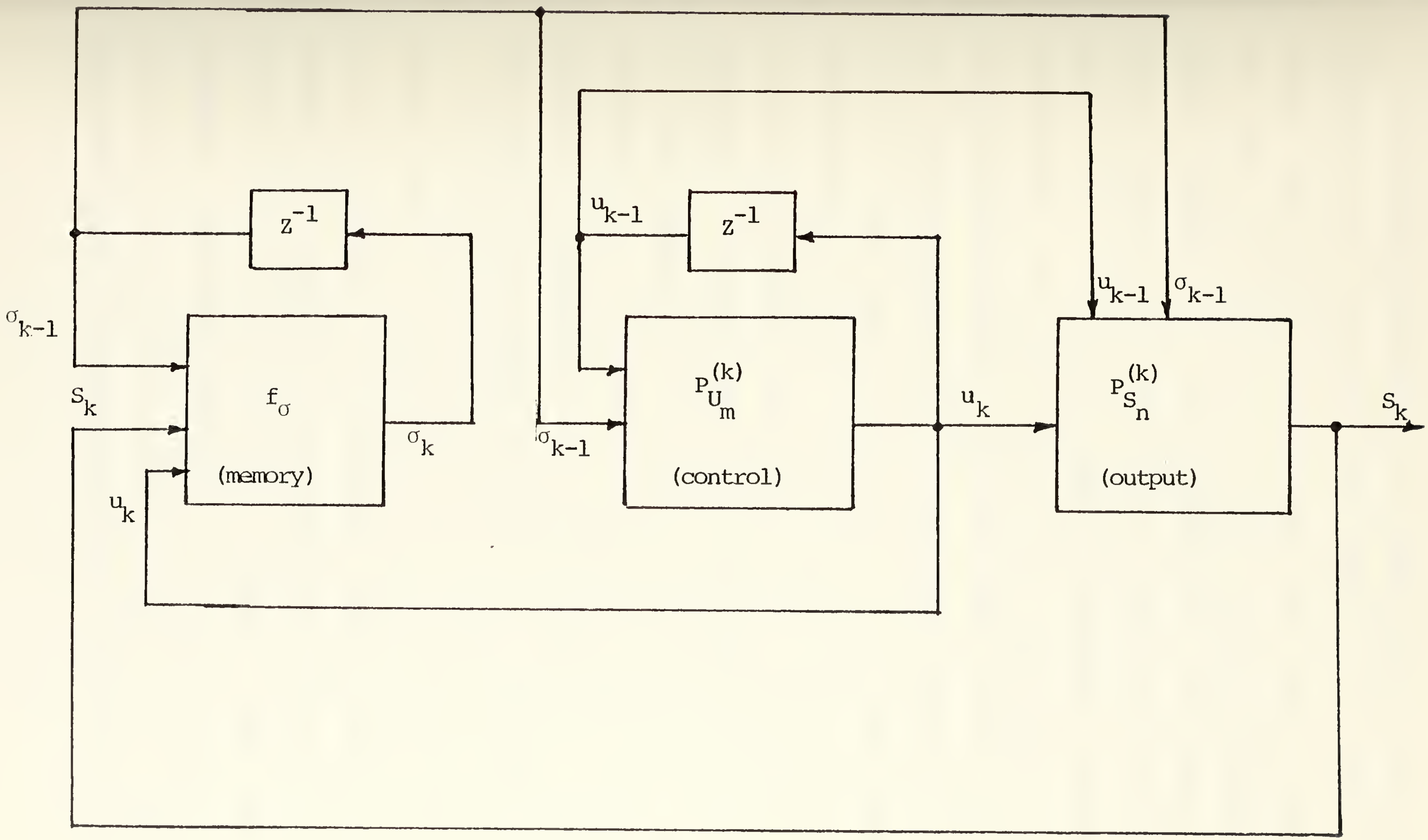

FIGURE 9. Block Diagram of HKM Signal Model

The sampled version of the amplitude of the transmitted carrier c_k is a constant value while $x_k = 1$. During the period when $x_k = 0$, the amplitude will remain constant at the same value as for $x_k = 1$ for a large percentage of the time. However, it is not uncommon for the operator to go into a pause during which time he readjusts the transmitter power either up or down. These adjustments are usually made between messages, but also can occur during a short pause between letters. Thus the signal carrier amplitude is a random variable with a transition probability density which is conditioned on the memory of the HKM process and the current key state. In the simplest case, the model may be made conditional only on x_k and x_{k-1}, having, as a consequence, the result that the carrier amplitude is allowed to change randomly during every 0-state duration. More realistically, one level of complexity greater allows the transition probability to be conditioned on β_{k-1} such that the amplitude can change only when β_{k-1} indicates a pause.

The effect of transmitter power fluctuations at the output of the receiver is dependent on SNR and on the AGC employed for gain-leveling. For moderate to high received SNR, the effective c_k observed at the receiver output stays relatively constant because of AGC action. However, when noise power becomes a significant portion of the total power controlling the AGC, then c_k varies nearly the same as C_k. Thus an efficient model of transmitter power fluctuations must take

into consideration not only the actual power variations of the transmitter, but also the effect of the receiver RF, IF, and AGC sections as well.

Consider now the multiplicative noise term, which has the observable effect of varying signal amplitude. If it arises because of relatively slow fading, then its effect will be cancelled by the combination of AGC and low-pass filtering. If, on the other hand, it is caused by fast fading (perhaps due to multipath), then the AGC cannot respond fast enough to keep the output signal-level constant. On an OOK signal, the effect is the same as if the transmitter power were changed during the carrier off-time.

The term $c_k b_k$, then, represents an effective transmitter power fluctuation, dependent on both the HKM process and the HF channel, with the result that the marks of the HKM process appear to be transmitted with random amplitude. During the period of a MARK, the effective fluctuations are caused by the slow fading component with intensity and rate determined by the channel, the AGC, and the low-pass filter.

In view of the above consideration, it is appropriate to model the apparent transmitted amplitude y_k as a conditional gauss-Markov process, dependent on both the HKM process, and the channel:

(10a) $y(k) = \gamma F(s_k\ \sigma_{k-1})\ y(k-1) + \Gamma(s_k\ \sigma_{k-1})\ w_t(k)$

where $w_t(k)$ is a zero-mean gaussian random sequence with unit variance;

$F(s_k\ \sigma_{k-1})$ is a function of the state of the HKM source;

$\Gamma(s_k\ \sigma_{k-1})$ is a similar function,

γ is a channel-dependent fading parameter.

Now, since the amplitude is observed only during a MARK period, the measurement equation is given by:

(10b) $$z_k = x_k y_k + u_k,$$

where n_k is the low-pass filtered, gain-controlled channel noise.

Equations (10) represent the described HKM Baseband channel model, which accounts for the effects of fading on an OOK signal and the effect of actual transmitter power fluctuations caused by the sending operator.

Generalizing these intuitive concepts to a vector channel results in the following channel-measurement model. Consider that the output sequence s_k of the HKM is observed through the following channel and measurement processes:

$$y_k = \Phi(s_k\ \sigma_{k-1})\ y_{k-1} + \Gamma(s_k\ \sigma_{k-1})\ w_k$$

$$z_k = H(s_k)\ y_k + n_k$$

where

y_k	is a p-dimensional state vector;
z_k	is a q-dimensional measurement vector;
$\Phi(s_k\ \sigma_{k-1})$	is a p x p state transition matrix;
$H(s_k)$	is a q x p measurement matrix;
$\Gamma(s_k\ \sigma_{k-1})$	is a p x p matrix;
w_k	is a p-dimensional plant noise vector;
n_k	is a q-dimensional measurement noise vector;

w_k is statistically independent of w_ℓ for $\ell \neq k$;

n_k is statistically independent of n_ℓ for $\ell \neq k$;

w_k is statistically independent of n_k;

$p(y_0), p(w_k), p(n_k)$ are given probability densities.

It is to be noted that this observation model, when conditioned on s_k, σ_{k-1}, is linear. Further if the probability densities are gaussian, then the $s_k\ \sigma_{k-1}$ - conditional estimate of y_k, given the sequence z_k, $k = 1,2,\ldots,$ is given by the well-known Kalman filter recursions.

V. THE ESTIMATION PROBLEM

The estimation problems of interest, based on the HKM source, channel, and measurement models, can be divided into two broad classes. The first results when the HKM transition and mapping probabilities are known a-priori for all k; the problem then is to find an optimal (in some sense) estimator for s_k and/or u_k given noisy observations. It will be shown that the desired estimator is not physically realizable in general because it requires an exponentially expanding memory. In Section VIII, however, practical realizations of a suboptimal estimator are discussed, and it is shown that one can systematically come as close to optimal estimation as desired. The second class of estimation problems results when the HKM model probabilities are known only to the level of an initial probability distribution. The problem here is to estimate s_k and/or u_k and the transition and mapping probabilities themselves. Only the first class will be treated here.

In this class of estimation problems, the transition and mapping probabilities are specified, and the problem is to estimate the state of the system at time k, given the sequence of all past measurements $z^k \triangleq \{z_1, z_2, \ldots, z_k\}$. The state estimate of the system is given by the joint estimate of the output, control, and memory states $s_k\ u_k\ \sigma_k$. The problem of obtaining an optimal estimate of the state

is approached in the traditional manner; that is, the (posterior) conditional probability distribution $p(s_k\ u_k\ \sigma_k | z^k)$ is determined for all k, and a suitable optimality criterion is applied to this distribution to arrive at an optimal estimator.

Using the Bayesian approach to the problem of obtaining the posterior distribution, a recursive form for the estimator is obtained. It will be shown that the resulting structure can be realized by a set of simpler, identical filters, operating on a tree or trellis. In the case of parameter-conditional linear-gaussian observation and measurement models, these "elemental" filters are Kalman filters. In case the observation and/or measurement models are not linear-gaussian, then the body of knowledge on non-linear filtering can be brought to bear on the design of these elemental filters.

A. ESTIMATOR DERIVATION

In the following it will be necessary to keep track of both the time index, k, and the state value indices for the states $s_k \in \{S_i\}$, $u_k \in \{U_j\}$, $\sigma_k \in \{\Lambda_\ell\}$. To reduce the notational burden which would result from the explicit notation of probability statements such as $Pr[s_k = S_i | u_k = U_j, u_{k-1} = U_m, \sigma_{k-1} = \Lambda_n]$, the following abbreviated notation will be used. The subscript k is the time index, and the superscript is the index of the set of state values. When k is used as a superscript, it refers to the time <u>sequence</u> of values, 0,1,2,...,k; e.g.,

$z^k \triangleq z_1 z_2 \ldots z_k$. Additionally the vector notation using an underbar will be dropped, with the understanding that all variables are implicitly vector-valued. In terms of this notation, the HKM signal and observation models are:

(11) Output State Mapping probabilities:

$$p(s_k^i | u_k^j \; u_{k-1}^m \; \sigma_{k-1}^q) \triangleq \Pr[s_k = S_i | u_k = U_j, u_{k-1} = U_m, \sigma_{k-1} = \Lambda_q]$$

(12) Control State Transition probabilities:

$$p(u_k^j | u_{k-1}^m \; \sigma_{k-1}^q) \triangleq \Pr[u_k = U_j | u_{k-1} = U_m, \sigma_{k-1} = \Lambda_q]$$

(13) Memory:

$$\sigma_k^\ell = f_\sigma(s_k^i, \sigma_{k-1}^q) \triangleq f_\sigma(S_i, \Lambda_q)$$

(14) Channel:

$$y_k = \Phi(s_k^i \; \sigma_{k-1}^q) \; y_{k-1} + \Gamma(s_k^i \; \sigma_{k-1}^q) \; w_k$$

(15) Measurement:

$$z_k = H(s_k^i) \; y_k + n_k .$$

The well-known Bayesian procedure (see, for example, Lee [8]) for recursively determining the posterior density

(distribution) is given as follows. At time k-1, the mixture density:

$$p(y_{k-1}\ s^n_{k-1}\ u^m_{k-1}\sigma^q_{k-1}|z^{k-1}) \equiv p(y_{k-1}|s^n_{k-1}\ u^m_{k-1}\ \sigma^q_{k-1};z^{k-1})$$

$$\cdot\ p(s^n_{k-1}\ u^m_{k-1}\ \sigma^q_{k-1}|z^{k-1})$$

has been obtained. The density at time k, after receipt of a new measurement z_k, is given by Bayes' rule:

$$(16)\quad p(y_k\ s^i_k\ u^j_k\ \sigma^\ell_k|z^k) = \frac{p(z_k|y_k\ s^i_k\ u^j_k\sigma^\ell_k\ z^{k-1})p(y_k\ s^i_k\ u^j_k\ \sigma^\ell_k|z^{k-1})}{p(z_k|z^{k-1})}$$

where:

$$(17)\quad p(y_k\ s^i_k\ u^j_k\ \sigma^\ell_k|z^{k-1}) =$$

$$\sum_{nmq}\int_{y_{k-1}} p(y_k\ s^i_k\ u^j_k\ \sigma^\ell_k|y_{k-1}\ s^n_{k-1}\ u^m_{k-1}\ \sigma^q_{k-1};z^{k-1})$$

$$\cdot\ p(y_{k-1}\ s^n_{k-1}\ u^m_{k-1}\ \sigma^q_{k-1}|z^{k-1})\ dy_{k-1}$$

$$(18)\quad p(z_k|z^{k-1}) =$$

$$\sum_{ij}\int_{y_k} p(y_k\ s^i_k\ u^j_k\ \sigma^\ell_k|z^{k-1})p(z_k|y_k\ s^i_k\ u^j_k\ \sigma^\ell_k;z^{k-1})\ dy_k$$

The desired state posterior probability distribution then is obtained from (16) by integrating over y_k:

$$(19) \quad p(s_k^i\ u_k^j\ \sigma_k^\ell | z^k) = \int_{y_k} p(y_k\ s_k^i\ u_k^j\ \sigma_k^\ell | z^k)\ dy_k.$$

Substituting expression (18) for $p(z_k | z^{k-1})$ into (16), expression (19) becomes:

$$(20) \quad p(s_k^i\ u_k^j\ \sigma_k^\ell | z^k) = \frac{\int_{y_k} p(z_k | y_k\ s_k^i\ u_k^j\ \sigma_k^\ell\ z^{k-1})\ p(y_k\ s_k^i\ u_k^j\ \sigma_k^\ell | z^{k-1})\ dy_k}{\sum_{ij} \int_{y_k} p(y_k\ s_k^i\ u_k^j\ \sigma_k^\ell | z^{k-1})\ p(z_k | y_k\ s_k^i\ u_k^j\ \sigma_k^\ell\ z^{k-1})\ dy_k}$$

and the problem is to obtain a result for the integral over y_k in terms of the prior density at time k-1, and the model transition probabilities.

The first term in the integrand, $p(z_k | y_k\ s_k^i\ u_k^j\ \sigma_k^\ell\ z^{k-1})$, is readily determined from the measurement equation (15) and the density of the noise, $p_n(n_k)$. In the case of n_k a white sequence, the density is given simply by:

$$(21) \quad p(z_k | y_k\ s_k^i\ u_k^j\ \sigma_k^\ell\ z^{k-1}) \equiv p(z_k | y_k\ s_k^i) = p_n(z_k - H(s_k^i) y_k).$$

The second term in the integrand is given by (17) in terms of the prior density and the transition probabilities. Rewriting the mixture densities in (17) in terms of the component conditional density for y_k and the discrete distributions for $s_k\ u_k\ \sigma_k$, expression (17) becomes:

$$(22)\quad p(y_k\ s_k^i\ u_k^j\ \sigma_k^\ell | z^{k-1}) =$$

$$\sum_{nmq} \int_{y_{k-1}} \{p(y_k | y_{k-1}\ s_k^i\ u_k^j\ \sigma_k^\ell\ s_{k-1}^n\ u_{k-1}^m\ \sigma_{k-1}^q ; z^{k-1}) \qquad (a)$$

$$\cdot\ p(s_k^i\ u_k^j\ \sigma_k^\ell | y_{k-1}\ s_{k-1}^n\ u_{k-1}^m\ \sigma_{k-1}^q ; z^{k-1}) \qquad (b)$$

$$\cdot\ p(y_{k-1} | s_{k-1}^n\ u_{k-1}^m\ \sigma_{k-1}^q ; z^{k-1}) \qquad (c)$$

$$\cdot\ p(s_{k-1}^n\ u_{k-1}^m\ \sigma_{k-1}^q | z^{k-1})\}\ dy_{k-1} \qquad (d)$$

Now since $s_k\ u_k\ \sigma_k$ are independent of y_{k-1}, the density on line (c) above is not changed by writing:

$$(e)\quad p(y_{k-1} | s_{k-1}^n\ u_{k-1}^m\ \sigma_{k-1}^q ; z^{k-1}) \equiv p(y_{k-1} | s_k^i\ u_k^j\ \sigma_k^\ell\ s_{k-1}^n\ u_{k-1}^m\ \sigma_{k-1}^q ; z^{k-1}).$$

Also, by virtue of this independence, the expression on line (b) becomes:

$$(f)\quad p(s_k^i\ u_k^j\ \sigma_k^\ell | y_{k-1}\ s_{k-1}^n\ u_{k-1}^m\ \sigma_{k-1}^q ; z^{k-1}) \equiv p(s_k^i\ u_k^j\ \sigma_k^\ell | s_{k-1}^n\ u_{k-1}^m\ \sigma_{k-1}^q).$$

Combining (a) & (e), substituting (f) for (b), and rearranging the terms of (22), the expression becomes:

$$p(y_k\ s_k^i\ u_k^j\ \sigma_k^\ell | z^{k-1}) =$$

$$\sum_{nmq} p(s_k^i\ u_k^j\ \sigma_k^\ell | s_{k-1}^n\ u_{k-1}^m\ \sigma_{k-1}^q)\ p(s_{k-1}^n\ u_{k-1}^m\ \sigma_{k-1}^q | z^{k-1})$$

$$\cdot \int_{y_{k-1}} p(y_k\ y_{k-1} | s_k^i\ u_k^j\ \sigma_k^\ell\ s_{k-1}^n\ u_{k-1}^m\ \sigma_{k-1}^q ; z^{k-1})\ dy_{k-1}.$$

Carrying out the integration over y_{k-1}, and noting that y_k is not dependent on $u_k\ \sigma_k\ s_{k-1}\ u_{k-1}$, the desired result for expression (17), in terms of the prior and transition probabilities, is given by:

$$(23)\quad p(y_k\ s_k^i\ u_k^j\ \sigma_k^\ell | z^{k-1}) =$$

$$\sum_{n,m,q} p(s_k^i\ u_k^j\ \sigma_k^\ell | s_{k-1}^n\ u_{k-1}^m\ \sigma_{k-1}^q)\ p(s_{k-1}^n\ u_{k-1}^m\ \sigma_{k-1}^q | z^{k-1})$$

$$\cdot\ p(y_k | s_k^i\ \sigma_{k-1}^q ; z^{k-1}).$$

The integral in (20) is then given in terms of (23) and (21) as:

$$(24)\quad \int_{y_k} p(z_k | y_k\ s_k^i\ u_k^j\ \sigma_k^\ell\ z^{k-1})\ p(y_k\ s_k^i\ u_k^j\ \sigma_k^\ell | z^{k-1})\ dy_k =$$

$$\sum_{nmq} p(s_k^i\ u_k^j\ \sigma_k^\ell | s_{k-1}^n\ u_{k-1}^m\ \sigma_{k-1}^q)\ p(s_{k-1}^n\ u_{k-1}^m\ \sigma_{k-1}^q | z^{k-1})$$

$$\cdot \int_{y_k} p(z_k | y_k\ s_k^i)\ p(y_k | s_k^i\ \sigma_{k-1}^q ; z^{k-1})\ dy_k.$$

The resulting integral over y_k in the above expression is seen to be a likelihood function since

$$\int_{y_k} p(z_k|y_k\ s_k^i)\ p(y_k|s_k^i\ \sigma_{k-1}^q;z^{k-1}) = p(z_k|s_k^i\ \sigma_{k-1}^q;z^{k-1}).$$

Denoting this integral, then, as the likelihood,

$$(25)\quad L_k^{iq} \triangleq \int_{y_k} p(z_k|y_k\ s_k^i)\ p(y_k|s_k^i\ \sigma_{k-1}^q;z^{k-1})\ dy_k,$$

the posterior conditional density (20) is given by (24) & (25) as

$$(26)\quad p(s_k^i\ u_k^j\ \sigma_k^\ell|z^k) = \frac{\sum\limits_{nmq} p(s_k^i\ u_k^j\ \sigma_k^\ell\ |\ s_{k-1}^n\ u_{k-1}^m\ \sigma_{k-1}^q)p(s_{k-1}^n\ u_{k-1}^m\ \sigma_{k-1}^q|z^{k-1})L_k^{iq}}{\sum\limits_{ij}\ \sum\limits_{nmq} p(s_k^i u_k^j \sigma_k^\ell|s_{k-1}^n\ u_{k-1}^m \sigma_{k-1}^q)p(s_{k-1}^n u_{k-1}^m \sigma_{k-1}^q|z^{k-1})L_k^{iq}}$$

This is the desired result for the recursive calculation of the probabilities of the states $s_k\ u_k\ \sigma_k$ given the measurement sequence z^k. In terms of the model transition probabilities (11) and (12) and the memory function (13), the transition probabilities are computed as:

$$p(s_k^i\ u_k^j\ \sigma_k^\ell|s_{k-1}^n\ u_{k-1}^m\ \sigma_{k-1}^q) \equiv$$

$$p(s_k^i|u_k^j\ u_{k-1}^m\ \sigma_{k-1}^q)\ p(u_k^j|u_{k-1}^m\ \sigma_{k-1}^q)$$

along the allowable transition paths specified by

$$\sigma_k^\ell = f_\sigma(s_k^i \; \sigma_{k-1}^q).$$

For each memory state and control state value at time k-1, the transition probability $p(u_k^j | u_{k-1}^m \; \sigma_{k-1}^q)$ is specified by (12) for all j,m,q. Then for each j,m,q, the mapping probability $p(s_k^i | u_k^j \; u_{k-1}^m \; \sigma_{k-1}^q)$ is given for all i by (11); the value for σ_k is found for each i,q pair by (13), and L_k^{iq} is computed by (25). The posterior probabilities are then computed by (26) and the state values and their probabilities are in place for the next recursion.

Clearly the ability to carry out the recursion (26) exactly depends on whether or not the likelihood (25) can be found in closed form. Such a form can indeed be found for the linear channel and measurement models (14) and (15) in case the noise n_k is white and gaussian, as will now be shown.

First note that the densities involved in the expression for the likelihood (25) are both conditioned on specific realizations of s_k and σ_{k-1}, namely $s_k = S_i$ and $\sigma_{k-1} = \Lambda_q$. The first density $p(z_k | y_k \; s_k^i)$ is given by (21) for the white noise sequence; for the white gaussian sequence, (21) becomes:

$$(27) \quad p(z_k | y_k \; s_k^i) = p_n(z_k - H(s_k^i) y(k)) = N_{z_k}(H(s_k^i) y(k), R),$$

where $N_x(m,V)$ is the gaussian density with mean $x = m$, variance V and $p_n(n_k) = N_{n_k}(0,R)$.

Consider now the second density in the integrand (25), $p(y_k|s_k^i\ \sigma_{k-1}^q;z^{k-1})$, the $s_k\ \sigma_{k-1}$ - conditional one-step prediction density for y_k, <u>along the path specified by the S_i transition at time k from the memory state Λ_q at time k-1</u>. The path label, then, at time k, resulting from the extension of the path labeled Λ_q at time k-1, is $\Lambda_\ell = f_\sigma(S_i,\Lambda_q)$. Now

$$p(y_k|s_k^i\ \sigma_{k-1}^q;z^{k-1}) = \int_{y_{k-1}} p(y_k|y_{k-1}\ s_k^i\ \sigma_{k-1}^q;z^{k-1}) \cdot p(y_{k-1}|s_k^i\ \sigma_{k-1}^q;z^{k-1})\ dy_{k-1},$$

and since the $s_k^i\ \sigma_{k-1}^q$ pair is uniquely embodied in $\sigma_k^\ell = f_\sigma(s_k^i\ \sigma_{k-1}^q)$, and y_{k-1} given z^{k-1} is independent of s_k, the above expression becomes

$$(28)\quad p(y_k|\sigma_k^\ell;z^{k-1}) = \int_{y_{k-1}} p(y_k|y_{k-1}\ s_k^i\ \sigma_{k-1}^q;z^{k-1}) \cdot p(y_{k-1}|\sigma_{k-1}^q;z^{k-1})\ dy_{k-1}$$

for each σ_k^ℓ along a path given by

$$\sigma_k^\ell = f_\sigma(s_k^i,\sigma_{k-1}^q).$$

Now when the σ-conditional density for the initial value of y_k is gaussian and the $s_k\ \sigma_{k-1}$ - conditional

channel model is linear gaussian, the above density (28) is gaussian for all k, and the mean and variance of the density is given by the Kalman filter recursions.

Specifically, this density is given by

$$(29) \quad p(y_k | \sigma_k^\ell\ z^{k-1}) = N_{y_k}(\hat{y}_{k|k-1}(\Lambda_\ell), V_{k|k-1}(\Lambda_\ell))$$

where

$$\hat{y}_{k|k-1}(\Lambda_\ell) = \Phi(S_i\ \Lambda_q)\ \hat{y}_{k-1|k-1}(\Lambda_q)$$

$$V_{k|k-1}(\Lambda_\ell) = \Phi(S_i\ \Lambda_q)\ V_{k-1|k-1}(\Lambda_q)\ \Phi^T(S_i\ \Lambda_q) + Q_k(S_i\ \Lambda_q)$$

$$\Lambda_\ell = f_\sigma(S_i\ \Lambda_q)$$

and the recursions for $\hat{y}_{k|k}(\cdot)$ and $V_{k|k}(\cdot)$ are given by the remaining Kalman filter equations:

$$\hat{y}_{k|k}(\Lambda_\ell) = \hat{y}_{k|k-1}(\Lambda_\ell) + G_k(\Lambda_\ell)[z_k - H(S_i)\hat{y}_{k|k-1}(\Lambda_\ell)]$$

$$V_{k|k}(\Lambda_\ell) = (I - G_k(\Lambda_\ell)H(S_i))\ V_{k|k-1}(\Lambda_\ell)$$

$$G_k(\Lambda_\ell) = V_{k|k-1}(\Lambda_\ell)H^T(S_i)[H(S_i)V_{k|k-1}(\Lambda_\ell)H^T(S_i) + R_k]^{-1}.$$

Substituting these expressions (27) and (29) back into (25), the integral to evaluate becomes:

$$L_{iq}^{k} = \int_{y_k} N_{z_k}(H(S_i)y_k, R_k) \cdot N_{y_k}(\hat{y}_{k|k-1}(\Lambda_\ell), V_{k|k-1}(\Lambda_\ell))dy_k.$$

The evaluation of this integral is a basic exercise in integration of gaussian densities and is given by [8]:

$$(29) \quad L_{iq}^{k} = c\left|V_{z_{k|k-1}}(\Lambda_\ell)\right|^{1/2} \text{Exp}\{-\tfrac{1}{2}[\tilde{z}_{k|k-1}(\Lambda_\ell)]^T[V^{-1}_{z_{k|k-1}}(\Lambda_\ell)] \cdot [\tilde{z}_{k|k-1}(\Lambda_\ell)]\}$$

where

$$\tilde{z}_{k|k-1}(\Lambda_\ell) = z_k - H(S_i)\ \hat{y}_{k|k-1}(\Lambda_\ell)$$

$$V_{z_{k|k-1}}(\Lambda_\ell) = H(S_i)\ V_{k|k-1}(\Lambda_\ell)H^T(S_i) + R_k.$$

B. IMPLEMENTATION STRUCTURE OF ESTIMATOR

The structure of the filter realization density (26), together with the likelihood calculation (29), is that of a tree with nodes given by the past state trajectories and with branches labeled by the states of process. For each transition, i.e., each path extension to a new node, the likelihood of the transition is computed from the Kalman filter recursions along that particular path. The likelihoods are multiplied by the transition probability for that path extension, and by the previous path probability. The

updated path probabilities are then obtained by normalizing these products. The tree structure showing the evolution of the path labels according to a particular function is illustrated in Figure 10.

The next stage of this structure would obviously contain $N \times I_k$ nodes where N is the number of possible states S_i and I_k is the number of nodes at stage k. Thus the number of nodes expands exponentially. However, in case the function f_σ depends only on a finite portion of the past trajectory, then the tree structure eventually becomes a finite trellis at the stage which accounts for the definition of f_σ, resulting in a trellis appropriate for Viterbi decoding. If the function f_σ has infinite memory, then obviously some approximation technique must be used to keep the number of nodes finite. One such possible approximation is to save only a given number of nodes at each stage, most likely those with the highest posterior probability. Another scheme which is possible is to save only enough nodes at each stage, the sum of whose posterior probabilities is less than or equal to some specified number, P_{opt}. This latter method is attractive from the standpoint that for high signal-to-noise ratios the number of nodes saved would be small, while for low SNR, the number saved would be larger. This scheme therefore would have the attractive feature that the processing load would automatically adapt to the SNR.

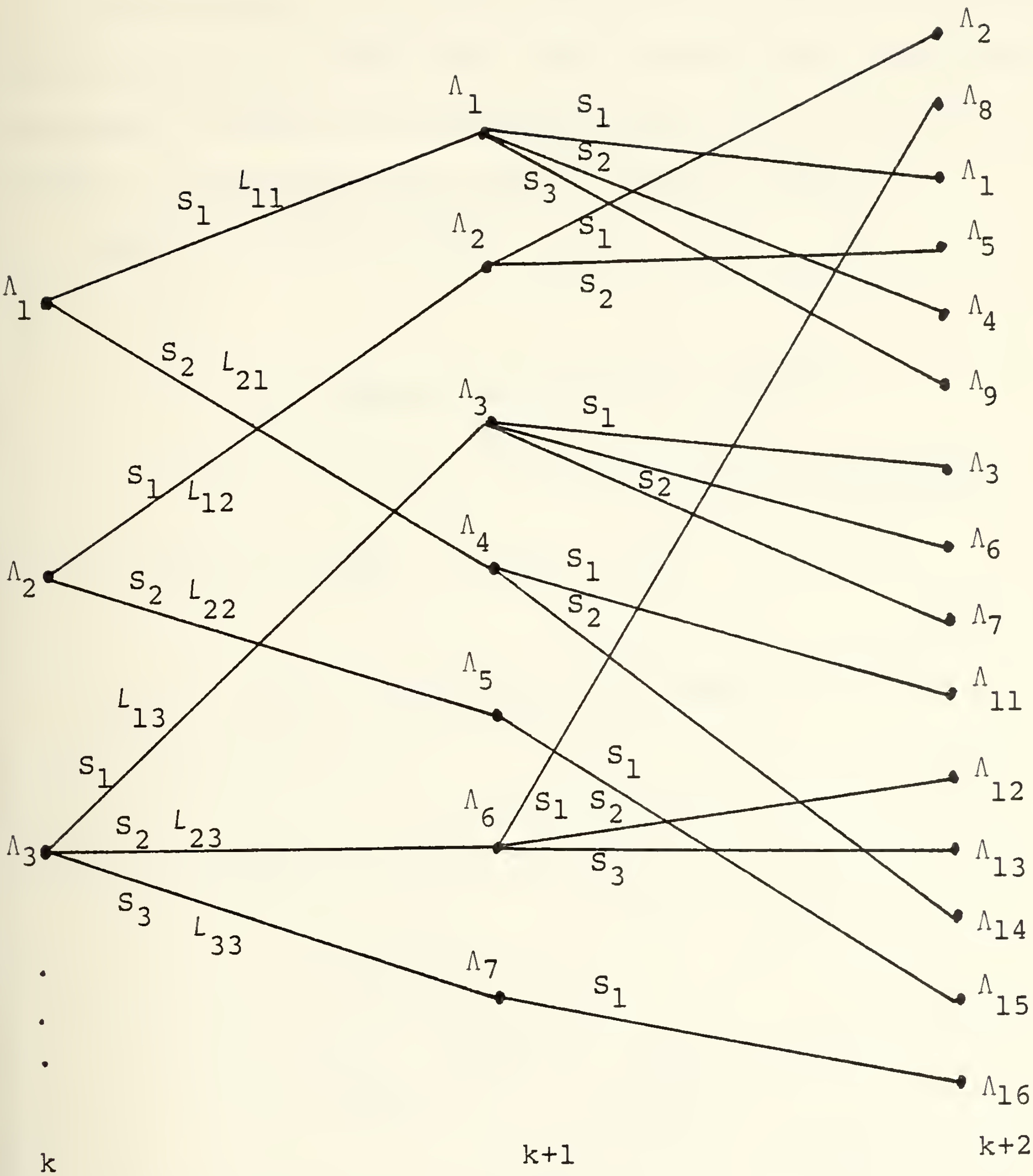

FIGURE 10. Estimator Structure

C. ESTIMATOR ALGORITHM

The following algorithm implements the estimator given by equations (26) and (29). For a practically realizable estimator, some rule which saves only a finite number of paths as discussed above must be used at step 8.

Step 0 Initialization:

$k = 0$

$I^{o} = MN$ (number of joint S_k, u_k states)

$\Lambda^{o}(i)$, $i = 1,2,\ldots,I^{o}$, arbitrarily specified

$P^{o}(i) = 1/MN$, $i = 1,2,\ldots,I^{o}$

Step 1 Obtain indices for new nodes:

a) $k = k + 1$

b) For $q = 1,2,\ldots I^{(k-1)}$

$m = 1,2,\ldots M$

$n = 1,2,\ldots N$

$$j = (q-1)\, I^{(k-1)} + (m-1)M + n$$

Step 2 Label each new node:

For each n, m, q, obtain

$$\Lambda^{k}(j) = f_{\sigma}(S_m, \Lambda^{k-1}(q))$$

Step 3 Obtain transition probabilities:

For each n, m, q, obtain

$$PTR(m, n, q) = PS(S_m | U_n, U_q, \Lambda_q^{k-1}) \cdot PR(U_k | U_q, \Lambda_q^{k-1}).$$

Step 4 Calculate L_{mq}^k for each hypothesized transition (some obvious indices are omitted):

For each n, m, q, compute:

a) Kalman step:

$$\hat{y}_{k|k-1}(j) = \underset{\sim}{\phi}(S_m \; \Lambda^{k-1}(q)) \; \hat{y}_{k-1|k-1}(q)$$

$$V_{k|k-1}(j) = \underset{\sim}{\phi}(S_m \; \Lambda^{k-1}(q)) V_{k-1|k-1}(q) \underset{\sim}{\phi}^T + Q_k(S_m \; \Lambda^{k-1}(q))$$

$$G_k(j) = V_{k|k-1}(j) H^T(S_m) [H V_{k|k-1} H^T + R_k]^{-1}$$

$$\tilde{z}_{k|k-1}(j) = z_k - H(S_m) \; \hat{y}_{k|k-1}(j)$$

$$\hat{y}_{k|k}(j) = \hat{y}_{k|k-1}(j) + G_k(j) \tilde{z}_{k|k-1}(j)$$

$$V_{k|k}(j) = (I - G_k(j) H(S_m)) V_{k|k-1}(j)$$

$$V_{z_{k|k-1}}(j) = H(S_m) V_{k|k-1}(j) H^T + R_k.$$

Step 8 Update number of paths

$$I^{(k)} = NMI^{(k-1)}$$

go to step 1.

It is to be noted that the computations cannot be carried out "in place"; that is, $\Lambda^k(j)$ cannot be stored in the same locations as $\Lambda^{k-1}(j)$ until all the $\Lambda^k(j)$ have been computed. Similarly, the Kalman filter means and variances must be stored in separate temporary locations until step 5 is completed.

D. DISCUSSION AND RELATION TO PREVIOUS RESULTS

In the language of the literature on non-linear filtering, the present result represents an extension of previous results in system identification problems to the case where the unknown discrete system parameter s_k is the result of a probabilistic mapping of an underlying memory-conditional Markov process. Previous investigations have treated both the case where s_k is a Markov process [10], [11], and the case for s_k an unknown time-invariant parameter [9]. The present result reduces to these results for the appropriate modeling of s_k.

Case I: Markovian Parameters [10] [11]

In this case, S_k is a finite-state discrete-time Markov chain with transition matrix $\{P_{ij}(k)\} \triangleq \{Pr[s_k = S_i | s_{k-1} = S_j]\}$. The n-dimensional, S-conditional system dynamics are given by:

$$y_k = \Phi(S_k)y_{k-1} + \Gamma(S_k)w_{k-1}$$

and the m-dimensional measurements are

$$z_k = H(S_k)y_k + n_k$$

The random variables w_k, n_k are zero-mean independent gaussian, and independent of the Markov chain S_k.

In terms of the generalized model developed above, the memory function f_σ (13) is specified, for this case, by $\sigma_k = [s_k\ s_{k-1}\ \ldots\ s_o]^T$ and the output state mapping probabilities (11) are independent of the U_k - process and given by $\{p_{ij}(k)\}$. The system dynamics and measurement equations, in terms of the realization of the S_k - process are then given by

$$y_k = \Phi(s_k\ \sigma_{k-1})y_{k-1} + \Gamma(s_k\ \sigma_{k-1})w_k$$

$$z_k = H(s_k\ \sigma_{k-1})y_k + n_k$$

The posterior measurement-conditional path probabilities are given exactly by equation (26). The likelihood equations (29) for L^h_{iq} are obtained in the same manner by replacing $H(S_i)$ with $H(S_i\ \Lambda_q)$ where Λ_q is a path specification obtained through the memory function: $\Lambda_q = [s_i^{(k-1)}\ s_j^{(k-2)}\ \ldots\ s_\ell^{(o)}]$. The posterior probability for the parameter s_k, then is given by summing over the paths:

$$P^k(S_i) \triangleq \Pr[s_k = S_i] = \sum_{q=1}^{M} P_{iq}^k$$

where

$$P_{iq}^k \triangleq \Pr[s_k = S_i; \sigma_k = \Lambda_q | z^k].$$

The CME or MAP estimate may then be obtained:

$$\text{CME:} \quad \hat{s}_k = \sum_{i=1}^{N} s_i \; P^k(S_i)$$

$$\text{MAP:} \quad \hat{s}_k = S_j: \quad P^k(S_j) = \max_i P^k(S_i).$$

Case II: Unknown Time-invariant Parameters [9]

For this case, since the parameter s_k does not change, the memory function is given by $\sigma_k = s_o$, with an initial probability given by $p_i^o = \Pr[s_o = S_i]$, $i = 1,2, \ldots N$.

The dynamics and measurement equations are

$$y_k = \Phi(\sigma_k) \; y_{k-1} + \Gamma(\sigma_k) \; w_{k-1}$$

$$z_k = H(\sigma_k) \; y_k + n_k.$$

Again the posterior path probabilities for s_o are given by equation (26). The likelihoods are determined from equation (29), but since there is no path branching, the Kalman filters all operate in parallel, each on a different conditioning S_i.

Additionally, since the parameter transition probabilities ($k \geq 1$) are given by $Pr[s_k = S_i | s_{k-1} = S_j] = \delta_k(i-j)$, the sum over the previous paths, nmq, in equation (26) becomes a single term for each path extension, and (26) reduces to

$$P^k(S_i) = \frac{P^{(k-1)}(S_i) L_i^k}{\sum_{j=1}^{N} P^{k-1}(S_j) L_j^k} \quad ; \quad i = 1,2 \ldots N$$

which is Lainiotis' result [9]. Note that since there is no branching of the paths, the exact optimum solution for this case is realizable.

VI. A PRACTICAL HKM MODEL

While the results of the preceding theoretical development show how optimum estimation of the state of the HKM process may be performed, it remains, of course, to specify the parameters of the model. In this section, specific values for the model parameters are derived and it is shown in principle how increasingly complex models may be obtained. While the specific model derived in this section is one which considers the letters of the text to be independent and equally likely, it is shown in principle how this model may be easily extended to include contextual message information as well.

The parameters to be determined are given by equations (9):

$$p(s_k\ u_k | u_{k-1}\ \sigma_{k-1}) \qquad \text{and} \qquad f_\sigma(s_k\ \sigma_{k-1}),$$

that is, the state probability transition matrix and the recursive memory function. These expressions are given in terms of the components of s_k, u_k, σ_k by equations 9a and 9b:

Keystate transition matrix: $p(x_k | a_k\ u_k\ \beta_{k-1}\ \alpha_{k-1})$

Morse symbol transition matrix: $p(a_k | \ell_k \; u_k \; \alpha_{k-1} \; \lambda_{k-1} \; \beta_{k-1})$

Text Letter transition matrix: $p(\ell_k | \lambda_{k-1} \; \alpha_{k-1})$

Control transition matrix: $p(u_k | u_{k-1} \; \alpha_{k-1} \; \beta_{k-1} \; \lambda_{k-1})$

Keystate memory function: $f_\beta(x_k, \beta_{k-1})$

Morse Encoder memory function: $f_\alpha(a_k, \alpha_{k-1})$

TEXT memory function: $f_\lambda(\ell_k, \lambda_{k-1})$

Thus the problem is to determine reasonable values for the probability assignments (9a) and to construct the recursive functions (9b) which account for the portion of the process which can be described deterministically.

A. KEYSTATE MODEL

The simplest usable model of the evolution of the keystate would be the simple Markov model described by:

$$P(x_k | x_{k-1}) \triangleq Pr[x_k = j \,|\, x_{k-1} = i]; \quad i,j = 0,1$$

This model suppresses any dependence of the transition probability on current and past Morse symbols (a_k, α_{k-1}) and speed of transmission (u_k), and limits the dependence on past history of the keystate to the immediate past, x_{k-1}. Such a model would have the memory function:

$$\beta_k = f_\sigma(x_k, \beta_{k-1}) \equiv x_k$$

The four Markov transition probabilities $Pr[x_k=1|x_{k-1}=1]$, $Pr[x_k=1|x_{k-1}=0]$, $Pr[x_k=0|x_{k-1}=0]$, $Pr[x_k=0|x_{k-1}=1]$ can be obtained empirically by determining the relative frequency of the states 11, 10, 00, 01 in a large ensemble of actual hand-keyed Morse messages. Clearly these probabilities are dependent on the sampling rate. As a simple example, consider the possible realization of an HKM sequence as illustrated in Figure 11, with the resulting transition probabilities for this sequence given in Table VIII.

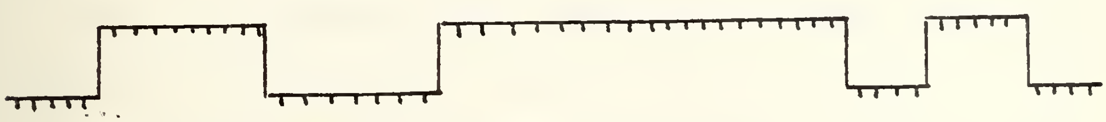

Figure 11. Example Of Sampled HKM Process

TABLE VIII

Transition Probabilities For Illustrative HKM Process

State Transition	No. of Occurrences	Relative Frequency	Probability Estimate
1/1	30	30/33	.91
1/0	3	3/33	.09
0/0	16	16/19	.84
0/1	3	3/19	.16

If the sample rate were different from that illustrated then obviously the relative frequency of each of the transitions would be different; this dependence on sample rate is shown in Table IX.

TABLE IX

Transition Probability As Function Of Sample Rate

Sample Rate (relative to illustration)	State Transitions 1/1 Freq	1/1 Prob	1/0 Freq	1/0 Prob	0/0 Freq	0/0 Prob	0/1 Freq	0/1 Prob
1X	30/33	.91	3/33	.09	16/19	.84	3/19	.16
.5X	13/16	.81	3/16	.19	7/10	.7	3/10	.3
2X	63/66	.95	3/66	.05	35/38	.92	3/38	.08

This artificially induced dependence of the keystate transition probability on sample rate is undesirable from a modeling viewpoint since, in reality, the continuous-time HKM process generated by the sending operator has no such dependence, and it is intuitively unsatisfactory to require the statistics of the sending operator to fit an arbitrarily selected time scale.

This dependence can be removed by normalizing the time-scale to the element-duration, whereby instead of measuring the sample rate in samples per second, the sample rate is measured in samples per duration in elements. Consider,

then, the following expressions for describing the keystate evolution:

$$p(x_k | u_k\ \beta_{k-1}) \triangleq Pr[x_k = j | u_k = U_i, \beta_{k-1} = B_n]$$

$$\beta_{k-1} = \begin{bmatrix} \phi_k \\ x_k \end{bmatrix}$$

$$\phi_k = \phi_{k-1}(1 - x_k - x_{k-1} + 2x_k\ x_{k-1}) + 1$$

where it is seen that the recursion for ϕ_k counts the number of samples since the last zero-one or one-zero keystate transition. This description then conditions the keystate transition probabilities not only on the immediate past keystate x_{k-1}, but also on the data rate u_k, and the number of samples, ϕ_k, that the key has been in a 1 or 0 state since the last transition.

Now if ϕ_k is given in samples with a sampling interval τ, then $T_k \triangleq \phi_k \tau$ is the amount of time (in seconds) since the last 0 to 1 or 1 to 0 transition. If u_k is given in terms of words-per-minute, then the element duration for this rate is $r_k \triangleq (6/5) \times (1/u_k)$. Thus the normalized time for this data rate is given by:

$$T_k' \triangleq T_k / r_k = \frac{5\phi_k\ u_k\ \tau}{6} .$$

This description of the keystate transition probabilities is clearly more satisfying since it depends only on the individual sending operator's rate of transmission and keying characteristics, and not on the sample rate.

The model is still not complete, however, since it does not allow for dependence on the type of Morse symbol being keyed, clearly for dots and element spaces, transitions between mark and space states occur more frequently than for dashes, character spaces, word spaces, and pauses. Additionally, these transition probabilities depend to some extent on the previously keyed symbols, with the degree of dependence being a function of the type of key used. For mechanical bugs, a series of dots separated by element spaces is sent by simply holding the paddle in one position, creating a string of symbols with virtually equal durations. When sending a dot/dash combination, however, the element space duration is determined by the operator's dexterity and not by a mechanical device, so the variability of this element space duration is higher than that for the repeated dot sequence. A similar effect occurs when the key is an electronic bug, although the variability of repeated symbols is even less than that for the mechanical bug. The same type of dependence on past symbols has been noted even for senders using a telegraph key [12] [13]. It has been found that the primary effect is that of reduced variability of element-space durations when the preceeding symbol was a

dot (a detailed analysis of the effect of key type on keystate statistics may be found in [13]).

While the keystate transition probabilities have been noted to be dependent on the preceeding symbol sequence, this dependence is clearly a second-order effect when conditioned on the current symbol. In the model developed here, then, these second-order effects are ignored and the final expressions for the keystate transition probability model are given by:

$$p(x_k | a_k \; u_k \; \beta_{k-1}) = Pr[x_k=j \,|\, a_k=A_i, u_k=U_m, \beta_{k-1}=B_n]$$

$$\beta_k = \begin{bmatrix} \phi_k \\ x_k \end{bmatrix}$$

$$\phi_k = \phi_{k-1}(1 - x_k - x_{k-1} + 2x_k \; x_{k-1}) + 1.$$

In terms of the normalized time scaled, the transition probabilities are $Pr[x_k=j \,|\, x_{k-1}=i, a_k=A_n, r_k, T_{k-1}]$. For example, the probability $Pr[x_k=1 \,|\, x_{k-1}=1, a_k=dot, r_k=r_1, T'_{k-1}=t]$ is the probability that at time k, the key will remain in state 1, given that the operator is sending a dot, that his average element duration is r_1, and that they key has been in state 1 for t element durations. Clearly if t is close to zero, then this probability is nearly 1; and similarly if t > 2, then the probability is small.

An equivalent expression of this probability is the probability that the duration T'_{k-1} becomes duration

$T'_k = T'_{k-1} + \tau/r_k$ since if $x_k = 1$, then $\tau\phi_k = \tau\ \phi_{k-1} + \tau = T_{k-1} + \tau$. This probability can be determined from the density of symbol durations, conditioned on speed r_k and symbol type.

The modeling of the symbol duration densities has been a topic of considerable interest among investigators working on the Morse decoding problem. In the past, because of lack of sufficient empirical data, these densities have been assumed to be truncated gaussian or uniform [2][14]. A recent intensive modeling investigation by Technology Services Corporation [13], did indeed demonstrate the not surprising result that when normalized for speed variation, the density of each symbol duration, averaged over several operators, approaches the gaussian density. For individual operators, however, the densities are far from gaussian, and no single normalizing technique was found which would allow for parametric estimation of the individual densities. Thus, the problem of parameterizing the symbol duration densities of individual Morse operators remains open. Indeed, the evidence supported by the data accumulated so far indicates that estimation of these highly individualistic densities must be accomplished on-line using a combination of parametric and non-parametric techniques.

It is not the purpose of the present research to delve, yet again, into this density estimation problem, but to show, whatever, the proper density, how it can be used most effectively for Morse transcription. For the purposes of the HKM

model developed here, then, a parametric symbol duration density is hypothesized and justified on the basis of intuitive arguments. Traditionally, the local speed of the Morse signal in wpm is defined as 1.2 times the reciprocal of the element duration (in sec), averaged over 10-20 mark-space pairs. A histogram of the normalized symbol duration (actual duration in seconds divided by average element duration) is then taken to be an estimate of the shape of the density function for that symbol. The new approach to be considered here is to hypothesize an <u>instantaneous</u> speed of transmission, defined to be the speed at which a single symbol is sent. The <u>instantaneous</u> element duration (baud) is likewise defined on an individual symbol basis. The effect produced by assigning appropriate probability densities to each results in the same description for an average 10-20 mark-space pair segment as does the traditional approach. The reason for hypothesizing such parameters is simply because it is more intuitively satisfying to propose the existence of individual symbol statistics whose average behavior duplicates the observed empirical behavior, rather than to propose that the statistics of each individual symbol are identical to the observed average statistics. Although this distinction is a fine point, it allows greater flexibility in estimating the keystate transition probability with fewer parameters.

Consider then the following hypothesized random variables:

r = instantaneous speed of transmission

Δ = instantantous element duration (baud)

and let dot and element-spaces have duration = Δ; dashes and character spaces = 3Δ; word-space = 7Δ; pause = 14Δ. Then in terms of the actual symbol duration, d_m:

$$\Delta \triangleq \frac{d_m}{m} ,$$

where m = 1, 3, 7, 14 as appropriate.

The normalized symbol duration, in terms of Δ and r is given by:

$$\phi_\Delta \triangleq \left(\frac{5}{6}\right) \Delta r$$

Note that while Δ is well-defined in terms of a measurable quantity, r is arbitrary. However, it is convenient to define r such that its value is indicative of the actual speed:

$$r_{mean} \triangleq \left(\frac{6}{5}\right) \frac{1}{\Delta}$$

Although this expression determines the statistical behavior of r_{mean} through its dependence on the random variable Δ, clearly it does not restrict the freedom to assign appropriate

statistical description to the other moments of the random variable r, independent of the statistics of Δ.

Consider now the random variable ϕ_Δ, and note that $m\phi_\Delta$ is the normalized symbol duration (in elements), given that the symbol was transmitted at rate r. A density for $m\phi_\Delta$, conditioned on r, then describes the keystate duration random variable, normalized for speed. Let this random variable be described by the Laplacian density (double-sided exponential) with mode m and parameter α, as illustrated in Figure 12, below.

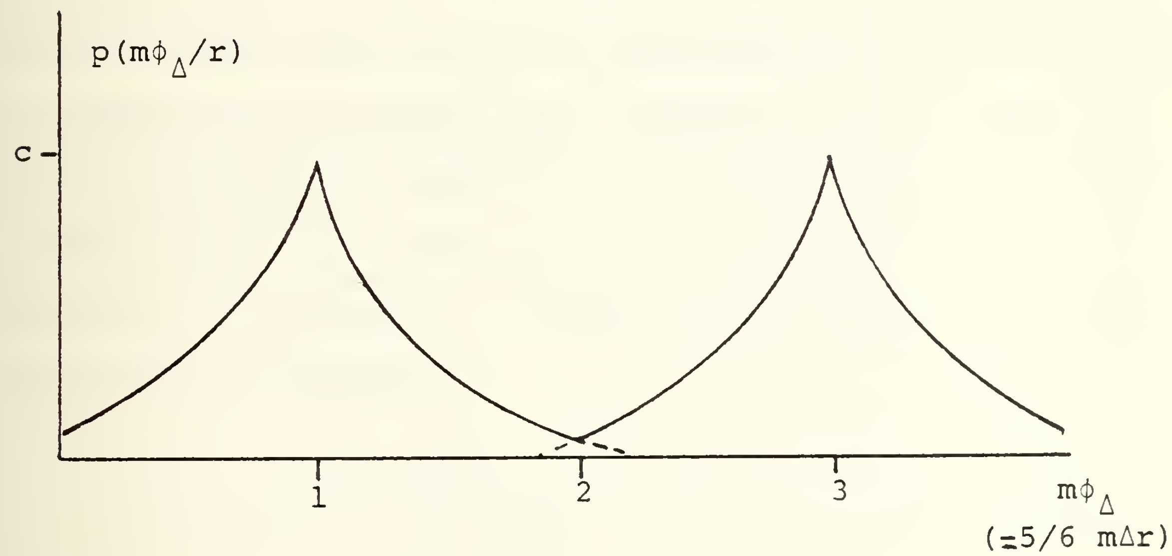

Figure 12. Laplacian Duration Densities

In terms of the speed r:

$$p(m\phi_\Delta/r) = \begin{cases} ce^{\alpha(5/6\ m\Delta r\ -\ m)}; & m\phi_\Delta \le m \\ \\ ce^{\alpha(m\ -\ 5/6\ m\Delta r)}; & m\phi_\Delta \ge m \end{cases}$$

The parameter α and coefficient c are to be chosen such that $Pr[1\phi_\Delta \ge 2] = Pr[3\phi_\Delta \le 2] = .0135$; that is, the probability of error in sending a dot for a dash or an element space for a character space (and vice versa) is arbitrarily selected to be 1.35%. This symbol error rate was found to be the average error using optimum separation thresholds for 55 samples of hand-keyed Morse studied in the TSC analysis [13]; and since the densities are conditioned on the instantaneous speed, the normalized optimum threshold is halfway between m = 1 and m = 3. On this basis, then, α and c are determined as follows:

$$\begin{aligned} Pr[1\phi_\Delta \ge 2] &= \int_2^\infty p(1\phi_\Delta/r)\, d\phi_\Delta \\ &= \int_2^\infty ce^{\alpha(1-\phi_\Delta)}\, d\,\phi_\Delta \\ &= c/\alpha\ e^{-\alpha} \end{aligned}$$

Likewise:

$$\Pr[3\phi_\Delta \leq 2] = c/\alpha \, e^{-\alpha}$$

The probability density requirement gives the other equation needed:

$$\int_{-\infty}^{\infty} p(m\phi_\Delta/r)\, d\phi_\Delta \equiv 1$$

$$\int_{-\infty}^{1} ce^{\alpha(\phi_\Delta - 1)}\, d\phi_\Delta + \int_{1}^{\infty} ce^{\alpha(1 - \phi_\Delta)}\, d\phi_\Delta = 1$$

$$c/\alpha + c/\alpha = 1$$

$$c = \alpha/2$$

Solving for α, c gives, for dots, dashes, element spaces, character spaces:

$$\alpha = 3.61$$

$$c = 1.81$$

Using the same procedure for word space (m=7) and pause (m=14), the values for the densities are:

word spaces: $\alpha = 1.81$, $c = .90$

pause: $\alpha = .90$, $c = .45$

Having constructed the duration densities, the speed-conditioned keystate transition probabilities can now be determined.

Let D_o be the current normalized keystate duration, i.e., the amount of time (in terms of instantaneous element duration) since the last 0 to 1 or 1 to 0 transition. Then the required probabilities are $Pr[\phi_\Delta \geq D_o + \varepsilon/x_{k-1}, a_k, r_k, \phi_\Delta \geq D_o]$, where ε is the normalized sampling interval given by $\varepsilon = \tau/\Delta$. It is seen that this expression gives the transition probabilities in terms of the probability of extending duration D_o for one more sample interval. The conditioning parameters provide the normalization coefficients to be used for $p(m\phi_\Delta/r)$. Given the appropriately scaled density then,

$$Pr[\phi_\Delta \geq D_o+\varepsilon/\phi_\Delta \geq D_o] = \frac{Pr[\phi_\Delta \geq D_o+\varepsilon;\phi_\Delta \geq D_o]}{Pr[\phi_\Delta \geq D_o]},$$

but $\varepsilon > 0$, so $D_o+\varepsilon > D_o$, and the joint probability becomes:

$$Pr[\phi_\Delta \geq D_o+\varepsilon;\phi_\Delta \geq D_o] \equiv Pr[\phi_\Delta \geq D_o+\varepsilon],$$

and so the conditional probability is given by:

$$Pr[\phi_\Delta \geq D_o+\varepsilon/\phi_\Delta \geq D_o] = \frac{Pr[\phi_\Delta \geq D_o+\varepsilon]}{Pr[\phi_\Delta \geq D_o]},$$

where $Pr[\phi_\Delta \geq D_o]$, $Pr[\phi_\Delta \geq D_o+\varepsilon]$ are computed as follows:

$$\Pr[\phi_\Delta \geq D_o+\varepsilon] = \int_{D_o+\varepsilon}^{\infty} p(\phi_\Delta)\, d\phi_\Delta$$

$$= \begin{cases} \frac{1}{2}e^{-\alpha(D_o+\varepsilon-m)} & ; \quad D_o+\varepsilon \geq m \\ 1-\frac{1}{2}e^{\alpha(D_o+\varepsilon-m)} & ; \quad D_o+\varepsilon \leq m \end{cases}$$

Similarly:

$$\Pr[\phi_\Delta \geq D_o] = \int_{D_o}^{\infty} p(\phi_\Delta)\, d\phi_\Delta$$

$$= \begin{cases} \frac{1}{2}e^{-\alpha(D_o-m)} & ; \quad D_o \geq m \\ 1-\frac{1}{2}e^{\alpha(D_o-m)} & ; \quad D_o \leq m \end{cases}$$

Forming the quotient of these probabilities in the appropriate ranges gives:

$$\Pr[\phi_\Delta \geq D_o+\varepsilon/\phi_\Delta \geq D_o] = \begin{cases} e^{-\alpha\varepsilon} & , \quad D_o \geq m \\ \dfrac{1-\frac{1}{2}e^{\alpha(D_o+\varepsilon-m)}}{\frac{1}{2}e^{-\alpha(D_o-m)}} & , \quad \begin{array}{l} D_o \leq m \\ D_o+\varepsilon \geq m \end{array} \\ \dfrac{1-\frac{1}{2}e^{\alpha(D_o+\varepsilon-m)}}{1-\frac{1}{2}e^{\alpha(D_o-m)}} & , \quad D_o+\varepsilon \leq m . \end{cases}$$

The above expression then represents the keystate transition probability for the "transitions" 1-1 and 0-0, conditional on the current symbol type, data rate, and length of time already in state 1 or 0. The probabilities for the transitions 1-0 and 0-1 are found, obviously, by subtracting from 1.

B. SPEED TRANSITION MODEL

The random control vector u_k may contain components which model operator sending peculiarities such as random insertions of extra dots, slurs, character splitting, or any other feature of interest which controls the manner in which encoding takes place; it is not limited to speed control alone. However, the peculiarities mentioned above are highly individualistic and little modeling of these peculiarities has been done. It is conjectured that such modeling will have the same fate as that of attempting to obtain a general parametric model of the keystate duration densities; that is, no general model will be found, and such modeling will require on-line estimation techniques. For the purposes of the HKM model developed here, these peculiarities are ignored, and the only component of the control vector u_k considered is the instantaneous speed r.

The speed transition probabilities are developed on an intuitive basis seasoned with experience and the results of the TSC study on observed hand-sent code speed variability. In that study it was found that, on the average, hand-sent

code exhibits a speed difference of about 2.5 wpm between segments of 10 mark-space pairs, but that it is not uncommon to observe a speed difference of 8-10 wpm between segments. Now observing that the speed transition probability expression of the HKM model, $p(u_k | u_{k-1}\ \alpha_{k-1}\ \beta_{k-1}\ \lambda_{k-1})$, allows for conditioning on the entire past history of the state of the HKM process, it can be seen that this transition probability may take into account such items as message duration (for modeling the effect of operator fatigue), the actual text itself (for modeling the effect of speed changes due to sending different types of text material), or any other feature which may have an effect on sending speed. The only conditioning to be considered here, however, is the immediate past speed u_{k-1}, the past history of the encoded output, α_{k-1}, and the keystate duration β_{k-1}. Let $R_i \in \{i;\ 10 \leq i \leq 60,\ i \text{ an integer}\}$; that is, a set of discrete speeds in wpm between 10 and 60 wpm. The following model for $p(u_k | u_{k-1}; \cdot)$ is proposed:

If $\beta_{k-1} \neq 0$ (no change in keystate), then

$$p(u_k | u_{k-1}\ \alpha_{k-1}\ \beta_{k-1}) \triangleq Pr[u_k = R_i | u_{k-1} = R_j, \alpha_{k-1}, \beta_{k-1} \neq 0]$$

$$= \begin{cases} 0, & \text{if} \quad i \neq j. \\ 1, & \text{if} \quad i = j. \end{cases}$$

That is, the speed is not allowed to change except when the keystate changes from 0 to 1 or 1 to 0, no matter what the previous symbol is. For $\beta_{k-1} = 0$, the speed transition probabilities are made conditional on the type of Morse symbol just completed:

For $\alpha_{k-1} \rightarrow$ indicates dot, dash, e-sp:

$$Pr[u_k = R_j \pm 2i | u_k = R_j, \alpha_{k-1}, \beta_{k-1} = 0] = p_{ji}(\alpha_{k-1})$$

where $i = 0, 1, 2$.

This assignment of tansition probabilities allows the speed to change by increments of 0, ±2, ±4 wpm according to the probability $p_{ji}(\alpha_{k-1})$.

For $\alpha_{k-1} \rightarrow$ indicates c-sp, then the increment remains the same, but the transition probability assignments may be different.

For $\alpha_{k-1} \rightarrow$ indicates word-sp, the increment is increased to 5, and for $\alpha_{k-1} \rightarrow$ indicates pause, the increment is 10.

To complete the model, the $p_{ji}(\alpha_{k-1})$ remain to be selected. These probabilities, which were selected on the basis of speed differences reported by TSC (and on intuitive appeal), are given in Table X.

Note that the absolute average speed differences for the four categories correspond roughly to the ranges observed by TSC.

TABLE X

Symbol-Conditional Speed Transition Probabilities

Symbol Just Completed	Speed Increment/Probability (wpm)					Average Increment (wpm)
dot, dash, e-sp	-4	-2	0	2	4	1.6
	.1	.2	.4	.2	.1	
c-sp	-4	-2	0	2	4	2.0
	.15	.2	.3	.2	.15	
w-sp	-10	-5	0	5	10	4.0
	.1	.2	.4	.2	.1	
pause	-20	-10	0	10	20	10.0
	.15	.2	.3	.2	.15	

C. MORSE SYMBOL TRANSITION MODEL

The symbol transition probabilities, conditional on the letter being sent, are obviously either zero or 1, since knowing the letter specifies the code sequence. If the model is only a first or second-order Markov model, then the symbol transition probabilities for various types of text may be computed. Since it is desired to test the performance of the estimator as a function of modeling complexity, these probabilities were estimated for both a first and second order model and are given in Tables XI and XII, respectively.

TABLE XI

First-Order Markov Symbol Transition Matrix

	.	-	^	∿	w	p
.	0	0	.58	.33	.07	.02
-	0	0	.54	.37	.07	.02
^	.55	.45	0	0	0	0
∿	.5	.5	0	0	0	0
w	.5	.5	0	0	0	0
p	.5	.5	0	0	0	0

TABLE XII

Second-Order Markov Symbol Transition Matrix

	.	-	^	∿	w	p
.^	.55	.45	0	0	0	0
.∿	.5	.45	0	0	0	0
.w	.5	.5	0	0	0	0
.p	.5	.5	0	0	0	0
-^	.55	.5	0	0	0	0
-∿	.5	.45	0	0	0	0
-w	.5	.5	0	0	0	0
-p	.5	.5	0	0	0	0
^.	0	.5	.581	.335	.069	.015
^-	0	0	.54	.376	.069	.015
∿.	0	0	.923	.062	.012	.003
∿-	0	0	.923	.062	.012	.003
w.	0	0	.923	.062	.012	.003
w-	0	0	.923	.062	.012	.003
p.	0	0	.95	.04	.009	.001
p-	0	0	.95	.04	.009	.001

The encoder memory function, f_α, may be constructed to record the previous symbol for the first-order model, or the previous two symbols in the second-order case. In case the symbol transition probability is made conditional on the letter being sent, there is no need to record previous symbols for use by the encoder. As a minimum, however, the function f_α must record the previous symbol for use by the speed transition probability, since it has been made conditional on this symbol.

D. TEXT LETTER TRANSITION MODEL

For equally likely independent letters, the letter transition probabilities are uniform, and the only conditioning necessary is on α_{k-1} so that when α_{k-1} indicates the end of a letter, the letter transition is allowed to occur. During the period when α_{k-1} does not contain a c-sp, w-sp, or pause, obviously the letter transition probability is zero. This case of equally likely letters is the highest complexity modeling actually coded and tested in this investigation. It is clear from the theoretical error-rate analysis of section III, however, that the largest payoff in terms of increase performance is to be found in more sophisticated models for this transition probability and memory function. This fact was recognized early by Gold [12] in his study of the Morse decoding problem, in which he developed the MAUDE algorithm for decoding of the demodulated Morse waveform: "The conclusion is inescapable,

therefore, that for the automatic reception of a language encoded by even a simple process like Morse code, a machine must have some knowledge of the language if it is to approximate the performance of a man."

The major difficulty, however, in modeling the message text is that the type of text is not constant. The letter dependencies are highly variable among such traffic types as call-up, response, chatter, formatted messages, plain language messages, code groups, etc. Here again, then, it is conjectured that the only real solution is to perform on-line modeling of this transition probability and memory function. Clearly a straightforward application of probability estimation techniques, while feasible, is simply not practical in this case. For a third-order model, the storage requirements would be on order of $36^4 = 1,679,616$ words, just to store the transition probability matrix. The f_σ function would require 36^3 locations to keep track of the three prior letters. Although some reduction in memory could be accomplished since some letter combination rarely occur, it is evident that the storage requirement is large. The most promising technique for utilizing the decrease in source entropy may be one similar to that for recognition of speech using a linguistic statistical decoder [15], with appropriately modeled linguistic elements and using an appropriate channel model [16]. If a suitably flexible grammar for a set of Morse messages can be defined

then perhaps a form of syntactic decoding is in order [17]. If the semantics of the message are well-understood then one possible approach is to use a dictionary look-up to form the f_σ function, on a word basis. This technique for English text messages is under investigation by an ARPA-funded MIT project, but a final report of the results has not yet been issued. The Army Research and Development Agency is currently studying the possibility of defining a grammar for a specified set of Morse messages for use in syntactic decoding. These kinds of techniques for dynamic on-line construction of the f_σ function and estimation of the transition probabilities are clearly the only realistic methods of reducing the entropy of the text sufficiently to obtain error rates comparable to that of the human operator, in any situation except for random letter groups.

VII. A PRACTICAL HKM CHANNEL MODEL

The general baseband HKM channel model developed in Section IV is given by the channel and observation equations (10):

$$y_k = \gamma\, F(s_k\, \sigma_{k-1})\, y_{k-1} + \Gamma(s_k\, \sigma_{k-1})\, w_k$$

$$z_k = H(s_k)\, y_k + n_k$$

where z_k is the sampled output of the detector. The specific model to be considered here requires the parameter γ and functions F, Γ, H, to be selected such that the resulting model has the following features:

(1) The noise process represented by n_k is a zero-mean white gaussian process, with known variance R_k.

(2) The amplitude y_k is observed only when $x_k = 1$, that is, during the signal on-time (MARK), so that $H(s_k) = H(x_k) \equiv x_k$.

(3) During a MARK, the fading amplitude process obeys a linear gauss Markov process given by:

$$y_k = \gamma\, y_{k-1} + v_k$$

where the parameter γ and the variance of v_k are selected to represent the fading observed at the detector output.

(4) The observed effective transmitted amplitude is a random variable which obeys the following time-varying linear gauss-Markov process:

$$y_k = F(x_k \; a_k \; \beta_{k-1})y_{k-1} + \Gamma(x_k \; a_k \; \beta_{k-1})w_k$$

where F and Γ are selected such that:

(a) During a MARK the transmitted amplitude remains constant.

(b) During a space the amplitude can change, the amount of change being dependent on the type and duration of the space.

(5) It is assumed that the detected signal has been gain-leveled by an AGC, so that the average detected output power is normalized.

The parameter selection and function construction process for each of these features is discussed below.

A. THE OBSERVED NOISE PROCESS

Since the noise process observed at the output of the detector is the result of envelope detection of a narrowband gaussian process, the resulting process is neither zero-mean, gaussian, nor white. The sampled process, however, has independent noise values if the sample interval τ satisfies $\tau \geq 1/2\ B_{BPF}$, where B_{BPF} is the bandwidth (in Hz) of the band-pass filter preceding the envelope detector, provided that also the bandwidth of the low-pass filter of the envelope

detector is greater than $2B_{BPF}$. If τ is less than this value, then the sampled noise is correlated, and a model which accounts for this correlation would theoretically provide for better estimation. Several techniques are available for such modeling, [18] and should be used if the noise is correlated. Clearly if τ is selected purely on this basis alone, then the assumption on independence can be satisfied. There may be, however, other competing constraints on the selection of τ, and although the value selected may render the independent noise assumption invalid, its effect can be minimized by selecting it as large as possible within the other constraints.

The bandwidth of the bandpass filter is selected on the basis of the largest signal bandwidth expected. The highest code-speed under consideration for this processor design was selected to be 50 wpm, which has a minimum pulse duration (MARK) of 24 msec. The specific filter implementation was selected to be a cascade of two single-tuned resonators, since this combination has a respectable ratio of noise-bandwidth to 3-dB bandwidth of 1.22 [19], and can be coded with relatively few multiplication per sample. For this filter implementation the optimum bandwidth as given by Skolnik [19] is .613/.024 = 25 Hz, and has only .56 dB of loss in SNR compared to the matched filter. Although such a narrow bandwidth greatly increases the SNR of a signal in a 4 kHz receiver bandwidth and effectively eliminates

most interferers, it is clearly too narrow to accept signals which have a significant carrier instability due to chirp or drift. Since it is not uncommon to observe carriers with a chirp on the order of 50 or so Hz, the bandwidth required is on the order of 100 Hz. There is obviously a strong motivation, therefore, to investigate filtering techniques which would adapt to the chirp, since a 100 Hz wide filter represents a loss of 6 dB compared to the optimum bandwidth of 25 Hz. Motivation for adaptive filtering techniques is also provided by the fact that at 20 wpm the optimum bandwidth is only .613/.060 = 10 Hz, thus there is a 10 dB loss in SNR compared to the optimum bandwidth when using a 100 Hz filter.

For this investigation, since the primary emphasis is on optimum demodulation and decoding techniques, a fixed 100 Hz band-pass filter is used. For this bandwidth, then, the sample rate may be selected to be 200 Hz, with a resulting sample interval of 5 msec. Since this quantization is considered adequate for representing the minimum duration 24 msec-long pulse of the 50 wpm code with sufficient precision, then τ is selected to be 5 msec., resulting in independent noise samples.

Since approximately 5 msec. is the largest quantization allowable for adequate precision in representation of the code symbols, and since adaptive techniques for the band-pass filter would result in narrower bandwidths, the assumption

on independent noise samples would be violated for this case, requiring a model which accounts for correlated noise, if optimum techniques are to be pursued.

Although the zero-mean assumption on the output noise process is violated, a zero-mean process may be approximated by estimation of the mean and subtraction of it from the detected output. Estimation of this mean value also provides an estimate of the noise variance, R_k, which has been assumed to be a known value throughout. (Again, although techniques are available for modeling in the case of unknown noise intensity, the simplified approach taken here is to use the estimate of R_k as if it were the true value. It can be seen in section IX, Table XIII, that the resulting processor is relatively insensitive to $\hat{R}_k$, as long as $\hat{R}_k$ is within a rather large range of the true value.) Estimation of the mean noise level relies on the following relationships.

Let X_t be a white gaussian random process with one-sided density N_o, input to the BPF; let Z_t be the output of the envelope detector, with $B_{LPF} \geq B_{BPF}$ as illustrated below:

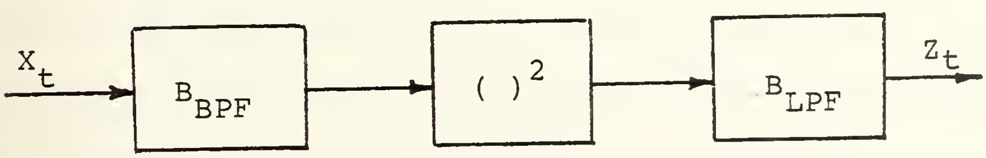

Figure 13. Envelope Detection Process

Then, from Davenport [20],

$$\mu_n \triangleq E(Z_t) = N_o B_{BPF}$$

$$R_n \triangleq Var(Z_t) = 2(N_o B_{BPF})^2$$

Thus if μ_n can be estimated in the absence of a MARK, then

$$\hat{R}_n = 2 \hat{\mu}_n^2$$

and the approximation to a zero-mean process is $Z_t - \hat{\mu}_n$. Implementation of such an estimator is described in Section VIII.

The assumption of a gaussian process for n_k is clearly violated since the output of the detector has a Rayleigh density in the absence of a MARK, and a Rician density when signal is present. Thus not only are the statistics not gaussian, but also they are correlated with the signal when a MARK is present. By choosing to ignore the higher-order moments of the density (greater than 2), the resulting estimator based on this assumption may not be optimal in the sense of providing as good a conditional-mean estimate as possible, but it will still provide the minimum-mean-squared-error estimate.

B. THE MEASUREMENT FUNCTION

During the period when $x_k = 0$, the transmitter is turned off and it is not possible to observe the amplitude which is being used to transmit the MARKS. Thus only noise is observed during this period, and by ignoring the correlation between signal and noise when signal is present, the measurement equation is simply:

$$z_k = x_k\, y_k + n_k$$

C. FADING MODEL

The effect of fading can be observed during a MARK period, with the maximum fade rate being determined by the band-pass filter/dectector bandwidth, under worst-case HF channel conditions (rapid, intense fading). For typical values of fading rate on the order of 1 Hz, the fading parameter γ, for a 5 msec sampling interval is given by:

$$\gamma = e^{-(.005)(2\pi)(1)} = .97$$

The intensity observed at the output of the gain-controlled detector can be approximated for the typical 1 Hz fade rate by noting that during a 1 sec fade period the amplitude can change by about 3 dB for a typical receiver AGC circuit. The intensity for this range of change, i.e., the variance of v_k is about:

$$\mathrm{Var}\ (v_k) \cong [2/(1./.005)]^2 = [2/200]^2 = .0001.$$

As discussed earlier, in Section IV.B, when no signal is present, the effect of fading is that the subsequent MARK appears at an amplitude which differs from the amplitude of the previous MARK in such a way that it appears as if the MARKS of the signal were transmitted at a random amplitude. Because of this effect, these mark-to-mark variations are lumped together with the variations caused by an actual change in transmitted power.

D. APPARENT TRANSMITTER POWER VARIATIONS

In addition to the Mark-to-Mark amplitude variations discussed above, the actual transmitted power may vary. Usually this effect is most prominent when working with a communications net, since the received power of each of the transmitters on the net will usually be different. These changes usually occur after a pause (during which one net member has signed off and another is preparing to sign on); however,, it is not uncommon for a new net member to sign on during a time duration for a word space or even a character space, especially if net discipline is good. It is assumed that changes do not occur during an element-space or a mark. The following model accounts for these effects:

a) For $\alpha_{k-1} \rightarrow$ mark:

$$Q_W = \mathrm{Var}(v_k) = .0001$$

$\gamma F(x_k \; a_k \; \alpha_{k-1} \; \beta_{k-1}) = \gamma = .97$

b) For $\alpha_{k-1} \rightarrow$ element space; $x_k = 0$:

$Q_w = 0.$

$\gamma F(\cdot) = 1.$

c) For $\alpha_{k-1} \rightarrow$ element space; $x_k = 1$:

$Q_w = .01$

$\gamma F(\cdot) = 1.$

d) For $\alpha_{k-1} \rightarrow$ any other space; $x_k = 0$:

$Q_w = 0.$

$\gamma F(\cdot) = .98$

e) For $\alpha_{k-1} \rightarrow$ any other space; $x_k = 1$:

$Q_w = .25$

$\gamma F(\cdot) = 1.$

Part (a) is just the fading model for Marks discussed above. Part (b) expresses the statement that no change in amplitude may occur during an element space. Part (c) states that, at the end of an element space the transmitted amplitude has not changed, but a variance of .01 is associated with the amplitude observed on this transition. The value .01 is obtained by considering that at the end of an element space transmitted at 50 wpm, the fade may have decreased the amplitude to $(.97)^4 = .89$ of its previous value, thus a variance of $(1 - .89)^2 \cong .01$ is appropriate. Part (d) states that for any other space, while the variance associated with the transmitted amplitude is zero, the amplitude is assumed to decrease exponentially with time at the rate (.98); and Part (e) allows a subsequent MARK to appear with amplitude determined by a gaussian random variable of variance .25. (The construction of the $\Gamma(\cdot)$ function is implied by the assignment of variances to the various Q_w.)

VIII. IMPLEMENTATION OF HKM STATE ESTIMATION ALGORITHM

The implementation of the estimator algorithm (Eqn. 26, 30) for the signal and channel models just described is now presented. In the context of this model, estimation of the keystate is referred to as demodulation, estimation of the Morse symbol is termed decoding, and estimation of the text letter is called translation. The estimation algorithm performs joint demodulation, decoding and translation, i.e., these estimates are not made in a serial fashion; rather the structure of the code is used in an optimal way to aid in demodulation, and the structure of the text is used to aid in decoding. From this viewpoint the algorithm represents a "correlator-estimator" [21] technique in which a sequence of all possible keystate transitions are hypothesized and correlated with the incoming signal, and the most likely sequence is output as the best estimate. From the viewpoint of coding theory, the algorithm represents a tree decoder in which all possible paths of the joint state evolution of the process are examined and extended in an optimal way. If the memory function were dependent on only a finite portion of the past history of the process (usually a good approximation) then the tree decoder reduces to the Viterbi decoder. As implemented herein, the decoder is most like the M-Path algorithm described by Haccoun [22], with the path metric being the product of the likelihood of the

received signal along the path and the transition probability for the path extension. If the decoder is constrained to save only one path, then the decision-directed optimal linear filter investigated in [2] is obtained.

Proceeding now to a detailed description, the algorithm is presented in terms of the Fortran code used to implement it. Subroutine PROCES is the main calling routine which takes an input signal sample each 5 msec, along with an estimate of the noise power, and calls the appropriate routines in order. The first routine called for each sample point is TRPROB, which computes, for each previously saved path ending at node J, the probability of extending the path to new nodes which are labeled to indicate the joint state (keystate, element state, letter state, data rate). These probabilities are computed using the model and equations described in the previous section. Next, subroutine PATH labels the new path extended to each new node with: (1) the number of samples since the previous keystate transition along that path; (2) the data rate of the new node; (3) the identity of the element state at the new node; (4) the identity of the letter state at the new node. These labels are obtained from the memory function f_σ with arguments provided by the identity of the path being extended and the identity of the new node to which the path is being extended. Subroutine LIKHD is then called to compute the likelihood of the input signal sample for each transition under the hypothesis that that particular transition occurred.

LIKHD maintains an array of Kalman filters for computing this likelihood as given in Section V.A by equation (30), and using the specific channel model described in the previous section.

Having obtained the new path identities, transition probabilities, and likelihoods, the posterior probability of each new node (i.e., each path extension) is computed using equation (26), in subroutine PROBP. Next, routine SPROB computes the posterior probability of each keystate (0,1) and each element state, and the conditional mean estimates of the data rate, by summing over the appropriate nodes. The MAP estimate of the keystate at this point is the demodulated signal, and the conditional mean estimate of the keystate is the (non-linear) filtered version of the detected signal. Also the evolution of the MAP estimator for the element state may be observed at this point, and represents the decoded message with zero decoder delay.

The next function to be accomplished is the saving of paths for the next iteration. It is at this point that the estimation algorithm becomes sub-optimal, since it is clearly not possible to save all paths at each stage of iteration. A technique which yields a high probability that the correct path will always be saved obviously provides the best sub-optimal performance. Several techniques for selecting the paths to save are available. The simplest idea is to always save a fixed number, say

M_{max}. It was determined empirically, however, that, while this technique does indeed give a high probability of saving the correct path, most of the time the posterior probabilities of many of the saved paths were very low and need not be extended at all. At the instant of a keystate transition, however, the probabilities become more uniform and it is necessary to save all the M_{max} paths. The next technique then was to save only enough paths such that the total probability saved was equal to P_{opt}, subject to the constraint that M_{max} is not exceeded. Another technique suggested by [22] is to make the number of paths saved a function of the probability of the highest probability path, such that when the highest probability path has a very high probability, fewer paths are saved. Either of the last two techniques has the attractive feature that the decoding computational burden is adaptive to the signal-to-noise ratio and the data rate, and the first of these was selected for use, with the additional constraint that at least one path for each element state is always saved. This algorithm is coded in subroutine SAVEP.

Also in subroutine SAVEP, the saved paths and their identities are renumbered in order of decreasing probability and a pointer array is maintained to identify the previous node from which the saved path was extended. Additionally, the parameters of the Kalman filters are reindexed to be consistent with the new path indices. After action by SAVEP, then, the arrays are ready for the next iteration.

Before proceeding to the next iteration, however, the trellis of saved paths is updated with the new saved nodes and connected to the proper previously saved paths by using the pointer array. Decoding and translation are accomplished within subroutine TRELIS by operating on the trellis of saved paths. Decoding is done by finding the one node, at sufficient delay, from which all successor paths originate. If no such single node exists within the trellis for a maximum delay of 200 samples (1 second delay) then decoding is obtained by reading the node at delay 200 which is connected to the current highest probability path, and all other paths not originating from this node are deleted from the trellis. Since the text has been modeled by a source of equiprobable, independent letters, translation is done by a simple mapping of the decoded Morse symbols into the proper letters and numerals.

There are three auxiliary processing routines for preprocessing of the signal, intended to simulate the operation of a receiver, bandpass filter and envelope detector, along with the routine to estimate the noise power in the detected signal and provide a zero-mean noise process. Subroutine RCVR converts the incoming signal at carrier frequency ω_o to a frequency of 1000 Hz using an 8 kHz sample rate, and provides a single-pole 500 Hz BW band-pass filter. Subroutine BPFDET implements the 100 Hz bandwidth band-pass filter by a series of two digital resonators centered at

1000 Hz, and accomplishes envelope detection. The low pass filter of the envelope detector is a 100 Hz bandwidth 3-pole Chebyshev filter. Subroutine NOISE estimates the noise power present during a space condition by obtaining the minimum value of the envelope detected signal over a period of 240 samples (1.2 seconds). This minimum value is obtained at each 5-msec sample point and averaged. The average is then scaled, with the scale parameter selected empirically, to provide the estimate of μ_n, the mean value of the envelope detected output during a space. This estimate is subtracted from the envelope detector output to provide an approximation to a zero-mean noise process; RN, the estimate of noise power in the detected output is then given by $2\hat{\mu}_n^2$.

IX. SIMULATION RESULTS

The Fortran coded algorithm just described has been programmed on a PDP-10 time sharing system, along with a signal simulation routine to generate a Morse code message, a routine to simulate transmitter effects, and a channel model routine. The text generation routine selects letters and numerals either at random or from a pre-defined text file. The corresponding Morse code sequences are generated by a table look-up, and the durations of each element are randomized according to a selectable probability law. (For the results presented here, the probability law used was a truncated gaussian such that no element is ever less than 16 msec or greater than 360 msec in duration. The variance was selected to give the error crossover probabilities on an element basis to correspond to the good, fair, and poor operator defined in section III.B.) The waveform generated by this process is used to modulate a carrier of frequency $\omega_o \leq 4$ KHZ, which is simulated by discrete-time process sampled at 8 kHz. This carrier is then subjected to the fading model (VII.C) and white gaussian noise of selectable power is added. This received carrier is then input to the receiver, bandpass filter and detection routines discussed previously. The output of the envelope detector, adjusted in level by subroutine NOISE, is then input to the main processing algorithm, PROCESS; the demodulated, decoded

and translated results are presented on a CRT from which hard copies may be obtained.

The overall objective of the simulation experiment is to determine how well the finite-path suboptimal estimator performs relative to the optimal estimator. Since it is not possible to code the exact optimal estimator due to exponentially expanding memory and computation, the lower bounds an error rate derived in Section III are used as a basis for comparison. Secondly the performance of the tree decoder (the term tree decoder will be used to refer to the suboptimal finite-path estimator) relative to other simpler techniques is to be evaluated. Finally the performance of the tree decoder as a near-optimal demodulator for Morse-code is to be obtained and compared to the performance of the linear matched filter with integration time equal to the basic element duration.

A. THE IDEALIZED KAM TREE DECODER

The idealization assumptions made in Section III for deriving the lower bounds on error rate can be obtained by constraining the estimation algorithm to have path branching only at the possible transition times of a synchronous KAM signal, and by making the input a true baseband Morse waveform with added white gaussian noise and no fading. This experiment was run in order to determine the validity of the lower bounds derived there and to obtain a data base for evaluating the sensitivity of the tree decoder to

non-ideal conditions. The results of this experiment are shown in Figure 14 for the three cases of first-order and second-order symbols and independent letters. Clearly under these ideal conditions the lower bound is very nearly obtainable.

Also shown for comparison are the results of demodulation accomplished by linear matched filtering with decoding accomplished by thresholding the durations at 2T, where T is the basic element duration. These results show that the demodulation provided by the tree decoder is clearly superior to the matched filter, and that the independent letter model is of sufficient complexity to obtain near-optimal demodulation.

Next, the effect of lack of synchronization was obtained by removing the branching constraint on the paths, but still keeping the same idealized input signal. The results are shown in Figure 15. By comparing with the results for the synchronous case, it is obvious that at the lower SNR's the performance is degraded.

The next effect to be investigated was the sensitivity to noise statistics in the estimator's lack of knowledge of the true noise power. These results, shown in Table XIII, indicate that the estimator is relatively insensitive to incorrect estimates of noise power within a reasonable range.

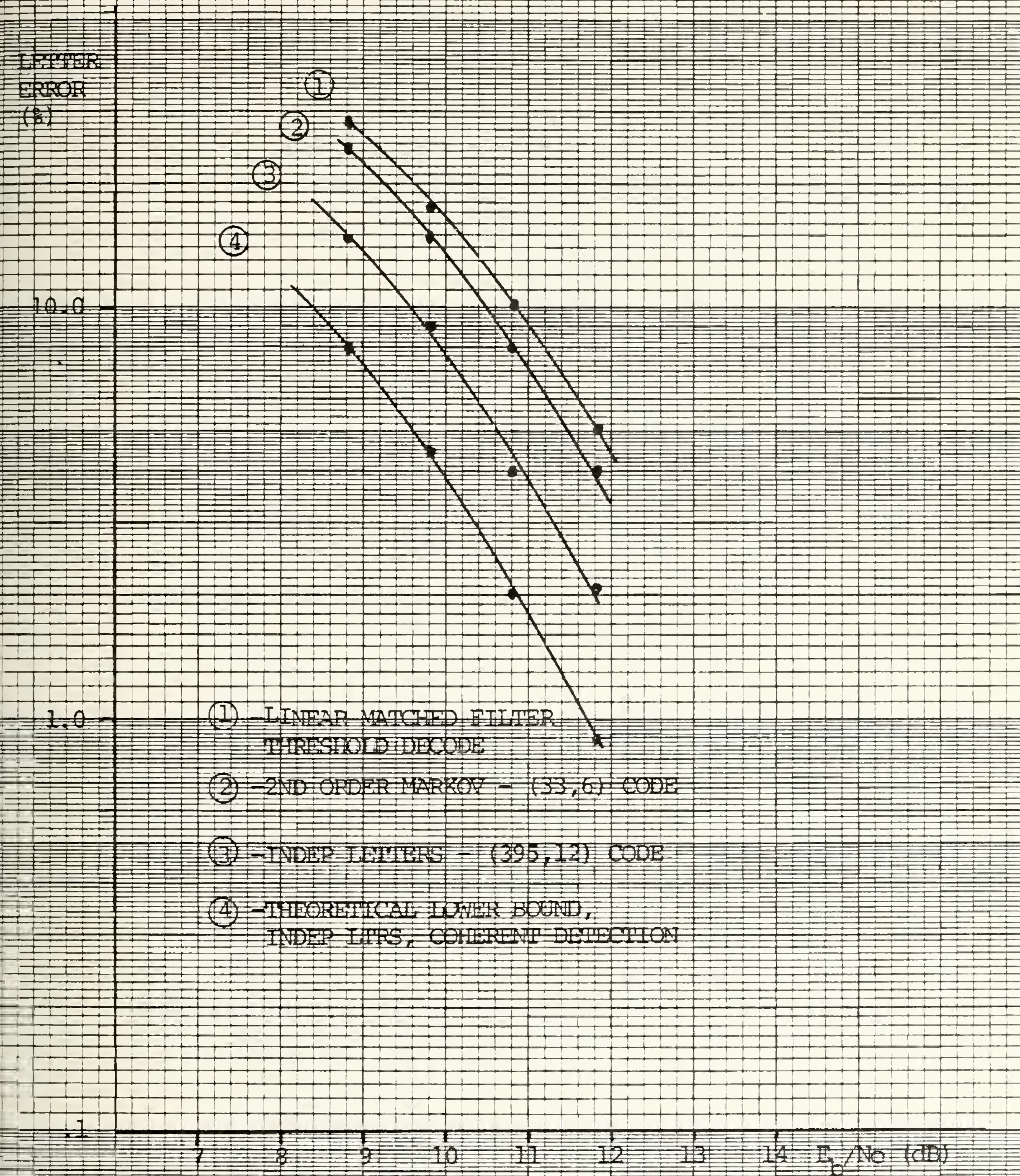

FIGURE 14. Performance of Idealized Synchronouse KAM

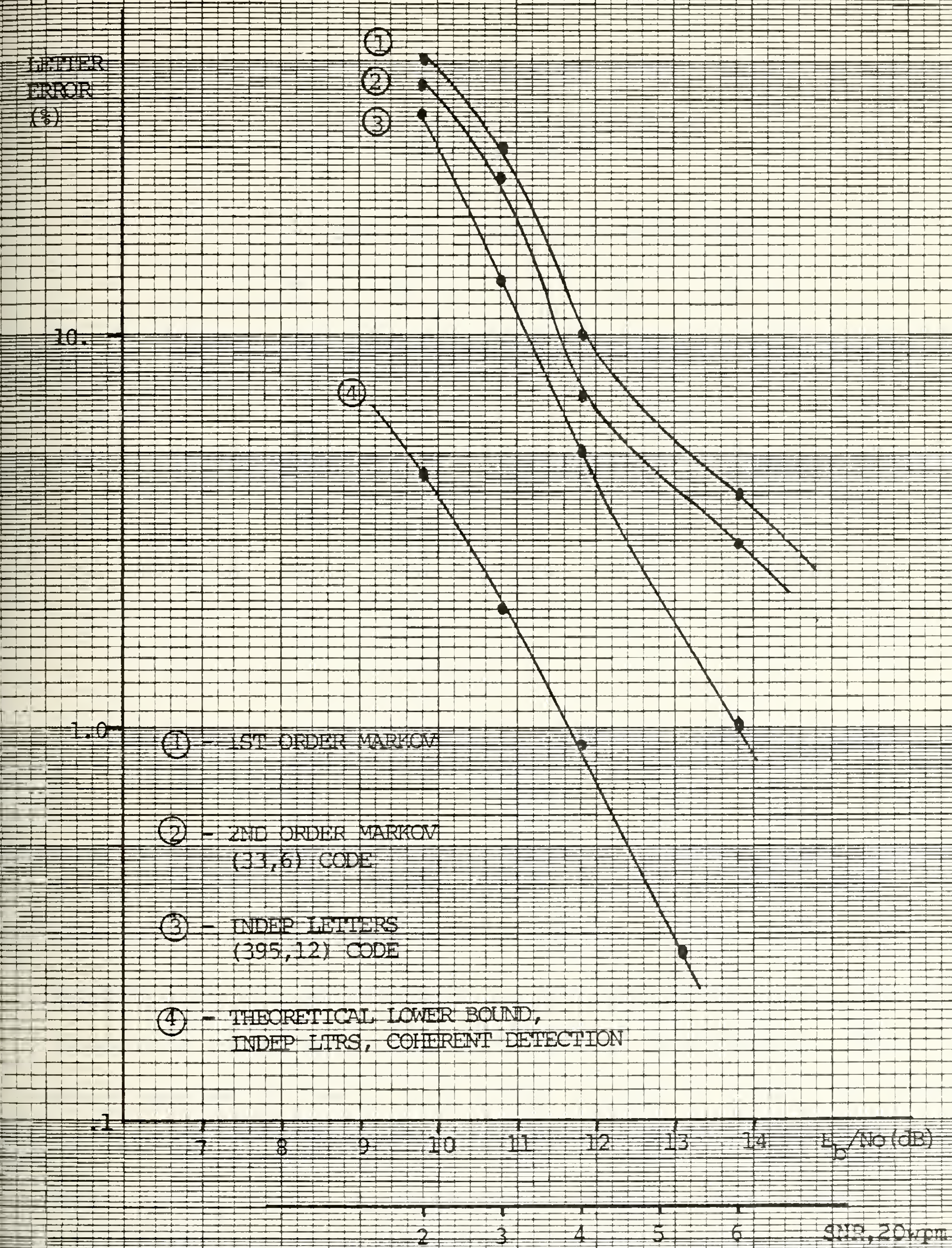

FIGURE 15. Performance of Idealized Non-Synchronous KAM Decoder

TABLE XIII

NOISE POWER EST SENSITIVITY
(20 wpm KAM)

TRUE SNR (dB) (100 Hz)	SNR Est Used by Decoder (dB) 9	6	3	2	1
	% LTR Error				
9	0	-	0	-	0
6	2	1	1	-	1
3	9	6	5	-	5
2	-	19	-	14	14

B. THE REALISTIC HKM TREE DECODER

Although the results discussed above are of theoretical interest since they demonstrate a high degree of correlation with theory, they have little practical value in determining the performance of the demodulator and decoder functions under more realistic signal conditions. The first series of tests used a KAM signal as input, in order to correspond the results to those above for the idealized case and to obtain a basis for comparison with the HKM case. Table XIV shows the performance of the tree decoder as a function of the decoder constraint length (decode delay) and as a function of the degree of optimality of the estimator. (The degree of optimality is given by the

TABLE XIV

Performance of First-Order Markov Decoder vs. Decode Delay and Degree Of Estimator Optimality — 50 wpm KAM

Degree of Optimality (P_{opt})	SNR (100 Hz) dB	Avg. No. of Paths Saved	Decode Delay (Samples) 0 % Error	40 % Error	200 % Error
	12	20	0	0	0
.98	9	20	9	5	5
	6	20	68	45	45
	12	17	0	0	0
.95	9	17	9	5	5
	6	18	68	45	45
	12	14	0	0	0
.9	9	15	12	8	5
	6	15	56	52	46
	12	12	3	3	2
.85	9	12	32	32	29
	6	12	58	56	53
	12	8	3	3	2
.8	9	8	38	39	36
	6	8	68	67	63

parameter P_{opt}, discussed above, where only enough paths are saved such that the sum of the computed posterior path probabilities $\geq P_{opt}$.) These results show that the 90%

optimal estimator with a decode delay of 200 (1 second) is very nearly as good the 98% optimal decoder. These values were selected, then, for the remaining tests. Table XV shows the performance of the tree decoder as a function of model complexity, and the improvement in performance with increasing complexity at the lower SNR's is evident. For comparison the results for the independent letter model are plotted in Figure 16 along with the results for the idealized case, and the lower bound for envelope detection.

TABLE XV

PERFORMANCE OF DECODER vs. MODEL COMPLEXITY — 90% OPTIMAL ESTIMATOR, KAM SIGNAL

		DECODER MODEL			
Speed (wpm)	SNR (dB) (100 Hz)	First Order % Error	Second Order % Error	Indep Char % Error	Avg no. of paths Saved
	12	0	0	0	14
50	9	5	4	3	15
	8	14	11	5	15
	7	36	30	16	16
	6	46	41	35	16
	9	0	0	0	8
20	6	10	6	3	8
	4	12	9	6	9
	3	43	38	31	9

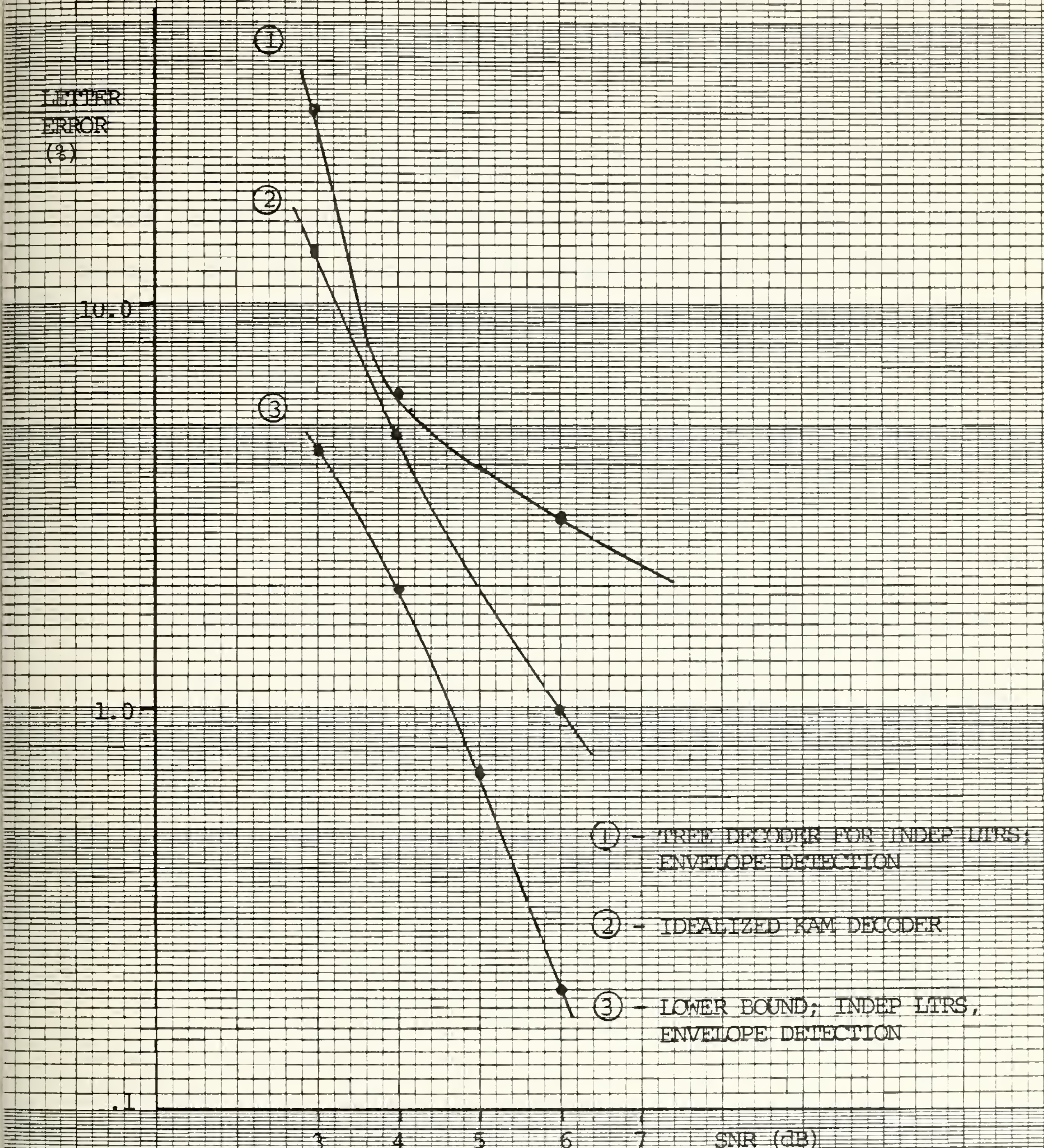

FIGURE 16. Performance Comparison of Idealized Decoder and Decoder Using Envelope Detection, 20 wpm KAM

The next series of tests used a simulated hand-keyed signal as input at nominal speeds of 20 and 30 wpm. The performance for the good, fair, and poor keying characteristics (element error probabilities of .00143, .0149, and .0403 respectively) was evaluated for P_{opt} = .9, and decode delay = 200 as a function of model complexity. These results are tabulated in Table XVI. The result for the fair sender is shown in Figure 17 along with the corresponding result for the KAM signal and the theoretical lower bound.

TABLE XVI

Decoder Performance For Simulated Hand-Keyed Morse

Sending Quality	SNR (dB) (100 Hz)	30 wpm % Letter Error	30 wpm Avg No of Paths Saved	20 wpm % Letter Error	20 wpm Avg No of Paths Saved
	9	3	8	1	9
Good	6	5	8	4	10
(Sending	4	36	9	6	10
Error Rate = 1%)	3	-	9	31	11
	9	5	9	4	10
Fair	6	7	10	6	10
(Sending	4	42	10	8	11
Error Rate = 10%)	3	-	11	34	11
	9	12	11	11	12
Poor	6	13	11	13	13
(Sending	4	46	12	14	13
Error Rate = 25%)	3	-	12	38	14

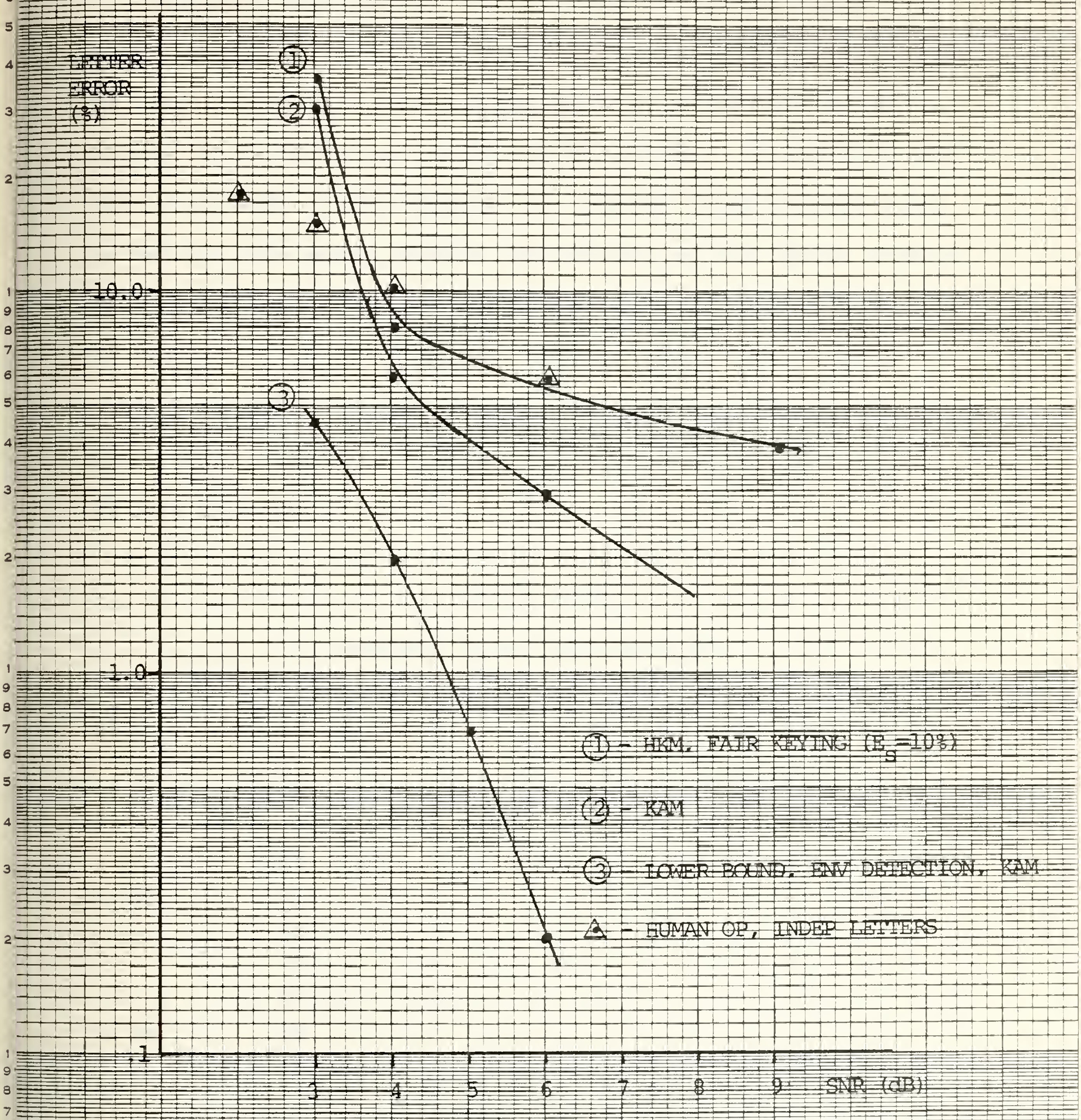

FIGURE 17. Comparison of HKM and KAM Performance, 20 wpm

The adaptability of the decoder to abrupt changes in speed of transmission was next evaluated at several values of SNR. This test was run by causing an abrupt speed change to occur after every tenth letter. The output was then compared to the output for the no speed change case to obtain the extra errors introduced by the speed change. This increase in error caused by speed change is tabulated in Table XVII, as a function of the magnitude of speed change and SNR. A KAM signal was used for the 50 wpm speed, and a fair sending operator was simulated for the 30 and 20 wpm signals.

TABLE XVII

Decoder Speed Adaptability

SNR	Speed of Previous Segment	% Error Increase Over Constant Speed		
		New Speed		
		50	30	20
	50	-	1	2
9 dB	30	0	-	1
	20	1	0	-
	50	-	2	4
8 dB	30	1	-	2
	20	1	1	-
	50	-	5	6
6 dB	30	4	-	4
	20	4	3	-

In order to compare the decoder performance with the performance of the MAUDE algorithm and Howe's quasi-Bayes decoder [14], the decoder was next tested against simulated hand-keyed signals using the same mark/space durations that were used in Howe's tests. The simulated signals consisted of the following keying characteristics:

S1 - Moderate variance handkeyed: Mark-space sequence with nominal 1-3-7 mean element duration ratios and element standard deviation-to-mean ratio of 0.2, nominal sending speed of 15 wpm. ($\bar{E}_s$, the average sending letter-error rate = 10%).

S2 - Abrupt speed changes, low variance handkeyed: Mark-space sequence with nominal 1-3-7 element duration ratios and element standard deviation to mean ratios of 0.15 with abrupt nominal speed changes among 10, 15, 20 wpm rates. ($\bar{E}_s$, each speed segment, = 3%).

S3 - Gradual speed change, low variance manual: Same as S2 above, but with gradual speed changes between approximately 10 and 20 wpm over a period of 30 seconds.

Each of these files was used to modulate a carrier of constant amplitude to which white gaussian noise was added for signal-to-noise ratios of 12 dB, 9 dB, 6 dB referenced to 100 Hz. The results of this test are shown in Table XVIII. A comparison of these results for the high SNR case (the only case considered by Howe) with the performance of the quasi-Bayes and MAUDE algorithms is shown in Table XIX.

TABLE XVIII

DECODER PERFORMANCE FOR SIMULATED HAND-KEYED MORSE USING HOWE'S MARK-SPACE FILES

File	SNR (dB)		
	12 % Error	9 % Error	6 % Error
S1	11	11	24
S2	4	6	11
S3	5	6	13

TABLE XIX

COMPARISON OF TREE DECODER WITH MAUDE AND HOWE'S QUASI-BAYES DECODER, HIGH SNR

File	Decoder Algorithm		
	Tree % Error	MAUDE* % Error	Quasi-Bayes* % Error
S1	11	20	8
S2	4	12	5
S3	5	14	6

* Data for MAUDE & Quasi-Bayes From [14, p. 74].

C. STATISTICAL SIGNIFICANCE OF EXPERIMENTAL RESULTS

The sample size used in each of the experiments described was approximately 200 letters. Since the sample size is greater than 30, and since each experiment was performed under well-controlled conditions, the outcome of each experiment (proportion of letter errors) may be reasonably assumed to be a sample point arising from a gaussian density. Under this assumption, the following 90% confidence intervals [23] are applicable (Table XX).

TABLE XX

90%-CONFIDENCE INTERVAL FOR EXPERIMENTAL RESULTS

MEASURED EXPERIMENTAL ERROR RATE	90% CONFIDENCE INTERVAL
5%	3%- 8%
10%	7%-14%
15%	11%-19%
20%	15%-26%
25%	20%-31%
30%	24%-36%

While the relatively small sample size of 200 letters is adequate for the well-controlled simulation experiments, because of the consistency of the input signals, a much larger sample size would be required for testing against actual data. Because of the lengthy processing time required on the PDP-10 implementation (one minute of data requires approximately 20 minutes of processing time), however, it was not feasible to obtain large quantities of test data against actual signals. The following field results given in Tables XXI and XXII, therefore should be considered a proof of feasibility of the tree-decoder, but not necessarily typical of results to be expected under a wide range of signal and keying characteristics.

X. PRELIMINARY RESULTS FROM FIELD DATA

In order to obtain an estimate of the projected performance of the tree decoder under actual signal and channel conditions, the algorithm was tested against several tape recordings of signals made in the field. Analog tape recordings of the output of a receiver using a 4 kHz IF band width with fast-attack, moderate-speed decay (approx. 200 msec) AGC were made. These tapes were digitized using a sample rate of 8 kHz. Each cut is approximately 50 seconds in duration, resulting in a relatively small, but significant, data base for analysis. The text in each case was context-free, and all signals were of sufficiently high signal-to-noise ratio so that the true transmitted text could be recovered from the detected output. The results of these tests are shown in Tables XXI and XXII for the KAM and HKM signals respectively.

TABLE XXI

PERFORMANCE OF TREE DECODER AGAINST ACTUAL SIGNALS, KAM SENDER

Sample	Data Rate (wpm)	Avg SNR (dB) (100 Hz)	Letter Error (%)
1	35	20	1%
2	30	16	2%
3	28	16	1%
4	32	18	10%
5	30	20	8%

TABLE XXII

PERFORMANCE OF TREE DECODER AGAINST
ACTUAL SIGNALS, HKM SENDER

Sample	Data Rate (wpm)	Avg SNR (dB) (100 Hz)	Letter Error (%)
1	18	20	4
2	16	16	3
3	22	18	15
4	20	20	8

The disappointing results for samples 4 and 5 of the KAM signals are attributed to two effects observed on these cuts. Sample 4 contains several long sequences of high-level "static" or "burst" noise, which appear in the envelope-detected output as energy which is inseparable from true marks of the desired signal. Although these false marks are of lower level than the actual signal, the algorithm assumes that they are faded marks of the incoming signal and demodulates them as such. Although the algorithm successfully rejects many of the shorter spurious marks because they are inconsistent with the speed of transmission, enough are accepted as valid marks to cause the error rate to be high.

In the case of sample 5, all of the errors are attributed to a low level Morse interferer which becomes predominant when the desired signal is in a word space or pause condition.

During these times, the receiver gain is not controlled by the relatively high-level desired signal, and the underlying interferer is of sufficient SNR (approx. 8 dB) to be demodulated by the tree decoder algorithm.

For the HKM cuts, the comparatively high error rates for samples 3 and 4 are attributed to the same type of interference/AGC effect discussed above, although in sample 3 the interferer is one leg of an FSK teletype signal. For all the HKM cuts, the sending quality is rated as good-to-fair.

XI. SUMMARY AND CONCLUSIONS

The extinction of communication by Morse telegraphy has been repeatedly predicted aperiodically since about 1950. While the commercial use of this mode of communications is virtually nonexistent in the U.S., except for some maritime services, it is still used in the military services of many countries. The reliability of Morse links is well-known and long-distance communication, particularly at HF, is possible under conditions of interference and atmospherics which would render other means of communication useless. The simplicity, reliability, and efficiency of the receiver (the human mind) preclude extinction of this oldest form of successful electrical communications.

Radio communication between two persons using Morse code is a distinctly human process, involving nuances of code variations and tacitly assumed conventions between the communicators, which make machine transcription of the human-sent code particularly difficult. The theoretical development of a unified structure for modeling a Morse message (not just the code itself) presented in this report shows how the various aspects of linguistic context, formatting, individualistic operator sending peculiarities, and code symbol dependencies may be combined in the design of an optimal Morse translator. As a practical example of modeling of the Morse message within this structure, a

model for independent equally-likely letter messages was derived, and the resulting decoder was tested against a variety of simulated and actual Morse messages.

The results of the simulations show that the error rate of the idealized KAM decoder [Fig. 14,15] approaches the theoretical lower bound for the gaussian channel, derived from coding theory arguments, and that the increase in performance compared to a linear dot-matched filter can be significant at low signal-to-noise ratios. Secondly, the performance of the HKM decoder using envelope detection [Fig. 16] was demonstrated to be only moderately sensitive to the non-gaussian nature of the noise statistics at the output of the envelope detector, for SNR's above approximately 4 dB in 100 Hz. Finally the performance of the HKM tree decoder against simulated hand-keyed Morse [Fig. 17] shows that, under these laboratory conditions, the tree decoder can be expected to provide an error rate no worse than that of a human transcriber for: (1) output copy with an acceptable error of 10% or less; (2) independent equally-likely letter messages. In comparison with the MAUDE algorithm, [Table XIX] the tree decoder shows a significant decrease in error rate on the simulated data, while in comparison with Howe's Quasi-Bayes decoder the error rates are about the same.

These results show that for the case of random letter text, the performance of a human operator can be very nearly obtained by optimal non-linear processing techniques. The

estimation algorithm derived in this investigation is adaptive to speed changes, varying noise levels and fading signals and has performed for approximately 90 hours of running time (approximately 21,000 characters total) without exhibiting any noticable signs of divergence or instability. The computational burden is severe, however, and for practical use would require possibly a pipe-lined approach with digital hardware under microprocessor control.

The strength of the tree decoder for random letters lies primarily in its use of the Morse code structure to perform channel decoding, i.e., demodulation, and secondarily in its use of the structure to accomplish source decoding. For contextual messages, however, a well-constructed model of the linguistics, semantics, ad format embodied in the structure of an appropriate f_{λ} text function, describing the evolution of the message states as a finite state machine, would add significantly to the error-correction capability of the decoder. To the extent that such a function can accurately describe the Morse message linguistically, the error-rate for contextual messages may be made to approach that for the human operator. As such, the parallel between the problems of Morse translation and automatic speech understanding is evident and therein lies the rub, and perhaps, the solution.

APPENDIX

SAMPLES OF OUTPUT DATA

I. In order to obtain an intuitive appeal for the errors produced by the tree decoder, several examples of output copy are shown below for various levels of keying quality and signal-to-noise ratios. Errors are indicated by an underline.

A. 50 wpm, KAM, 12 dB SNR:

A LAZY BROWN DOG JUMPED OVER 2 LOGS
ON A SUNNY SUNDAY AFTERNOON

B. 20 wpm, Fair Key, 9 dB SNR:

A LAZY BROWN DOG JU.ED OVF 2 LOGS
ON I SUNNY SUNDAY AMTERNOON

C. 20 wpm, Fair Key, 6 dB SNR:

A LS7 BORWN DOZ JUMPED JHF 2 LOGS
ON A SUNNY SUDDAS AFDRNOON

D. 20 wpm, Fair Key, 6 dB SNR (same as C., but with a different noise sequence):

A LSZY BROWN DOZ JUMPED OVEL 2 LOGS
ON A SUNNY IUTSANO AFTEGNOON

E. 20 wpm, Fair Key, 4 dB SNR

V LAZX HROWN DUD JUMPED JVEL IMI
L_OGS ON A SUNNY IM6ACN AFORNOON

F. 15 wpm, KAM, 12 dB SNR

CWA6 DE LAB IAW THE QUICK GREY FOX
JUMPED OVER THE LAZY BROWN DOG ON A
SUNNY SUMMER AFTERNOON. THIS IS A
TEST. VVV JVXI JGBA GBEY IQNH
OPRP CIPU URUC RHIC MUJX SKYQ

G. 15 wpm, Fair Key, 12 dB SNR

CWA6 DE HHH IAW THE QUICK GREY FOX
JUMPL OVER THE LAZY BROWN NR0GON
ASUNNY SUMMER AFTERNGON. 6IS IS A
NSCK VVV JVXI JGBA GBEY IHIH
OPRP CIPU UKUC RMIC MUJX SKYQ

H. 15 wpm, Fair Key, 6 dB SNR

C%A6 DE 5HH IAW 5E QUICO GREY FOX
JUMPED OHER T5 LAZY B5OW5 NR0G QN
ASUNNY SUMMER AFTERNOON 65IS A
NSCK VVV JVXI JGBA GBE3SHIH OPRAS
CIPU SKUC RHIC MUJX SKYQ

II. The waveforms shown in the following Figures (Fig. 18) are provided to give a visual appeal to the quality of the signals processed by the tree decoder. In each figure the input Morse keying signal is on line a. Immediately underneath, on line b is the output of the envelope detector after the carrier has been modulated by the keying signal, additive noise applied, filtered and finally detected. On line c is the detected signal, after downsampling to 200 Hz and adjusted in level by subroutine NOISE. The output of the zero-delay MAP estimate of the keystate (the demodulated signal) is on line d. These waveforms are the result of processing message E. above. Note that although the demodulated output in many cases is not correct, the correct letter is still decoded, because of the soft decisions utilized in the tree-decoder.

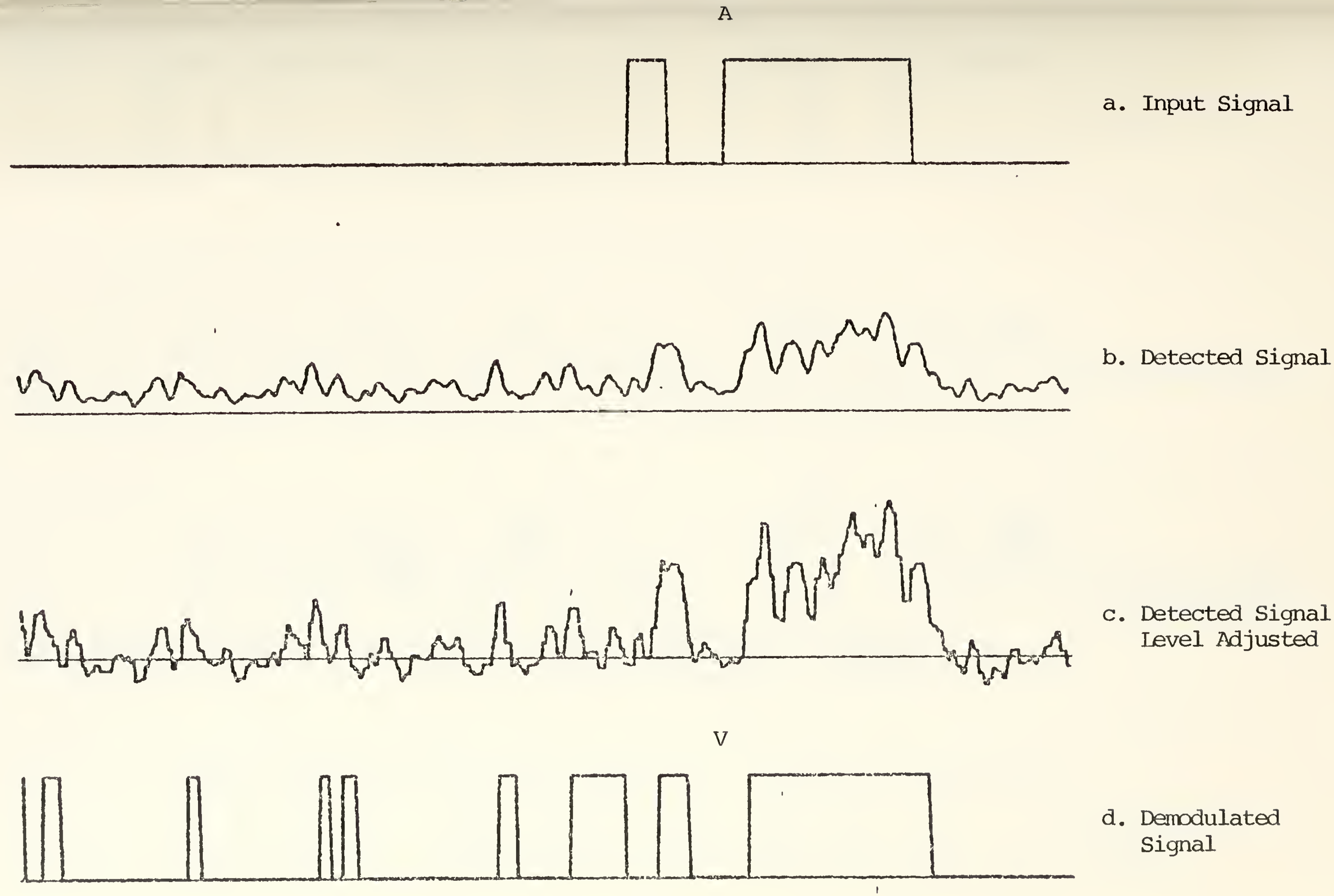

FIGURE 18a. Output Waveforms

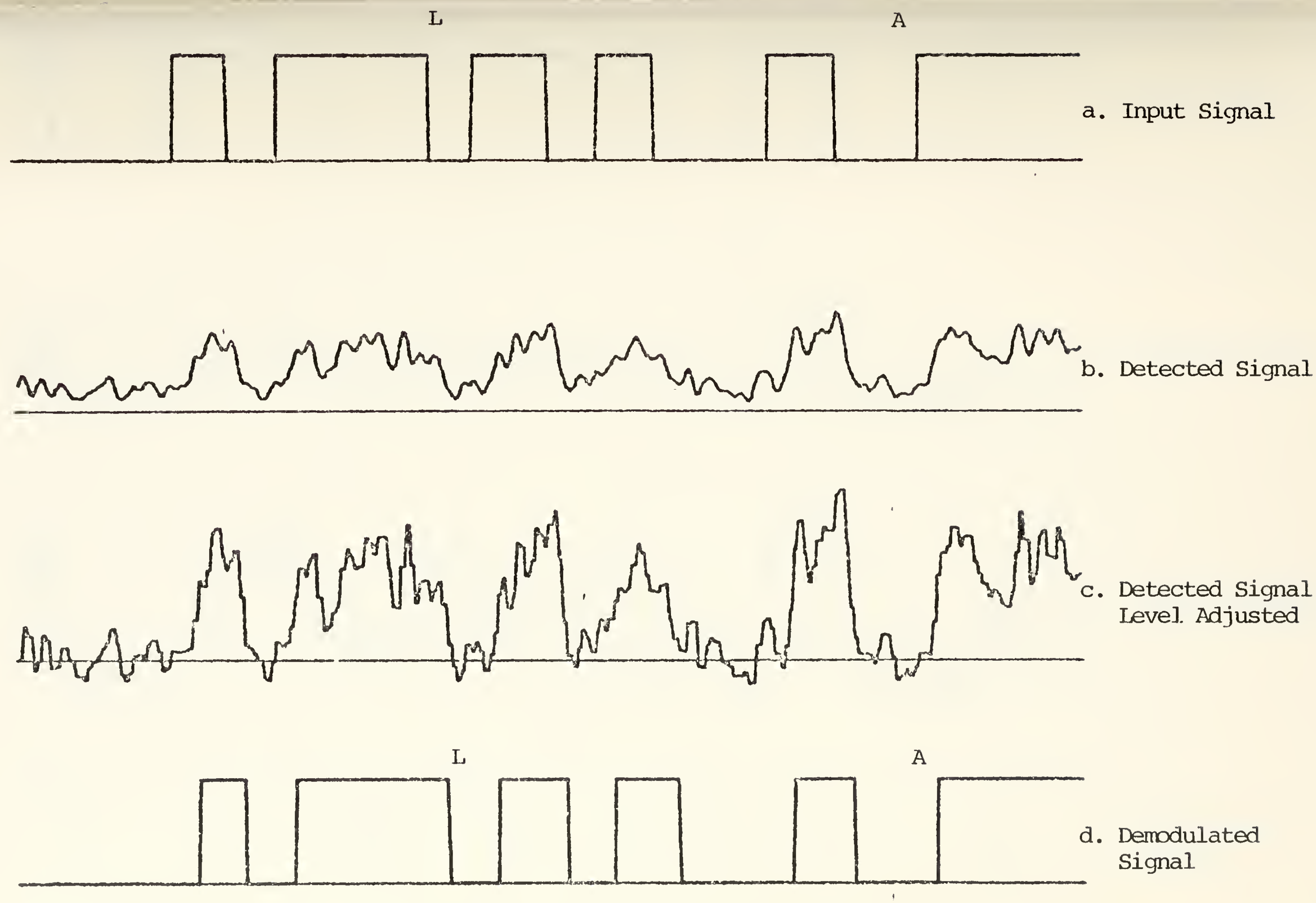

FIGURE 18b. Output Waveforms

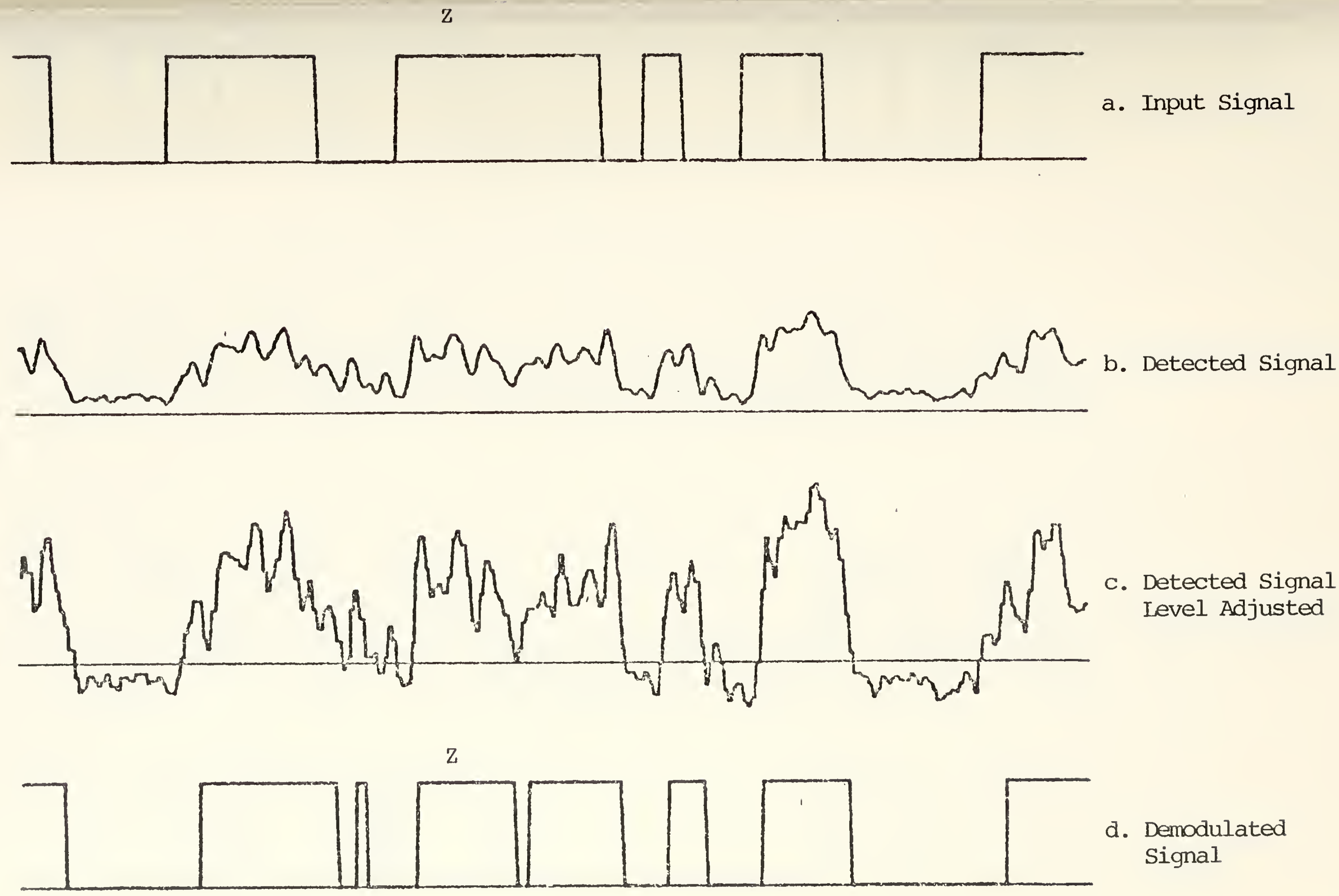

FIGURE 18c. Output Waveforms

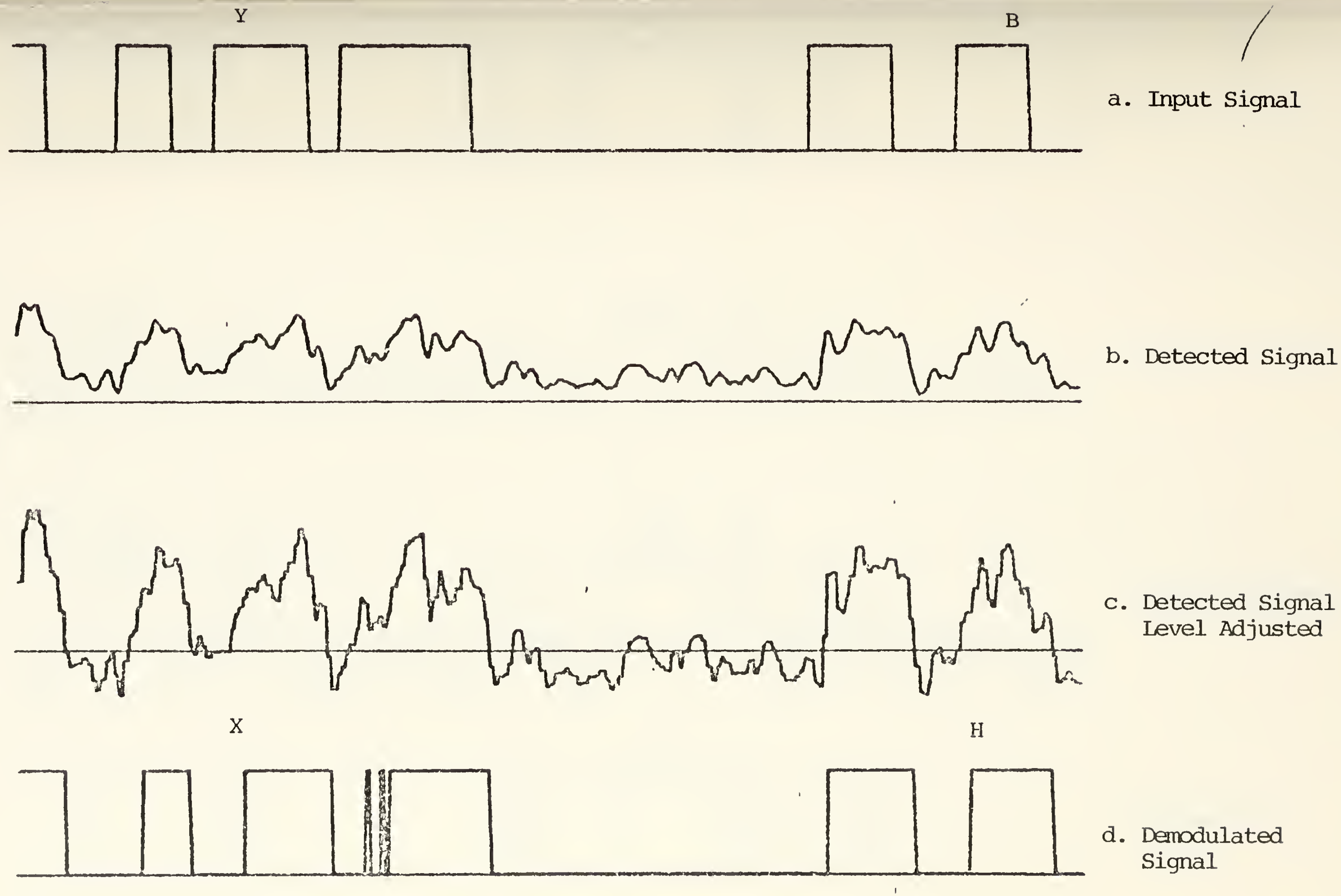

FIGURE 18d. Output Waveforms

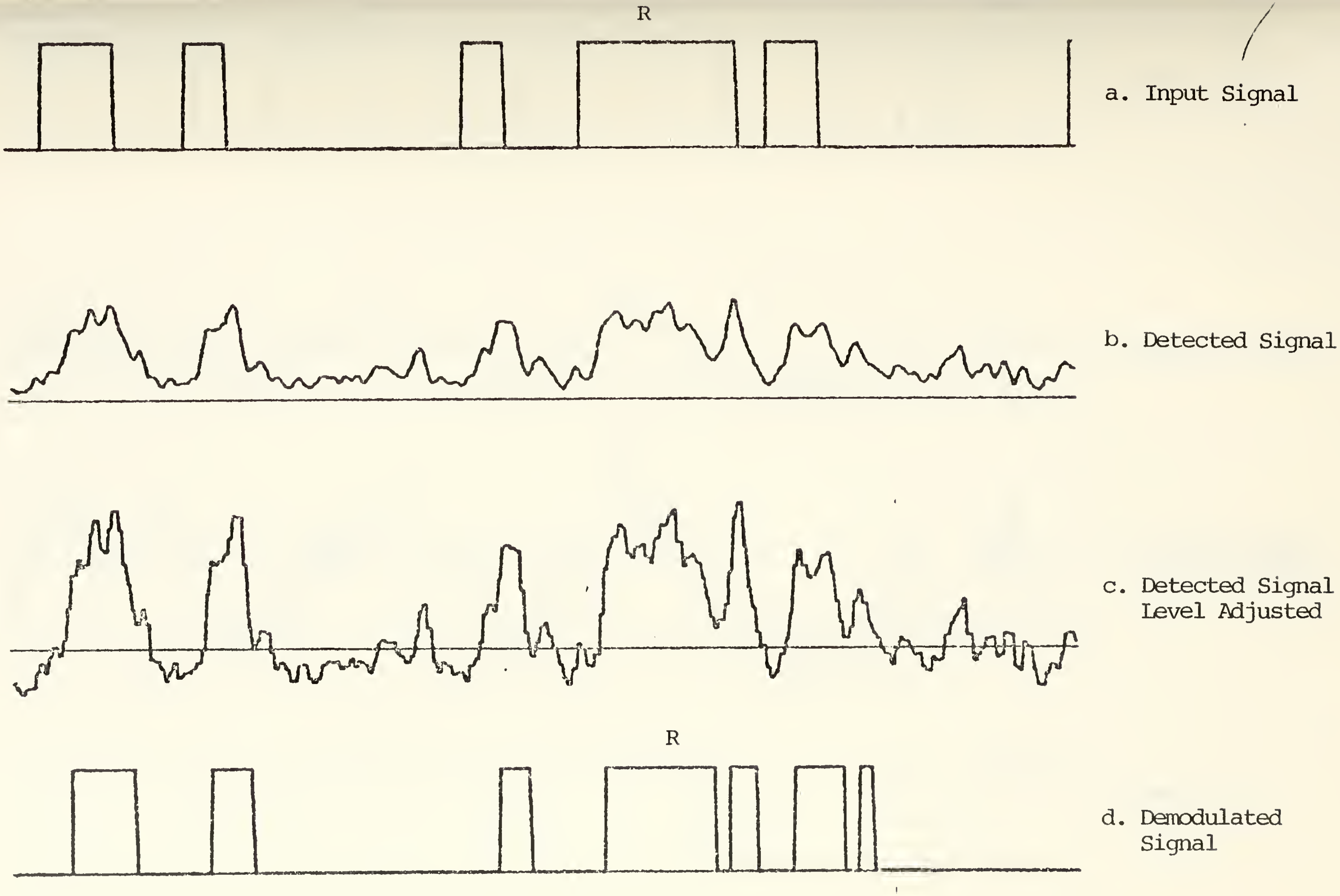

FIGURE 18e. Output Waveforms

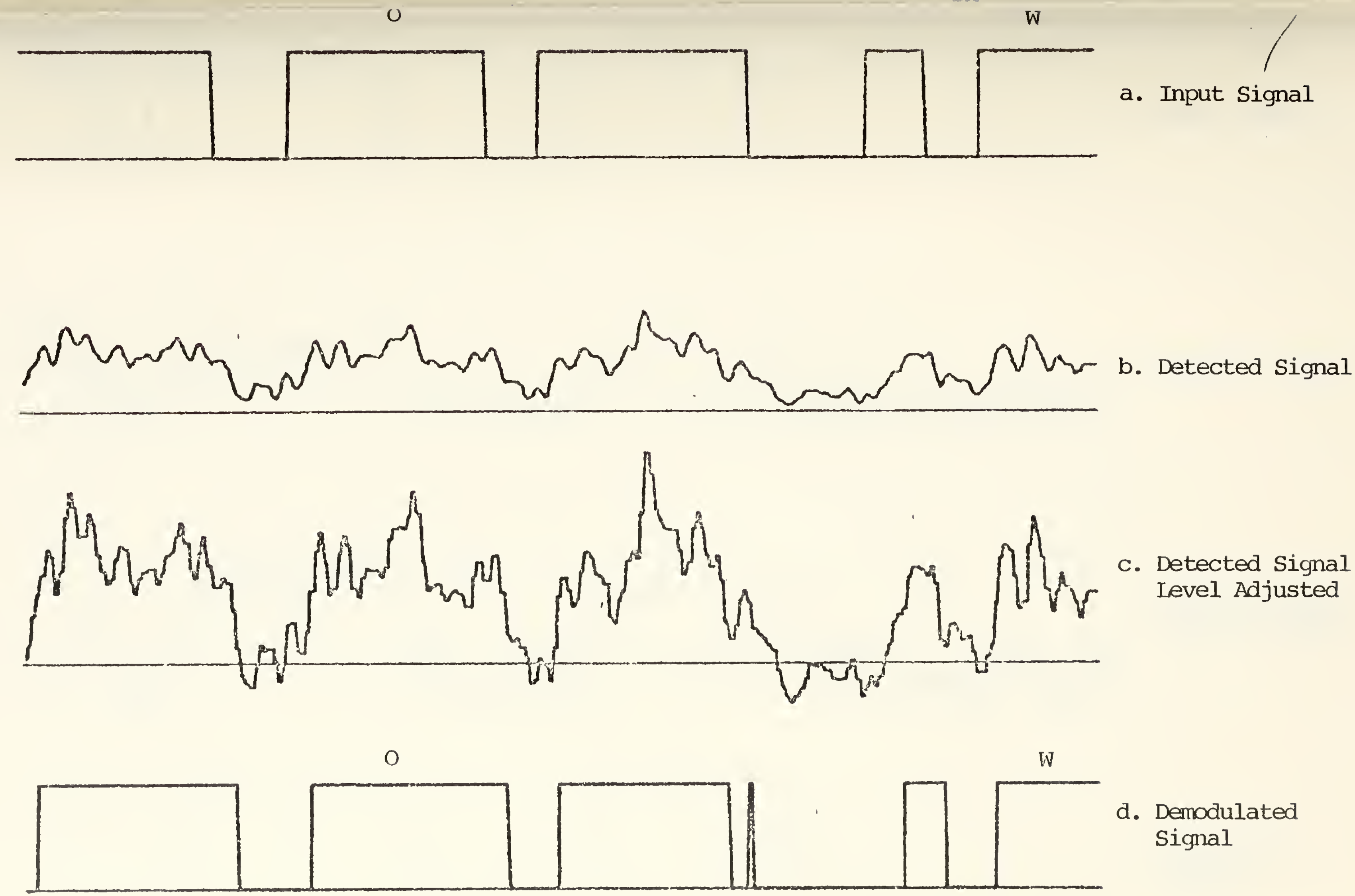

FIGURE 18f. Output Waveforms

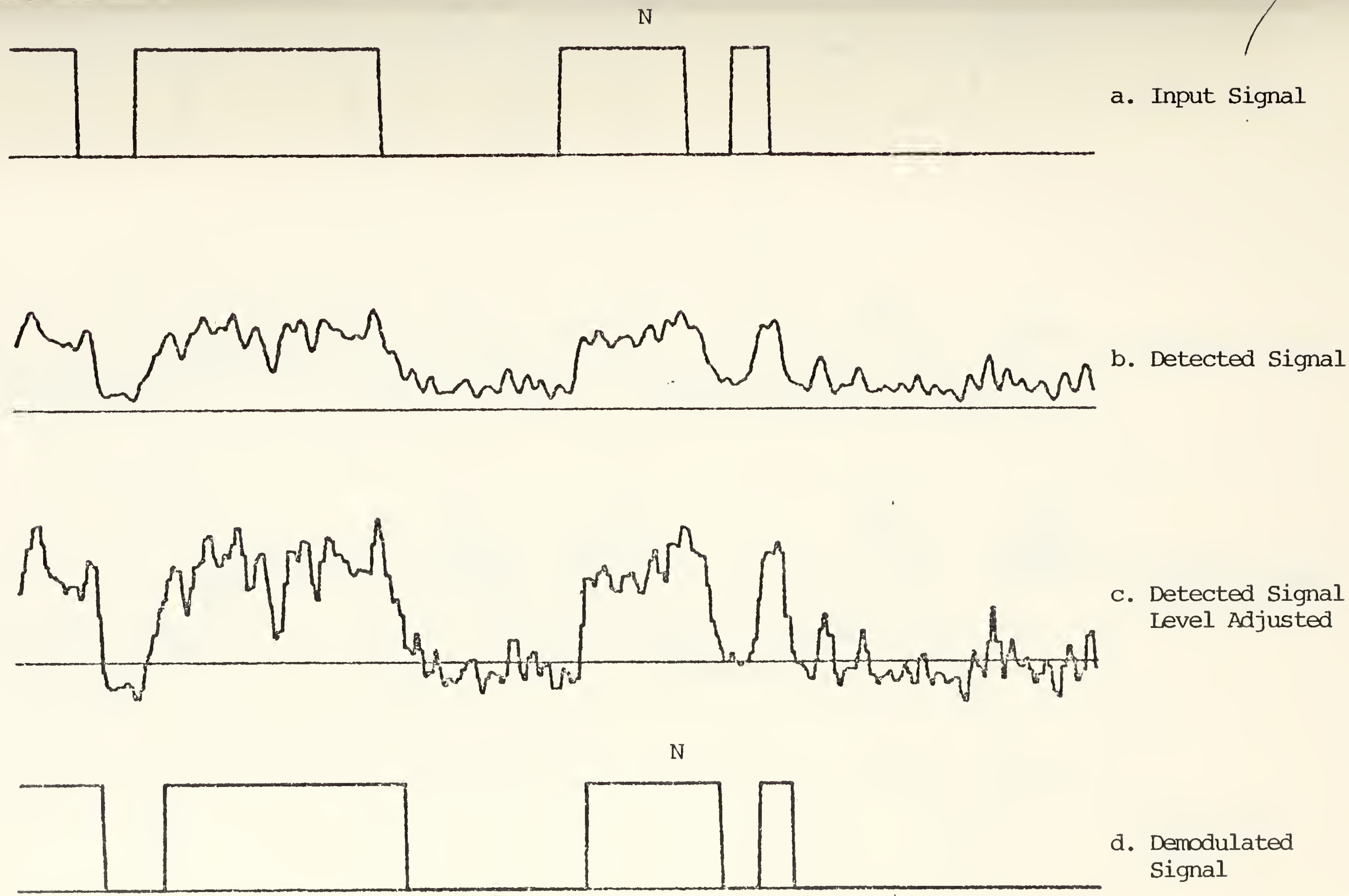

FIGURE 18g. Output Waveforms

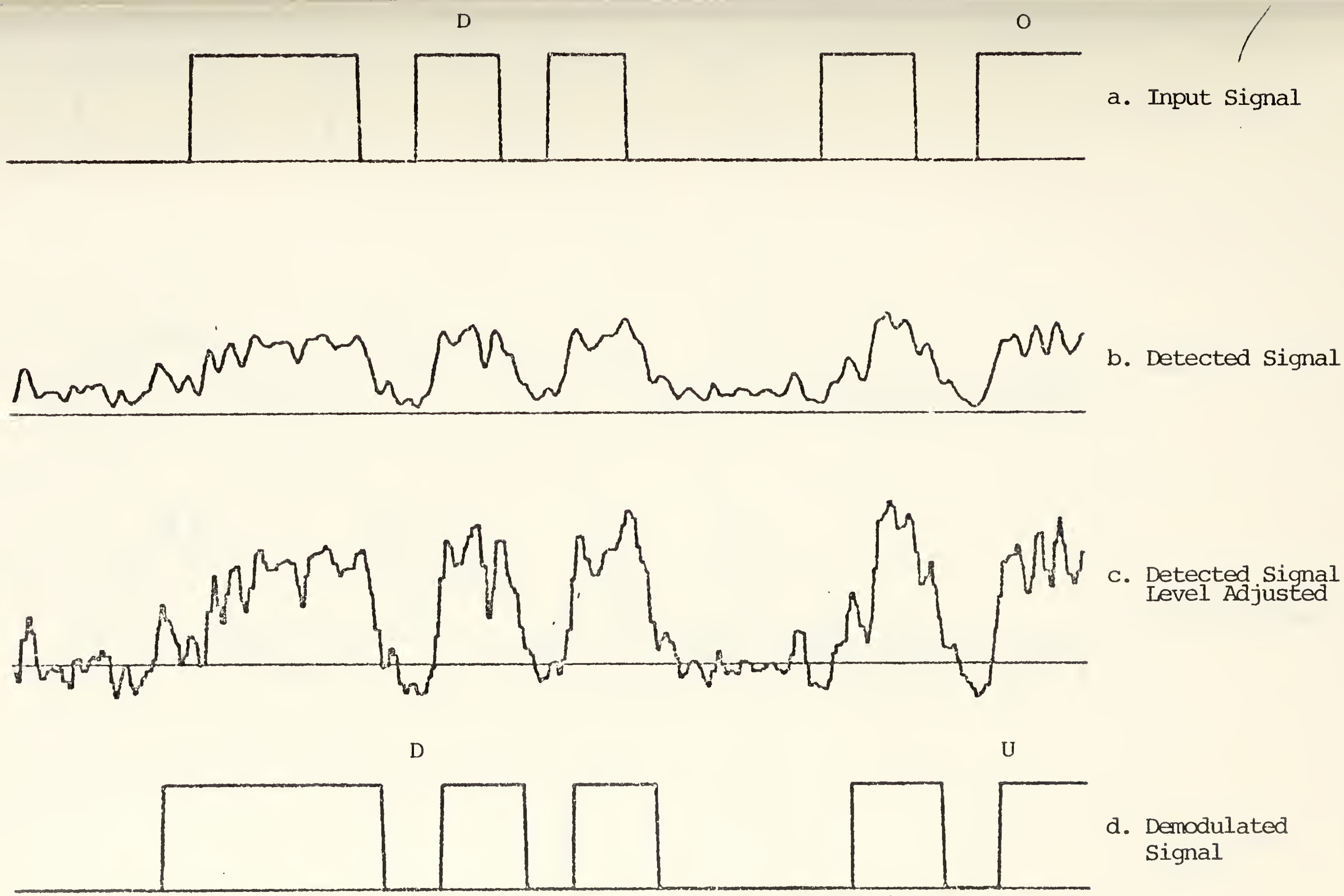

FIGURE 18h. Output Waveforms

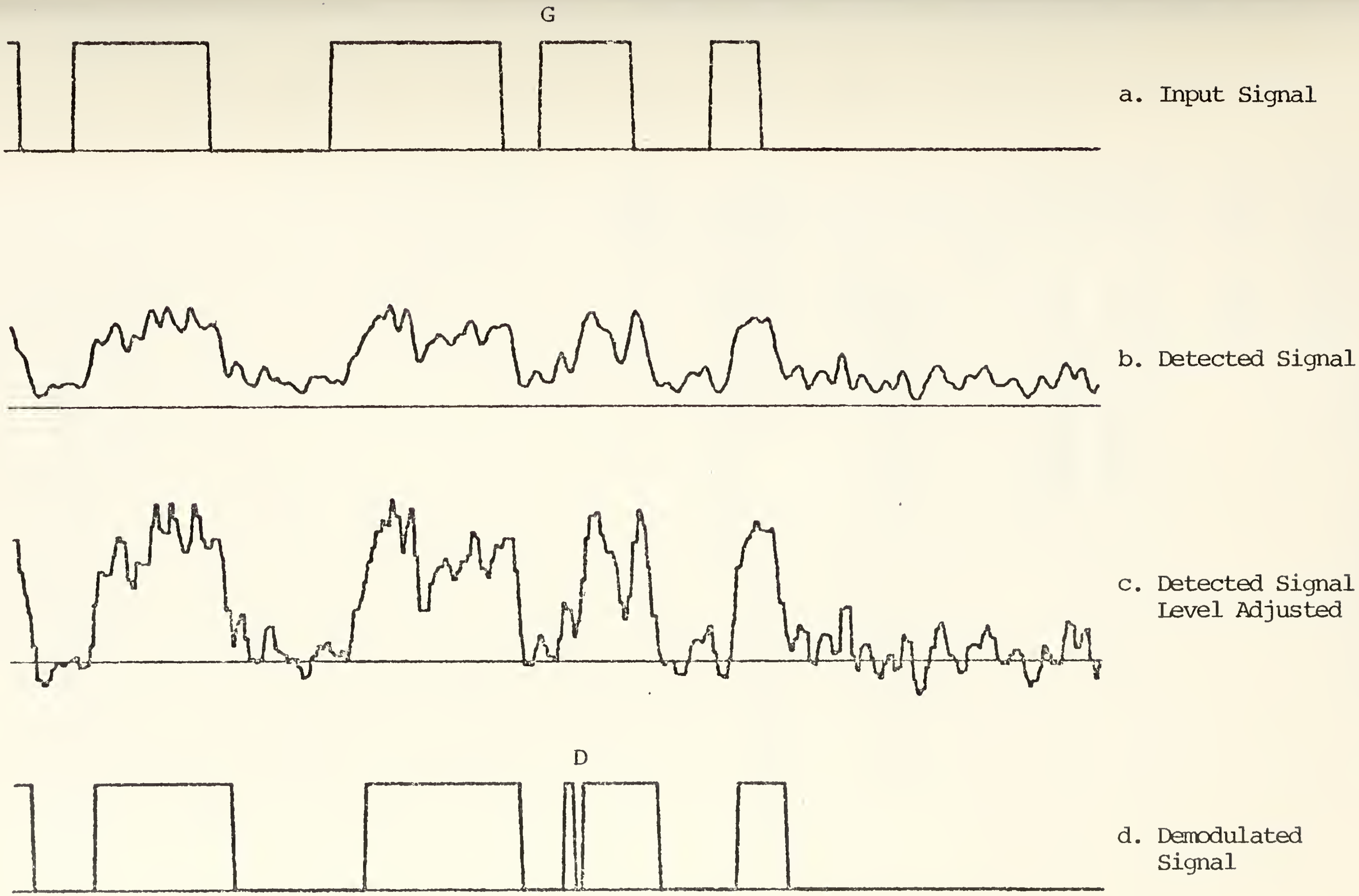

FIGURE 18i. Output Waveforms

```
      INTEGER ELMHAT,XHAT
      DIMENSION S1(512),S2(512),S3(512)
      DIMENSION S4(512)
      DATA RN/.1/
      DATA NP/0/

      CALL INITL
      CALL INPUTL

 1    DO 2 N1=1,512
      DO 3 N2=1,18
      CALL SIMSG1(X,ZSIG)

      CALL RCVR(ZSIG,ZRCV)
      CALL BPFDET(ZRCV,ZDET)

      NP=NP+1
      IF(NP.LT.40) GO TO 3
      NP=0
      CALL NOISE(ZDET,RN,Z)
      CALL PROCES(Z,RN,XHAT,PX,ELMHAT,LTRHAT)
 3    CONTINUE

      N=N1
      CALL STATS(ZDET,Z    ,PX   ,XHAT,S1,S2,S3,S4,N)
 2    CONTINUE
      CALL DISPLA(S1,S2,S3,S4)

      GO TO 1
      STOP
      END
```

```
      SUBROUTINE INPUTL
      DIMENSION ESEP(6),EDEV(6)
      COMMON/BLK1/TAU/BLK6/DMEAN,XDUR,ESEP,EDEV
      COMMON/BLK2/WC,WCHIRP,ASIGMA,BSIGMA,PHISGM,
     2RSIGM,TCHIRP,GAMMA
      DATA TAU/.000125/,ESEP/1,3,1,3,7,14/,EDEV/6*0./
      DATA XDUR/0./


      TYPE 100
100   FORMAT(1X,'INPUT KEYING PARMS: RATE,MEAN ELEM DURATIONS')
      ACCEPT 200,RATE,(ESEP(K),K=1,6)
      TYPE 150
150   FORMAT(1X,'INPUT ELEM DURATION STD DEVIATIONS')
      ACCEPT 200,(EDEV(K),K=1,6)
200   FORMAT(7F)
      TYPE 300
300   FORMAT(1X,'INPUT SIG PARMS- AVAR,BVAR,FCHIRP,TCHIRP,PHIVAR')
      ACCEPT 200,AVAR,BVAR,FCHIRP,TCHIRP,PHIVAR
      TYPE 400
400   FORMAT(1X,'INPUT SIG PARMS: GAMMA,FREQ,NOISE')
      ACCEPT 200,GAMMA,FC,RNOISE

      ASIGMA=SQRT(AVAR)
      BSIGMA=SQRT(BVAR)
      PHISGM=SQRT(PHIVAR)
      RSIGM=SQRT(RNOISE)

      DMEAN=1200./RATE
      WC=6.28319*FC
      WCHIRP=6.28319*FCHIRP


      IF(ESEP(1).NE.0.) GO TO 500
      ESEP(1)=1.
      ESEP(2)=3.
      ESEP(3)=1.
      ESEP(4)=3.
      ESEP(5)=7.
      ESEP(6)=14.


500   RETURN
      END




       SUBROUTINE INITL
       DIMENSION IELMST(400),ILAM1(16),ILAMX(6)
       DIMENSION ELEMTR(16,6),RTRANS(5,2),ISX(6)
       DIMENSION MEMFCN(400,6),LTRMAP(400),IALPH(70)
       DIMENSION MEMDEL(6,6),MEMPR(6,6),IBLANK(400)
       DIMENSION IARRAY(8),ITEXT(200)

       COMMON/BLKLAM/IELMST,ILAM1,ILAMX
       COMMON/BLKRAT/MEMDEL
       COMMON/BLKELM/ELEMTR/BLKSPD/RTRANS,MEMPR
       COMMON/BLKMEM/MEMFCN/BLKS/ISX
       COMMON/BLKTRN/LTRMAP,IALPH,IBLANK
```

```
      COMMON/BLKTXT/ITEXT

      DATA ISX/1,1,0,0,0,0/
      DATA MEMFCN/9,11,13,15,9,11,13,15,9,0,11,0,13,0,15,0,
     2 384*0,
     2 10,12,14,16,10,12,14,16,0,10,0,12,0,14,0,16,384*0,
     2 1,0,0,0,5,0,0,0,1,5,1,5,1,5,1,5,384*0,
     2 0,2,0,0,0,6,0,0,2,6,2,6,2,6,2,6,384*0,
     2 0,0,3,0,0,0,7,0,3,7,3,7,3,7,3,7,384*0,
     2 0,0,0,4,0,0,0,8,4,8,4,8,4,8,4,8,384*0/

      DATA IELMST/1,2,3,4,5,6,7,8,9,10,11,12,
     2 13,14,15,16,384*0/
      DATA ILAM1/3,4,5,6,3,4,5,6,1,2,1,2,1,2,1,2/
      DATA ILAMX/1,1,0,0,0,0/

      DATA LTRMAP/3,4,5,6,3,4,5,6,1,2,1,2,1,2,1,2,384*0/
      DATA IALPH/'A','B','C','D','E','F','G','H','I',
     2 'J','K','L','M','N','O','P','Q','R','S','T','U',
     2 'V','W','X','Y','Z','1','2','3','4','5','6','7',
     2 '8','9','0',';',':','%','&',0,0,'K',',','AS','SN',
     2 0,0,0,0,'VR','NO','GA','OK','AR','SK',0,0,0,0,
     2 'IMI',0,0,0,0,'BT',0,0,0,'EEE'/
      DATA IBLANK/400*0/


      DATA ELEMTR/.55,.5,.5,.5,.55,.5,.5,.5,8*0.,
     2 .45,.5,.5,.5,.45,.5,.5,.5,8*0.,
     2 8*0.,.581,.54,.923,.923,.923,.923,.95,.95,
     2 8*0.,.335,.376,.062,.062,.062,.062,.04,.04,
     2 8*0.,.067,.069,.012,.012,.012,.012,.009,.009,
     2 8*0.,.015,.015,.003,.003,.003,.003,.001,.001/


      DATA RTRANS/.1,.2,.4,.2,.1,.15,.2,.3,.2,.15/
      DATA MEMDEL/0,0,2,2,5,10,0,0,2,2,5,10,
     2 2,2,0,0,0,0,2,2,0,0,0,0,2,2,0,0,0,0,
     2 2,2,0,0,0,0/
      DATA MEMPR/0,0,1,2,1,2,0,0,1,2,1,2,1,1,0,0,0,0,
     2 1,1,0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,0/


      OPEN(UNIT=20,FILE='MORSEM')
      DO 10 I=1,300
      READ(20,30) (IARRAY(K),K=1,8)
 30   FORMAT(8I3)
      DO 11 K=1,6
 11   MEMFCN(I,K)=IARRAY(K+2)
      LTRMAP(I)=IARRAY(1)
      IELMST(I)=IARRAY(2)
      IF((IELMST(I).EQ.7).OR.(IELMST(I).EQ.3))
     2 IBLANK(I)=1
      IF((IELMST(I).EQ.8).OR.(IELMST(I).EQ.4))
     2 IBLANK(I)=2
 10   CONTINUE

      ENDFILE 20
      OPEN(UNIT=20,FILE='OUTPUT')
      DO 50 I=1,300
      WRITE(20,40) (MEMFCN(I,K),K=1,6)
```

```
   40   FORMAT(10X,6(I3,2X))

  50    CONTINUE
        ENDFILE 20

       OPEN(UNIT=20,FILE='TEXT')
       DO 60 I=1,105
       READ(20,70) ITEXT(I)
  70   FORMAT(I2)
  60   CONTINUE
       ENDFILE 20

       RETURN
       END
```

```
      SUBROUTINE SIMSG1(X,SIG)

      COMMON/BLK1/TAU
      COMMON/BLK2/WC,WCHIRP,ASIGMA,BSIGMA,PHISGM,
     2RSIGM,TCHIRP,GAMMA
      DATA XLAST/1./,BETA/1./
      DATA AMP/1./,BFADE/0./,THETA/0./,PHI/0./

      DUR=BETA
      CALL KEY(DUR,X)
      BETA=BETA*(1.-X-XLAST+2.*X*XLAST)+1.
      TK=X*(1.-XLAST)
      XLAST=X

      CALL RANDN(W,1,0.,ASIGMA)
      AMP=AMP+TK*W
      IF(AMP.LT..01) AMP=.01

      CALL RANDN(W,1,0.,BSIGMA)
      BFADE=GAMMA*BFADE+W

      AMPB=AMP+BFADE
      IF(AMPB.LT.0.001) BFADE=0.001-AMP
      AMPB=AMP+BFADE
      TDUR=1000.*TAU*BETA
      WCHRP=X*WCHIRP*EXP(-TDUR/TCHIRP)
      THETA=THETA+(WC+WCHRP)*TAU
      THETA=AMOD(THETA,6.28319)

      CALL RANDN(W,1,0.,PHISGM)
      PHI=PHI+TK*W
      PHI=AMOD(PHI,6.28319)

      SIG=X*AMPB*SIN(THETA+PHI)

      CALL RANDN(ZN,1,0.,RSIGM)
      SIG=SIG+ZN



      RETURN
      END

        SUBROUTINE KEY(DUR,X)
        DIMENSION ESEP(6),EDEV(6),MORSE(10,40)
        DIMENSION IOUT(500),ISYMBL(6),ITEXT(200)
        COMMON/BLKEND/IEND
        COMMON/BLK1/TAU/BLK6/DMEAN,XDUR,ESEP,EDEV
        COMMON/BLKTXT/ITEXT
        DATA IK/"001000000200/
        DATA LTR/20/,NELM/0/,N/0/,NLTR/1/
        DATA MORSE/1,3,2,0,0,0,0,0,0,0,
     2  2,3,1,3,1,3,1,0,0,0,2,3,1,3,1,3,1,0,0,0,
     2  2,3,1,3,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,
     2  1,3,1,3,2,3,1,0,0,0,2,3,2,3,1,0,0,0,0,0,
     2  1,3,1,3,1,3,1,0,0,0,1,3,1,0,0,0,0,0,0,0,
     2  1,3,2,3,2,3,2,0,0,0,2,3,1,3,2,0,0,0,0,0,
     2  1,3,2,3,1,3,1,0,0,0,2,3,2,0,0,0,0,0,0,0,
     2  2,3,1,0,0,0,0,0,0,0,2,3,2,3,2,0,0,0,0,0,
     2  1,3,2,3,2,3,1,0,0,0,2,3,2,3,1,3,2,0,0,0,
```

```
     2  1,3,2,3,1,0,0,0,0,0,1,3,1,3,1,0,0,0,0,0,
     2  2,0,0,0,0,0,0,0,0,0,1,3,1,3,2,0,0,0,0,0,
     2  1,3,1,3,1,3,2,0,0,0,1,3,2,3,2,0,0,0,0,0,
     2  2,3,1,3,1,3,2,0,0,0,2,3,1,3,2,3,2,0,0,0,
     2  2,3,2,3,1,3,1,0,0,0,1,3,2,3,2,3,2,3,2,0,
     2  1,3,1,3,2,3,2,3,2,0,1,3,1,3,1,3,2,3,2,0,
     2  1,3,1,3,1,3,1,3,2,0,1,3,1,3,1,3,1,3,1,0,
     2  2,3,1,3,1,3,1,3,1,0,2,3,2,3,1,3,1,3,1,0,
     2  2,3,2,3,2,3,1,3,1,0,2,3,2,3,2,3,2,3,1,0,
     2  2,3,2,3,2,3,2,3,2,0,40*0/
        DATA ISYMBL/1H.,1H_,1H ,1H/,1H!,1H\/

        BETA=1000.*TAU*DUR
        IF(BETA.LT.XDUR) GO TO 200
        NELM=NELM+1
        IELM=MORSE(NELM,LTR)
        IF(IELM.NE.0) GO TO 100
        NELM=0
        Y=RAN(IK)
        IELM=4
        IF(Y.GT..9) IELM=5
        IF((Y.LE..9).AND.(Y.GT..3)) IELM=6
        Y=RAN(IK)
        Y=35*(Y-.001)+1.
        IY=Y
        LTR=IY+1

        GO TO 100
        NLTR=NLTR+1
        LTR=ITEXT(NLTR)
        IF(LTR.EQ.0) IELM=4
        IF(LTR.EQ.37) IELM=5
        IF(LTR.EQ.38) IELM=6
        NLTR=NLTR+1
        LTR=ITEXT(NLTR)

 100    N=N+1
        IOUT(N)=ISYMBL(IELM)
        IF(N.LT.300) GO TO 110
        N=0
        NLTR=0
        IEND=1
        TYPE 900
 900    FORMAT(/,/,1X,'> END OF RUN; INPUT DATA WAS:',/)
        DO 10 K=1,10
        K1=(K-1)*50+1
        K2=K*50
        TYPE 1000,(IOUT(L),L=K1,K2)
 1000   FORMAT(/,1X,50A1)
 10     CONTINUE
        ACCEPT 1000,WAIT

 110    XM=ESEP(IELM)*DMEAN
        XSIGM=EDEV(IELM)*DMEAN
        Y=RAN(IK)
        Y=2.*(Y-.5)
        XDUR=XM+Y*XSIGM
        IF(XDUR.LT.20.) XDUR=20.
        X=1.
        IF(IELM.GE.3) X=0.
```

```
      200 RETURN
          END
```

```
      SUBROUTINE DISPLA(S1,S2,S3,S4)
      DIMENSION S1(512),S2(512),S3(512),S4(512)
      CALL ERASE
      CALL PLOTR(S1,512,0,XM,400)
      CALL PLOTR(S2,512,0,XM,275)
      CALL PLOTR(S3,512,1,1.,150)
      CALL PLOTR(S4,512,0,XM,40)
      CALL VIEW('1')
        ACCEPT 1000,WAIT
 1000 FORMAT(A5)
      RETURN
      END

        SUBROUTINE STATS(XIN1,XIN2,XIN3,XIN4,S1,
     2 S2,S3,S4,N)
        DIMENSION S1(512),S2(512),S3(512),S4(512)
        S1(N)=XIN1
        S2(N)=XIN2
        S3(N)=XIN3
        S4(N)=XIN4
        RETURN
        END

        SUBROUTINE AUTOCR(S5,RS)
        DIMENSION S5(512),RS(512),S(1000),RS1(500)
        DATA S/1000*0./,XN/0./
        XN=XN+1
        DO 100 I=1,500
        S(I)=S5(I)
        RS1(I)=0.
 100    CONTINUE
        DO 200 I=1,500
        DO 300 K=1,500
        RS1(I)=RS1(I)+S(K+I-1)*S(K)
 300    CONTINUE
 200    CONTINUE
        DO 400 I=1,500
        RS(I)=(RS(I)*(XN-1.)+RS1(I))/XN
 400    CONTINUE
        RETURN
        END
```

```
      SUBROUTINE PROCES(Z,RN,XHAT,PX,ELMHAT,LTRHAT)

C*********************************************************************
C
C   THIS SUBROUTINE IMPLEMENTS THE PROCESSING ALGORITHM
C   FOR JOINT DEMODULATION,DECODING,AND TRANSLATION OF
C   THE RECEIVED MORSE PROCESS. IT TAKES IN A NEW MEASURE-
C   MENT,Z,OF THE DETECTED SIGNAL EVERY 5 MSEC AND PRO-
C   DUCES AN ESTIMATE OF THE CURRENT KEYSTATE,ELEMENT
C   STATE,AND LETTER OF THE RECEIVED SIGNAL.
C
C   DEFINITIONS OF VARIABLE NAMES:
C       Z-          INPUT SAMPLE OF DETECTED SIGNAL
C       RN-         INPUT NOISE POWER ESTIMATE
C       XHAT-       OUTPUT ESTIMATE OF KEYSTATE
C       ELMHAT-     OUTPUT ESTIMATE OF ELEMENT STATE
C       LTRHAT-     OUTPUT ESTIMATE OF LETTER STATE
C
C       ISAVE-      NO. OF PREVIOUS PATHS SAVED
C       IPATH-      IDENTITY OF SAVED PATH
C       LAMBDA(I)-IDENTITY OF LTR STATE OF SAVED PATH I
C       DUR(I)-     DURATION OF ELEMENT ON PATH I
C       ILRATE(I)-IDENTITY OF DATA RATE ON PATH I
C       PIN(I,N)- COMPUTED TRANS PROB FROM PATH I TO STATE N
C       LAMSAV(J)-IDENTITY OF LTR STATE AT NEW NODE J
C       ILRSAV(J)-IDENTITY OF DATA RATE AT NEW NODE J
C       LKHD(J)-  LIKELIHOOD VALUE FOR NODE J
C       P(J)-       COMPUTED POSTERIOR PROB OF PATH
C                   ENDING AT NEW NODE J
C       PSELEM(K)-COMPUTED POSTERIOR PROB OF ELEM K
C       SPDHAT    -COND MEAN ESTIMATE OF INSTANT DATA RATE
C       PX-         POSTERIOR PROB THAT KEYSTATE EQUALS 1
C
C   THE FOLLOWING SUBROUTINES ARE UTILIZED:
C       TRPROB- COMPUTES TRANSITION PROBABILITIES
C       PATH-   COMPUTES IDENTITY OF NEW PATHS
C       LIKHD-  COMPUTES THE LIKELIHOOD OF EACH PATH EXTENSION
C       PROBP-  COMPUTES POSTERIOR PROBS OF EACH NEW PATH
C       SPROB-  COMPUTES POSTERIOR PROBS OF EACH STATE
C       SAVE-   SAVES THE HIGHEST PROB PATHS
C       TRELIS- FORMS A TRELIS OF SAVED PATHS
C       TRANSL- TRANSLATES THE LETTER ESTIMATE
C
C   ALL TABLES OF CONSTANTS ARE STORED IN COMMON.
C
C**********************************************************************

      REAL LKHD
      INTEGER ELMHAT,XHAT,PATHSV,SORT
      DIMENSION LAMBDA(25),DUR(25),ILRATE(25),PIN(25,30)
      DIMENSION LAMSAV( 750),DURSAV( 750),ILRSAV( 750)
      DIMENSION LKHD(750),P(750),PSELEM(6)
      DIMENSION PATHSV(25),SORT(25)

      DATA ISAVE/25/
      DATA LAMBDA/25*5/
      DATA ILRATE/5*10,5*20,5*30,5*40,5*50/
      DATA P/750*1./

      DATA LAMSAV/750*5/,DUR/25*1000./
```

```
      DATA ILRSAV/750*20/,PATHSV/25*5/


C   FOR EACH SAVED PATH, COMPUTE:
C     TRANSITION PROBABILITY TO NEW STATE (TRPROB);
C     IDENTITY OF EACH NEW PATH EXTENDED (PATH);
C     LIKELIHOOD OF EACH STATE EXTENSION (LIKHD):
C
C
      DO 100 I=1,ISAVE
      IPATH=I

      CALL TRPROB(IPATH,LAMBDA(I),DUR(I),ILRATE(I),PIN)
      CALL PATH(IPATH,LAMBDA(I),DUR(I),ILRATE(I),LAMSAV,DURSAV,ILR
      CALL LIKHD(Z,RN,IPATH,LAMBDA(I),DUR(I),
     2 ILRATE(I),PIN,LKHD)

 100  CONTINUE

C   HAVING OBTAINED ALL NEW PATHS, COMPUTE:
C     POSTERIOR PROBABILITY OF EACH NEW PATH (PROBP);
C     POSTERIOR PROBABILITY OF KEYSATE,ELEM STATE,
C       CONDITIONAL MEAN ESTIMATE OF SPEED (SPROB);
C

      CALL PROBP(P,PIN,ISAVE,LKHD)
      CALL SPROB(P,ISAVE,ILRSAV,PELM,KHAT,
     2 SPDHAT,PX)

      XHAT=0
      IF(PX.GT.0.5) XHAT=1
C
C   SAVE THE PATHS WITH HIGHEST PROBABILITY,AND
C   STORE THE VALUES CORRESPONDING TO THESE PATHS:
C
      CALL SAVEP(P,PATHSV,ISAVE,IMAX,LAMSAV,DURSAV,
     2 ILRSAV,LAMBDA,DUR,ILRATE,SORT)
      GO TO 1
      TYPE 1000,Z
 1000 FORMAT(//,4X,F10.7,/)
      DO 1 IN=1,ISAVE
      TYPE 1100, IN,P(IN),PATHSV(IN),LAMBDA(IN),DUR(IN),ILRATE(IN)
     2 ,LKHD(SORT(IN))
 1100 FORMAT(1X,I3,2X,F10.7,2X,I3,2X,I3,2X,F6.1, 2X,I3,2X,F10.7)
 1    CONTINUE


C
C   UPDATE TRELLIS WITH NEW SAVED NODES,AND
C   OBTAIN LETTER STATE ESTIMATE:
C
      CALL TRELIS(ISAVE,PATHSV,LAMBDA,IMAX)

 200  RETURN
      END
```

```
      SUBROUTINE TRPROB(IP,LAMBDA,DUR,ILRATE,P)

C*********************************************************************
C
C   THIS SUBROUTINE COMPUTES THE TRANSITION PROBABILITY
C   FROM SAVED PATH IP TO EACH STATE N AND STORES THE
C   RESULT IN P(IP,N).
C
C   VARIABLES:
C       IP-     INPUT SAVED PATH IDENTITY
C       LAMBDA- INPUT SAVED LTR STATE IDENTITY
C       DUR-    INPUT SAVED ELEMENT DURATION
C       ILRATE- INPUT SAVED DATA RATE IDENTITY
C        P-      OUTPUT TRANSITION PROBABILITY MATRIX
C
C   THE FOLLOWING FUNCTION SUBROUTINES ARE USED:
C       XTRANS- RETURNS THE KEYSTATE TRANSITION PROBABILITY
C               CONDITIONED ON ELEMENT TYPE AND DATA RATE
C       PTRANS- RETURNS THE PATH-CONDITIONAL STATE TRANSITION PROB
C
C
C*********************************************************************

      DIMENSION P(25,30),IELMST(400),ILAM1(16),ILAMX(6)
      DIMENSION PIN(30)

      COMMON /BLKLAM/IELMST,ILAM1,ILAMX

C
C   LOOK UP ELEMENT TYPE FOR LTR STATE LAMBDA:
C
      IF(LAMBDA.NE.0) GO TO 20
      DO 10 N=1,30
      P(IP,N)=0.
 10   CONTINUE
      GO TO 200


 20   IELEM=ILAM1(IELMST(LAMBDA))
C
C   COMPUTE KEYSTATE TRANSITION PROBABILITY:
C
      PTRX=XTRANS(IELEM,DUR,ILRATE)
C
C   FOR EACH STATE, COMPUTE STATE TRANSITION PROBABILITY:
C
      PSUM=0.
      DO 100 K=1,6
      DO 100 I=1,5
      N=(I-1)*6+K
      KELM=K
      IRATE=I
      CALL PTRANS(KELM,IRATE,LAMBDA,ILRATE,PTRX,PSUM,PIN,N)
 100  CONTINUE

      DO 300 N=1,30
      P(IP,N)=PIN(N)/PSUM
```

```
  300   CONTINUE

  200   RETURN
        END


        FUNCTION XTRANS(IELEM,D0,IRATE)

C***********************************************************************
C
C   THIS FUNCTION IMPLEMENTS THE CALCULATION OF KEYSTATE
C   TRANSITION PROBABILITY, CONDITIONED ON ELEMENT TYPE,
C   CURRENT DURATION, AND DATA RATE.
C
C   VARIABLES:
C       IELEM- INPUT CURRENT ELEMENT TYPE
C       D0-    INPUT CURRENT ELEMENT DURATION
C       IRATE- INPUT CURRENT DATA RATE
C
C   TABLES IN COMMON CONTAIN DENSITY PARMS FOR EACH
C   ELEMENT TYPE, DATA RATE.
C
C***********************************************************************

        DIMENSION KIMAP(6),APARM(3)
        DATA KIMAP/1,3,1,3,7,14/
        DATA APARM/3.000,1.500,1.000/

C
C   SCALE DURATION AND OBTAIN DENSITY PARAMETER:
C
        MSCALE=KIMAP(IELEM)
        RSCALE=1200./IRATE
        B0=D0/(MSCALE*RSCALE)
        B1=(D0+5.)/(MSCALE*RSCALE)

        IF(IELEM.EQ.6) GO TO 20
        IF(IELEM.EQ.5) GO TO 10
        ALPHA=MSCALE*APARM(1)
        GO TO 100

  10    ALPHA=7.*APARM(2)
        GO TO 100

  20    ALPHA=14.*APARM(3)

  100   IF(B1.LE.1.) GO TO 200
        IF((B0.LT.1.).AND.(B1.GT.1.)) GO TO 300
        XTRANS=EXP(-ALPHA*(B1-B0))
        GO TO 400

  200   P1=1.-0.5*EXP(ALPHA*(B1-1.))
        P0=1.-0.5*EXP(ALPHA*(B0-1.))
        XTRANS=P1/P0
        GO TO 400

  300   P1=0.5*EXP(-ALPHA*(B1-1.))
        P0=1.-0.5*EXP(ALPHA*(B0-1.))
        XTRANS=P1/P0

```

```
  400   RETURN
        END





        SUBROUTINE PTRANS(KELEM,IRATE,LAMBDA,ILRATE,PTRX,
     2  PSUM,PIN,N)

C*****************************************************************
C
C   THIS FUNCTION SUBROUTINE RETURNS THE PATH CONDITIONAL
C   TRANSITION PROBABILITIES TO EACH ALLOWABLE STATE N.
C
C   VARIABLES:
C       KELEM-  INPUT CURRENT ELEMENT STATE
C       IRATE-  INPUT CURRENT DATA RATE STATE
C       LAMBDA- INPUT IDENTITY OF CURRENT LTR STATE
C       PTRX-   INPUT KEYSTATE TRANSITION PROBABILITY
C       ELEMTR- ELEMENT TRANSITION PROBABILITY MATRIX
C
C   FUNCTION SUBROUTINE USED:
C       SPDTR-  RETURNS DATA RATE TANSITION PROBS,
C               CONDITIONED ON CURRENT SPACE TYPE.
C
C*****************************************************************

        DIMENSION IELMST(400),ILAM1(16),ELEMTR(16,6)
        DIMENSION ILAMX(6),PIN(30)

        COMMON/BLKLAM/IELMST,ILAM1,ILAMX
        COMMON/BLKELM/ELEMTR

C   IF THE SAVED ELEMENT AND THE ELEMENT OF THE STATE
C    N TO WHICH THE PATH IS BEING EXTENDED ARE THE
C    SAME, THEN THE STATE TRANS PROB IS SIMPLY
C   KEYSTATE TRANS PROB:
C
        IF(KELEM.NE.ILAM1(IELMST(LAMBDA))) GO TO 100
        PIN(N)=PTRX
        IF(IRATE.NE.3) PIN(N)=0.
        GO TO 200

C
C   OTHERWISE:
C
C    OBTAIN ELEM TRANS PROBS FROM TABLE:
C
  100   PELEM=ELEMTR(IELMST(LAMBDA),KELEM)

C
C   NEXT COMPUTE  ELEM-CONDITIONAL SPEED TRANS PROB:
C
        PRATE=SPDTR(IRATE,ILRATE,KELEM,ILAM1(IELMST(LAMBDA)))

C
C   PTRANS IS THE PRODUCT:
C
        PIN(N)=(1.-PTRX)*PELEM*PRATE
  200   PSUM=PSUM+PIN(N)
```

```
      RETURN
      END




      FUNCTION SPDTR(ISRT,ILRT,ISELM,ILELM)

C***********************************************************************
C
C   THIS FUNCTION RETURNS THE DATA RATE (SPEED) TRANSITION
C   PROBABILITY BASED ON THE CURRENT ELEM  TYPE. THE  ALLOW-
C   ABLE TRANSITION PROBS ARE STORED IN THE TABLE RTRANS.
C
C    VARIABLES:
C       ISRT-   DATA RATE IDENTITY FOR STATE TO WHICH
C               PATH IS BEING EXTENDED
C       ILRT-   DATA RATE ON CURRENT PATH
C       ISELM-  ELEM TYPE FOR NEXT STATE
C       ILELM-  ELEM TYPE ON CURRENT PATH
C
C***********************************************************************

      DIMENSION RTRANS(5,2),MEMPR(6,6),MEMDEL(6,6)
      COMMON/BLKSPD/RTRANS,MEMPR
      COMMON/BLKRAT/MEMDEL

C
C   IF SAVED ELEMENT AND NEW ELEMENT ARE THE
C   SAME, THEN THERE CAN BE NO SPEED CHANGE:
C
      IF(ILELM.NE.ISELM) GO TO 100
      SPDTR=1.
      IF(ISRT.NE.3) SPDTR=0.
      GO TO 300

C
C   OTHERWISE, OBTAIN SPEED TRANSITION PROB:
C

 100  IDEL=MEMDEL(ILELM,ISELM)
      IND1=MEMPR(ILELM,ISELM)
      IF(IND1.NE.0) GO TO 200
      SPDTR=0.
      GO TO 300

 200  IDELSP=(ISRT-3)*IDEL
      SPDTR=RTRANS(ISRT,IND1)
      ISRATE=ILRT+IDELSP
      IF(ISRATE.GT.60) SPDTR=0.
      IF(ISRATE.LT.10) SPDTR=0.

 300  RETURN
      END
```

```
      SUBROUTINE PATH(IP,LAMBDA,DUR,ILRATE,LAMSAV,DURSAV,ILRSAV)

C***********************************************************************
C
C   PATH COMPUTES THE LTR STATE, DURATION, AND DATA RATE OF
C   EACH NEW PATH EXTENDED TO STATE N.
C
C   VARIABLES:
C       IP-       SAVED PATH IDENTITY
C       LAMBDA-   LTR STATE OF SAVED PATH
C       DUR-      DURATION OF ELEMENT ON SAVED PATH
C       ILRATE-   DATA RATE OF ELEMENT ON SAVED PATH
C       LAMSAV-   NEW LTR STATES FOR EACH PATH EXTENSION
C       DURSAV-   NEW ELEM DURATIONS FOR EACH PATH EXTENSION
C       ILRSAV-   NEW DATA RATES FOR EACH PATH EXTENSION
C       J-        NEW PATH IDENTITY
C
C   THE LETTER TRANSITION TABLE, MEMFCN, IS STORED IN COMMON.
C
C***********************************************************************

      DIMENSION LAMSAV( 750),DURSAV( 750),ILRSAV( 750)
      DIMENSION MEMFCN(400,6),IELMST(400),ILAM1(16)
      DIMENSION ILAMX(6),ISX(6),MEMDEL(6,6)

      COMMON/BLKLAM/IELMST,ILAM1,ILAMX
      COMMON/BLKMEM/MEMFCN
      COMMON/BLKS/ISX
      COMMON/BLKRAT/MEMDEL

C   FOR EACH ELEM STATE K, AND EACH SPEED I, COMPUTE:
C
      DO 100 I=1,5
      DO 100 K=1,6
C
C    STATE IDENTITY N:
C
      N=(I-1)*6+K
C
C    NEW PATH IDENTITY:
C
      J=(IP-1)*30+N
C
C    NEW LTR STATE:
C
      IF(LAMBDA.NE.0) GO TO 50
      LAMSAV(J)=0
      GO TO 100

 50   LAMSAV(J)=MEMFCN(LAMBDA,K)
      IF(LAMSAV(J).EQ.0) GO TO 100
C
C    NEW DURATION:
C
C     OBTAIN KEYSTATE OF SAVED PATH AND NEW STATE:
C
```

```
      ILELM=ILAM1(IELMST(LAMBDA))
      IXL=ILAMX(ILELM)
      IXS=ISX(K)
C
C     CALCULATE DURATION:
C
      DURSAV(J)=DUR*(1-IXS-IXL+2*IXS*IXL)+5.
C
C    NEW DATA RATE:
C
      ILRSAV(J)=ILRATE+(I-3)*MEMDEL(ILELM,K)

 100  CONTINUE

 200  RETURN
      END




      SUBROUTINE LIKHD(Z,RN,IP,LAMBDA,DUR,
     2  ILRATE,P,LKHD)

C*****************************************************************
C
C   THIS SUBROUTINE CALCULATES, FOR EACH PATH
C   EXTENSION TO STATE N, THE LIKELIHOOD OF THAT
C   TRANSITION GIVEN THE MEASUREMENT Z.  IT USES
C   AN ARRAY OF LINEAR (KALMAN) FILTERS TO DO SO.
C
C   VARIABLES:
C       Z-       INPUT MEASUREMENT
C       RN-      INPUT NOISE POWER ESTIMATE
C       IP-      INPUT SAVED PATH IDENTITY
C       LAMBDA-  INPUT SAVED LTR STATE IDENTITY
C       DUR-     INPUT SAVED DURATION OF ELEMENT ON PATH IP
C       ILRATE-  INPUT SAVED DATA RATE (SPEED)
C       P-       INPUT TRANSITION PROBABILITIES
C       LKHD-    OUTPUT COMPUTED LIKELIHOODS FOR EACH TRANS
C
C   SUBROUTINES USED:
C       KALFIL- KALMAN FILTER FOR EACH NEW PATH
C
C*****************************************************************

      REAL LKHD,LKHDJ
      DIMENSION P(25,30),LKHD(750)
      DIMENSION IELMST(400),ILAM1(16),ILAMX(6)
      DIMENSION ISX(6)

      COMMON/BLKLAM/IELMST,ILAM1,ILAMX
      COMMON/BLKS/ISX


C   OBTAIN SAVED KEYSTATE:
C

      KELEM=ILAM1(IELMST(LAMBDA))
      ILX=ILAMX(KELEM)
```

```
C
C     FOR EACH STATE:
C
            DO 100 K=1,6
            DO 100 I=1,5
C
C     OBTAIN KEYSTATE,RATE STATE,STATE N,NEW NODE:
C
            IXS=ISX(K)
            ISRATE=I
            N=(I-1)*6+K
            J=(IP-1)*30+N
            PIN=P(IP,N)
C
C     COMPUTE AND STORE LIKELIHOOD:
C
            CALL KALFIL(Z,TP,RN,ILX,IXS,KELEM,J,ISRATE,
         2  DUR,ILRATE,PIN,LKHOJ)

            LKHO(J)=LKHOJ
            GO TO 100
            IF(PIN.LE.1.E-06) GO TO 100
            TYPE 1000,IP,Z,LAMBDA,K,ILRATE,ISRATE,DUR,PIN,LKHOJ,RN
      1000  FORMAT(1X,I2,1X,F5.3,2X,I3,2X,I1,2X,I2,2X,I2,3X,F5.1,
         2  2X,F8.6,2X,F8.4,2X,F8.4)

      100   CONTINUE
      200   RETURN
            END
```

```
      SUBROUTINE KALFIL(Z,IP,RN,ILX,IXS,KELEM,
     2  JNODE,ISRATE,DUR,ILRATE,PIN,LKHDJ)

C************************************************************************
C
C   THIS SUBROUTINE COMPUTES THE ARRAY OF KALMAN FILTER
C   RECURSIONS USED TO DETERMINE THE LIKELIHOODS.
C
C   VARIABLES:
C       Z-       INPUT MEASUREMENT
C       IP-      INPUT PATH IDENTITTY
C       RN-      INPUT NOISE POWER ESTIMATE
C       ILX-     INPUT SAVED KEYSTATE ON PATH IP
C       IXS-     INPUT KEYSTAT OF NEW NODE
C       KELEM-   INPUT ELEM STATE OF NEW NODE
C       ISRATE-  INPUT SPEED STATE OF NEW NODE
C       DUR-     INPU CURRENT DURATION OF ELEMENT ON IP
C       ILRATE-  INPUT SPEED STATE ON PATH IP
C       PIN-     TRANS PROB FROM PATH IP TO NODE N
C       LKHDJ-   OUTPUT CALCULATED LIKELIHOOD VALUE
C
C   SUBROUTINES USED
C       MODEL- OBTAINS THE SIGNAL-STATE-DEPENDENT LINEAR
C       MODEL FOR THE KALMAN FILTER RECURSIONS
C
C************************************************************************

      REAL LKHDJ
      DIMENSION YKKIP(25),PKKIP(25)
      DIMENSION YKKSV(750),PKKSV(750)


      COMMON/BLKSV1/YKKIP,PKKIP,YKKSV,PKKSV



      DATA YKKIP/25*.5/,PKKIP/25*.10/
      DATA PINMIN/.00010/

C
C   IF TRANSITION PROBABILITY IS VERY SMALL, DON'T
C   BOTHER WITH LIKELIHOOD CALCULATION:
C
      IF(PIN.GT.PINMIN) GO TO 100
      LKHDJ=0.
      GO TO 400

C
C   OBTAIN STATE-DEPENDENT MODEL PARAMETERS:
C
 100  CALL MODEL(DUR,KELEM,ILRATE,ISRATE,IXS,PHI,QA,HZ)
C
C   GET PREVIOUS ESTIMATES FOR PATH IP
C
      YKK=YKKIP(IP)
      PKK=PKKIP(IP)

C
C   IMPLEMENT KALMAN FILTER FOR THIS TRANSITION:
C
```

```
      YPRED=PHI*YKK
      PPRED=PHI*PKK*PHI+QA
      PZ=HZ*PPRED+RN
      PZINV=1./PZ
      G=PPRED*HZ*PZINV
      PEST=(1.-G*HZ)*PPRED
      ZR=Z-HZ*YPRED
      YKKSV(JNODE)=YPRED+G*ZR
      PKKSV(JNODE)=PEST
      IF(YKKSV(JNODE).LE..01) YKKSV(JNODE)=.01
      A=0.5*PZINV*ZR**2
      IF(A.LE.1000.) GO TO 200
      LKHDJ=0.
      GO TO 400
200   LKHDJ=(1./SQRT(PZ))*EXP(-A)
      GO TO 400
      TYPE 1000,Z,HZ,QA,PHI,PZ,ZR,G,PEST,YKK,YKKSV(JNODE),LKHOJ
1000  FORMAT(1X,11(F6.3,1X),/)
400   RETURN
      END
      SUBROUTINE MODEL(DUR,IELM,ILR,ISR,IXS,PHI,QA,HZ)
C*************************************************************
C
C   THIS SUBROUTINE COMPUTES THE PARAMETERS OF THE
C   OBSERVATION STATE TRANSITION MATRIX PHI, THE
C   MEASUREMENT MATRIX, AND THE COVARIANCES.
C
C   VARIABLES:
C        DUR-   INPUT ELEMENT DURATION
C        IELM-  INPUT ELEMENT TYPE
C        ILR-   INPUT SAVED RATE
C        ISR-   INPUT RATE OF NEW STATE
C        IXS-   INPUT KEYSTATE OF NEW STATE
C        PHIA-  OUTPUT STATE TRANSITION MATRIX ENTRY FOR
C        SIGNAL AMPLITUDE STATE
C        QA-    OUTPUT COVARIANCE FOR AMPLITUDE STATE
C        HZ-    OUTPUT MEASUREMENT MATRIX VALUE
C
C*************************************************************
C
C   COMPUTE MEASUREMENT COEFFICIENT:
C
      HZ=IXS
```

```
C
C   COMPUTE PHI AND AMPLITUDE STATE VARIANCE (Q):
C
          R1=1200./ILR
          BAUDS=DUR/R1
          IF(BAUDS.GE.14.) BAUDS=14.

          IF(IFLM.GE.3) GO TO 100
          QA=.0001
          PHI=1.
          GO TO 300

 100      IF(IXS.EQ.0) GO TO 200
          PHI=1.
          QA=0.15*EXP(0.6*(BAUDS-14.))
          QA=QA+.01*BAUDS*EXP(.2*(1.-BAUDS))
          GO TO 300

 200      XSAMP=22.4*R1
          PHI=10.**(-2/XSAMP)
          IF(BAUDS.GE.14.) PHI=1.
          QA=0.

 300      RETURN
          END




          SUBROUTINE PROBP(P,PIN,ISAVE,LKHD)

C*****************************************************************
C
C     PROBP COMPUTES THE POSTERIOR PROBABILITY OF EACH
C     NEW PATH,
C
C   VARIABLES:
C         P-       INPUT: SAVED PROBS OF PRIOR PATHS
C         OUTPUT:COMPUTED POSTERIOR PROBS OF NEW PATHS
C         PIN-     INPUT TRANSITION PROBABILITIES
C         LKHD-    INPUT LIKELIHOODS OF EACH TRANSITION
C         PSUM-    NORMALIZING CONSTANT (SUM OF P(J))
C
C*****************************************************************

          REAL LKHD
          DIMENSION P( 750),PIN(25,30),LKHD( 750)
          DIMENSION PSAV( 750)


          PMAX=0.
          PSUM=0.
C
C   FOR EACH SAVED PATH, EACH TRANSITION:
C
          DO 100 I=1,ISAVE
          DO 100 N=1,30
C
C    COMPUTE IDENTITY OF NEW PATH:
```

```
C
      J=(I-1)*30+N
C
C     PRODUCT OF PROBS, ADD TO PSUM
C
      PSAV(J)=P(I)*PIN(I,N)*LKHD(J)
      PSUM=PSUM+PSAV(J)

      IF(PSAV(J).LE.PMAX) GO TO 100
      PMAX=PSAV(J)

 100  CONTINUE


C
C   NORMALIZE TO GET PROBABILITIES; SAVE:
C
      NI=30*ISAVE
      DO 200 J=1,NI
      P(J)=PSAV(J)/PSUM
 200  CONTINUE

      RETURN
      END





      SUBROUTINE SPROB(P,ISAVE,ILRSAV,PELM,KHAT,
     2 SPDHAT,PX)

C**************************************************************
C
C   SPROB COMPUTES THE POSTERIOR PROBS OF THE ELEMENT
C   STATES,DATA RATE STATES,AND KEYSTATES BY SUMMING
C   OVER THE APPROPRIATE PATHS.
C
C   VARIABLE:
C        P-       INPUT PATH PROBABILITIES
C        ISAVE-   NUMBER OF PATHS SAVED
C        PSELEM-  OUTPUT ELEMENT PROB
C        SPDHAT-  OUTPUT SPEED ESTIMATE (DATA RATE WPM)
C        PX-      OUTPUT KEYSTATE PROBABILITY
C
C**************************************************************

      DIMENSION P(750),PSELEM(6),ILRSAV(750)


C
C   INITIALIZE:
C
      SPDHAT=0.
      PX=0.
C
C   FOR EACH STATE EXTENSION OF PATH M:
C   OBTAIN ELEMENT STATE PROBS,KEYSTATE PROBS,SPEED EST:
C
      DO 100 K=1,6
      PSELEM(K)=0.
```

```
      DO 100 I=1,5

      N=(I-1)*6+K

      DO 100 M=1,ISAVE
      J=(M-1)*30+N
      PSELEM(K)=PSELEM(K)+P(J)
      SPDHAT=SPDHAT+ILRSAV(J)*P(J)
      IF(K.GT.2) GO TO 100
      PX=PX+P(J)
100   CONTINUE

      PELM=0.
      DO 200 K=1,6
      IF(PSELEM(K).LT.PELM) GO TO 200
      PELM=PSELEM(K)
      KHAT=K
200   CONTINUE

      RETURN
      END


      SUBROUTINE SAVEP(P,PATHSV,ISAVE,IMAX,LAMSAV,
     2 DURSAV,ILRSAV,LAMBDA,DUR,ILRATE,SORT)

C*********************************************************************
C
C   THIS SUBROUTINE PERFORMS THE ALGORITHM TO SAVE
C   THE PATHS WITH HIGHEST POSTERIOR PROBABILITY.
C   IT WILL SAVE A MINIIMUM OF  7 PATHS (ONE FOR EACH
C   ELEMENT STATE AND ONE ADDITIONAL NODE), AND
C   A MAXIMUM OF 25 PATHS.  WITHIN THESE LIMITS, IT
C   SAVES ONLY ENOUGH TO MAKE THE TOTAL SAVED PROBABILITY
C   EGUAL TO POPT.
C
C   ADDITIONALLY, IT RESORTS THE LAMBDA,DUR,AND ILRATE
C   ARRAYS TO CORRESPOND TO THE SAVED NODES.
C
C
C   VARIABLES:
C        P-       INPUT PROBABILITY ARRAY OF NEW NODES
C        PATHSV-  OUTPUT ARRAY OF THE PREVIOUS NODES TO
C                 WHICH THE SAVED NODES ARE CONNECTED.
C        ISAVE-   INPUT: NO. OF PREVIOUS NODES SAVED
C                 OUTPUT: NO. OF NODES SAVED AT CURRENT STAGE
C        IMAX-    INDEX OF HIGHEST PROBABILITY NODE
C        LAMSAV-  INPUT ARRAY OF LTR STATES AT EACH NEW NODE
C        DURSAV-  INPUT ARRAY OF SAVED DURATIONS
C        ILRSAV-  INPUT ARRAY OF SAVED RATES
C        LAMBDA-  OUTPUT ARRAY OF SAVED LTR STATES, SORTED
C                 ACCORDING TO PROBABILITY
C        DUR-     OUTPUT ARRAY OF SORTED DURATIONS
C        ILRATE-  OUTPUT ARRAY OF SORTED RATES
C
C*********************************************************************

      INTEGER PATHSV,SORT
      DIMENSION P( 750),PATHSV(25),PSAV(25),SORT(25)
      DIMENSION LAMSAV( 750),DURSAV( 750),ILRSAV( 750)
```

```
      DIMENSION LAMBDA(25),DUR(25),ILRATE(25)
      DIMENSION YKKIP(25),PKKIP(25)
      DIMENSION YKKSV(750),PKKSV(750)
      DIMENSION ICONV(25)

      COMMON/BLKSV1/YKKIP,PKKIP,YKKSV,PKKSV

      DATA POPT/.90/

      NSAV=0
      PSUM=0.
C
C   SELECT SIX HIGHEST PROB ELEMENT STATE NODES:
C
      DO 200 K=1,6
      PMAX=0.
      DO 100 IP=1,ISAVE
      DO 100 I=1,5
      J=(IP-1)*30+(I-1)*6+K
      IF(P(J).LT.PMAX) GO TO 100
      PMAX=P(J)
      JSAV=J
      IPSAV=IP
 100  CONTINUE
      IF(PMAX.GE.0.000001) GO TO 150
      GO TO 200


 150  NSAV=NSAV+1
      PSUM=PSUM+PMAX
      PSAV(NSAV)=PMAX
      PATHSV(NSAV)=IPSAV
      SORT(NSAV)=JSAV
 200  CONTINUE


C
C   SELECT ENOUGH ADDITIONAL NODES TO MAKE TOTAL
C   PROBABILITY SAVED EQUAL TO POPT,  OR A MAX
C   OF 25:
C
 520  PMAX=0.
      DO 500 IP=1,ISAVE
      DO 500 N=1,30
      J=(IP-1)*30+N
      DO 510 I=1,NSAV
      IF(J.EQ.SORT(I)) GO TO 500
 510  CONTINUE

      IF(P(J).LE.PMAX) GO TO 500
      PMAX=P(J)
      JSAV=J
      IPSAV=IP
 500  CONTINUE

      PSUM=PSUM+PMAX
      NSAV=NSAV+1
      PSAV(NSAV)=PMAX
      PATHSV(NSAV)=IPSAV
      SORT(NSAV)=JSAV
```

```
      IF(PSUM.GE.POPT) GO TO 600
      IF(NSAV.GE.25  ) GO TO 600
      GO TO 520

C
C   NEW ISAVE EQUALS NO. OF NODES SAVED:
C
 600  ISAVE=NSAV

C
C   SORT THE SAVED ARRAYS TO OBTAIN THE ARRAYS
C   TO BE USED FOR THE NEXT ITERATION:
C   ALSO OBTAIN HIGHEST PROBABILITY NODE:
C
      DO 700 I=1,ISAVE
      P(I)=PSAV(I)/PSUM
      LAMBDA(I)=LAMSAV(SORT(I))
      DUR(I)=DURSAV(SORT(I))
      ILRATE(I)=ILRSAV(SORT(I))
      YKKIP(I)=YKKSV(SORT(I))
      PKKIP(I)=PKKSV(SORT(I))
 700  CONTINUE

      DO 790 I=1,ISAVE
      ICONV(I)=1
 790  CONTINUE

      ISAVM1=ISAVE-1
      DO 800 N=1,ISAVM1
      IF(ICONV(N).EQ.0) GO TO 800

      NPLUS1=N+1
      DO 810 K=NPLUS1,ISAVE
      IF(ICONV(K).EQ.0) GO TO 810

      IF(ILRATE(K).NE.ILRATE(N)) GO TO 810
      IF(DUR(K).NE.DUR(N)) GO TO 810
      IF(LAMBDA(K).NE.LAMBDA(N)) GO TO 810
      ICONV(K)=0

 810  CONTINUE
 800  CONTINUE

      PSUM=0.
      N=1
      DO 900 I=2,ISAVE
      IF(ICONV(I).EQ.0) GO TO 900
      N=N+1
      LAMBDA(N)=LAMBDA(I)
      DUR(N)=DUR(I)
      ILRATE(N)=ILRATE(I)
      YKKIP(N)=YKKIP(I)
      PKKIP(N)=PKKIP(I)
      PATHSV(N)=PATHSV(I)
      SORT(N)=SORT(I)
      P(N)=P(I)
      PSUM=PSUM+P(N)
 900  CONTINUE

      ISAVE=N
```

```
      PMAX=0.
      DO 950 I=1,ISAVE
      P(I)=P(I)/PSUM
      IF(P(I).LE.PMAX) GO TO 950
      PMAX=P(I)
      IMAX=I
  950 CONTINUE

      RETURN
      END
```

```
      SUBROUTINE TRELIS(ISAVE,PATHSV,LAMBDA,IMAX)

C**********************************************************
C
C   THIS SUBROUTINE STORES THE SAVED NODES AT EACH
C   STAGE AND FORMS THE TREE OF SAVED PATHS LINKING
C   THE NODES.  DECODING IS ACCOMPLISHED BY FINDING
C   THE CONVERGENT PATH IF IT OCCURS WITHIN A MAXIMUM
C   DELAY SET BY THE PARAMETER NDELAY.  IF CONVERGENCE
C   TO A SINGLE PATH DOES NOT OCCUR, THEN DECODING IS
C   DONE BY READING THE LETTER ON THE PATH WITH HIGHEST
C   PROBABILITY.
C
C**********************************************************

      INTEGER PTHTRL,PATHSV
      DIMENSION PATHSV(25),LAMBDA(25),PTHTRL(200,25)
      DIMENSION LMDSAV(200,25),IPNOD(25),LTRSV(200)
      COMMON/BLKEND/IEND

      DATA PTHTRL/5000*5/,LMDSAV/5000*5/
      DATA N/0/,NDELAY/200/
      DATA IPNOD/25*1/,NCALL/0/,NMAX/0/,MMAX/0/

C
C   KEEP AVERAGE OF ISAVE,NDEL FOR DATA ANALYSIS:
C
      NCALL=NCALL+1
      IF(IEND.NE.1) GO TO 10
      ISAVG=XSAVG
      NDLAVG=XDLAVG
      IEND=0
      TYPE 2000,ISAVG,NDLAVG
 2000 FORMAT(1X,'AVG NO OF PATHS SAVED: ',I2,2X,
     2 'AVG DECODE DELAY: ',I3)
      TYPE 3000,XMMAX,XNMAX
 3000 FORMAT(1X,'PERCENT OF TIME PATHS=25: ',F3.2,
     2 2X,'PERCENT OF TIME DELAY=200: ',F3.2)
      ACCEPT 2000,WAIT
 10   XSAVG=(XSAVG*(NCALL-1)+ISAVE)/NCALL
      XDLAVG=(XDLAVG*(NCALL-1)+NDEL)/NCALL
      IF(NDEL.NE.NDELAY) GO TO 20
      NMAX=NMAX+1
      XNMAX=NMAX
      XNMAX=XNMAX/NCALL
 20   IF(ISAVE.NE.25) GO TO 30
      MMAX=MMAX+1
      XMMAX=MMAX
      XMMAX=XMMAX/NCALL
 30   CONTINUE


C
C   STORE PATHSV AND CORRESPONDING LAMBDA IN THE
C   TRELLIS USING A CIRCULAR BUFFER OF LENGTH NDELAY:
C
      N=N+1
      IF(N.EQ.NDELAY+1) N=1
      DO 100 I=1,ISAVE
      PTHTRL(N,I)=PATHSV(I)
```

```
      LMDSAV(N,I)=LAMBDA(I)
  100 CONTINUE

C
C   PERFORM DYNAMIC PROGRAM ROUTINE TO FIND
C   CONVERGENT PATH:
C
      K=0
      DO 180 I=1,ISAVE
      IPNOD(I)=I
  180 CONTINUE

  190 K=K+1
      IF(K.EQ.NDELAY) GO TO 700
      DO 200 IP=1,ISAVE
      I=N-K+1
      IF(I.LE.0) I=NDELAY+I
      IPNOD(IP)=PTHTRL(I,IPNOD(IP))
      IF(IP.EQ.IMAX) IPMAX=IPNOD(IP)
  200 CONTINUE

C   IF ALL NODES ARE EQUAL, THEN PATHS CONVERGE:
C
      DO 300 IEQ=2,ISAVE
      IF(IPNOD(1).NE.IPNOD(IEQ)) GO TO 190
  300 CONTINUE

C
C   PATHS CONVERGE; SET NDEL:
C
      NDEL=K+1

C
C   IF POINT OF CONVERGENCE IS SAME AS IT WAS ON
C   LAST CALL, THEN NO NEED TO RE-DECODE SAME NODE:
C
      IF(NDEL.EQ.NDELST+1) GO TO 800

C
C   IF POINT OF CONVERGENCE OCCURS AT SAME DELAY AS
C   LAST CALL, THEN TRANSLATE:
C
      IF(NDEL.NE.NDELST) GO TO 350
      I=N-NDEL+1
      IF(I.LE.0) I=NDELAY+I
      LTR=LMDSAV(I,IPNOD(1))
      CALL TRANSL(LTR)
      GO TO 800

C   OTHERWISE, POINT OF CONVERGENCE HAS OCCURED
C   EARLIER ON THIS CALL, SO NEED TO TRANSLATE
C   EVERYTHING ON THE CONVERGENT PATH FROM
C   PREVIOUS POINT OF CONVERGENCE TO THIS POINT:
C

  350 KD=0
      IP=IPNOD(1)
      DO 400 K=NDEL,NDELST
      KD=KD+1
      I=N-K+1
```

```
      IF(I.LE.0) I=NDELAY+I
      LTRSV(KD)=LMDSAV(I,IP)
      IP=PTHTRL(I,IP)
 400  CONTINUE

C
C   REVERSE ORDER OF DECODED LETTERS, SINCE THEY
C   WERE OBTAINED FROM THE TRELLIS IN REVERSE;
C   TRANSLATE EACH:
C
      DO 500 I=1,KD
      LTR=LTRSV(KD-I+1)
      CALL TRANSL(LTR)
 500  CONTINUE
      GO TO 800

 700  CONTINUE

C
C   PATHS HAVE NOT CONVERGED AT MAXIMUM ALLOWABLE
C   DELAY, SO TRANSLATE WHAT IS ON HIGHEST
C   PROBABILITY PATH:
C
      NDEL=NDELAY
      I=N-NDELAY+1
      IF(I.LE.0) I=NDELAY+I
      LTR=LMDSAV(I,IPMAX)
      CALL TRANSL(LTR)

C
C   PRUNE AWAY NODES WHICH ARE NOT ON
C   THIS PATH:
C
      DO 750 K=1,ISAVE
      IF(IPNOD(K).EQ.IPMAX) GO TO 750
      LAMBDA(K)=0
 750  CONTINUE


 800  NDELST=NDEL
      RETURN
      END





      SUBROUTINE TRANSL(LTR)

C**************************************************************************
C
C   THIS SUBROUTINE PRODUCES THE OUTPUT TEXT ON A CRT.
C   IT USES THE SIMPLE FORMATTING RULES DESCRIBED IN THE
C   TEXT.
C
C**************************************************************************

      INTEGER SPFLAG,ELMHAT,ELMOUT
      DIMENSION LTRMAP(400),IALPH(70),IBLANK(400)
      DIMENSION IELMST(400),ILAM1(16),ILAMX(6)
```

```
      COMMON/BLKTRN/LTRMAP,IALPH,IBLANK
      COMMON/BLKLAM/IELMST,ILAM1,ILAMX

      DATA ISPACE/' '/,SPFLAG/0/,NCHAR/0/


C     DETERMINE IF A CSP,WSP,OR PAUSE TO MARK TRANSITION
C     HAS OCCURED; IF SO LTR IS READY FOR OUTPUT:
C
      ELMHAT=ILAM1(IELMST(LTR))
      IXL=ILAMX(ELMHAT)
      IF(IXL.EQ.IXLAST) GO TO 700
      IF((IXL.EQ.1).AND.(LSTELM.GE.4)) GO TO 10
      IF((IXL.EQ.0).AND.(LSTELM.LE.2)) GO TO 700
      GO TO 700

 10   LTRHAT=LSTLTR
      LTROUT=IALPH(LTRMAP(LTRHAT))
      NBLANK=IBLANK(LTRHAT)
      ELMOUT=ILAM1(IELMST(LTRHAT))
      GO TO 40
      TYPE 5000,ELMOUT
 5000 FORMAT(1X,I1,$)
      NCHAR=NCHAR+1


 40   CONTINUE
      IF(LTRMAP(LTRHAT).EQ.43) GO TO 50
      IF(LTRMAP(LTRHAT).LE.44) GO TO 100
      IF(LTRMAP(LTRHAT).LE.46) GO TO 200
      IF(LTRMAP(LTRHAT).LE.60) GO TO 300
      IF(LTRMAP(LTRHAT).EQ.61) GO TO 400
      IF(LTRMAP(LTRHAT).EQ.66) GO TO 500
      GO TO 550

 50   IF(SPFLAG.EQ.0) GO TO 100

      NCHAR=0
      TYPE 1500,LTROUT
 1500 FORMAT(2X,A1,/)
      SPFLAG=1
      GO TO 600

 100  NCHAR=NCHAR+1
      TYPE 1000,LTROUT
 1000 FORMAT(1X,A1,$)
      SPFLAG=0
      IF(NBLANK.EQ.0) GO TO 110
      SPFLAG=1
      DO 110 I=1,NBLANK
      NCHAR=NCHAR+1
      TYPE 1000,ISPACE
 110  CONTINUE
      GO TO 600

 200  NCHAR=NCHAR+2
      TYPE 1100,LTROUT
 1100 FORMAT(1X,A2,$)
      SPFLAG=0
```

```
          IF(NBLANK.EQ.0) GO TO 210
          SPFLAG=1
          DO 210 I=1,NBLANK
          NCHAR=NCHAR+1
          TYPE 1000,ISPACE
  210     CONTINUE
          GO TO 600

  300     NCHAR=NCHAR+4
          TYPE 1200,LTROUT
  1200    FORMAT(2X,A2,2X,$)
          SPFLAG=1
          IF(NBLANK.EQ.0) GO TO 310
          DO 310 I=1,NBLANK
          NCHAR=NCHAR+1
          TYPE 1000,ISPACE
  310     CONTINUE
          GO TO 600 .

  400     NCHAR=NCHAR+5
          TYPE 1300,LTROUT
  1300    FORMAT(2X,A3,2X,$)
          SPFLAG=1
          GO TO 600

  500     NCHAR=0
          TYPE 1400,LTROUT
  1400    FORMAT(/,10X,A2,/,10X)
          SPFLAG=1
          GO TO 600

  550     NCHAR=NCHAR+5
          TYPE 1700,LTROUT
  1700    FORMAT(2X,A3,2X,$)
          SPFLAG=0
          IF(NBLANK.EQ.0) GO TO 560
          SPFLAG=1
          DO 560 I=1,NBLANK
          NCHAR=NCHAR+1
          TYPE 1000,ISPACE
  560     CONTINUE

  600     IF(NCHAR.LT.52) GO TO 700

          TYPE 1600
  1600    FORMAT(/,10X)
          NCHAR=0

  700     IXLAST=IXL
          LSTELM=ELMHAT
          LSTLTR=LTR

          RETURN
          END
```

```
      SUBROUTINE RCVR(ZIN,ZOUT)

C*****************************************************************
C
C   THIS SUBROUTINE CONVERTS THE INPUT SIGNAL AT
C   RADIAN FREQ WC TO 1000 HZ.
C
C*****************************************************************

      COMMON/BLK1/TAU/BLK2/WC

      DATA THETA/0./,THETLO/0./

      THETA=THETA+WC*TAU
      THETA=AMOD(THETA,6.28319)

      ZI=ZIN*COS(THETA)
      ZQ=ZIN*SIN(THETA)
      ZILP=ZILP+.070*(ZI-ZILP)
      ZQLP=ZQLP+.070*(ZQ-ZQLP)

      THETLO=THETLO+6283.2*TAU
      THETLO=AMOD(THETLO,6.28319)

      ZOUT= ZILP*COS(THETLO)+ZQLP*SIN(THETLO)

      RETURN
      END










      SUBROUTINE BPFDET(ZIN,Z)

C*****************************************************************
C
C   THIS SUBROUTINE IMPLEMENTS THE BANDPASS FILTER AND
C   ENVELOPE DETECTOR FUNCTIONS.  THE BPF IS A SIMPLE CASCADE
C   OF TWO 2-POLE DIGITAL RESONATORS AT A CENTER FREQ OF
C   1000 HZ. THE LPF OF THE ENVELOPE DETECTOR IS A
C   THREE-POLE CHEBYSCHEV 100 HZ LPF.
C
C*****************************************************************


      DIMENSION A(4)

      DATA A/.000030051,2.9507982,2.90396345,-.953135172/
      DATA CK1/1.37158/,CK2/.9409/,CG/.0150/
      DATA C1/1.2726/,C2/.8100/,C/.1900/

C
C   BPF IS TWO 2-POLE RESONATORS:
C
      W3=W2
      W2=W1
```

```
      W1=C1*W2-C2*W3+C*ZIN

      X3=X2
      X2=X1
      X1=CK1*X2-CK2*X3+CG*W1
      ZBPF=X1

C
C   ENVELOPE DETECTOR (SQUARE-LAW):
C    SQUARE-
C
      XDET=SQRT(ZBPF**2)

C
C    LOW-PASS FILTER-
C

      Y3=Y2
      Y2=Y1
      Y1=Y0
      Y0=XDET*A(1)

      Z3=Z2
      Z2=Z1
      Z1=Z
      Z=Y0+3.*(Y1+Y2)+Y3
      Z=Z+A(2)*Z1-A(3)*Z2-A(4)*Z3


      RETURN
      END




      SUBROUTINE NOISE(ZIN,RN,Z)

C*****************************************************************
C
C   THIS SUBROUTINE ESTIMATES THE NOISE POWER IN THE
C   ENVELOPE DETECTED OUTPUT FOR USE BY THE KALMAN
C   FILTERS.  AN ESTIMATE OF THE NOISE POWER IS
C   ALSO SUBTRACTED FROM THE ENVELOPE DETECTED SIGNAL
C   IN ORDER TO PRODUCE A ZERO-MEAN NOISE PROCESS
C   AT THE INPUT TO THE MORSE PROCESSOR.
C
C*****************************************************************

      DIMENSION YLONG(200),YSHORT(50)

      DATA YLONG/200*1./,YSHORT/50*1./
      DATA KL/1/,KKL/1/,KS/1/,KKS/1/
      DATA YMIN1/1./,YMIN2/1./,YMAVG/.05/

      KL=KL+1
      IF(KL.EQ.201) KL=1
      KS=KS+1
      IF(KS.EQ.51)KS=1
      KKL=KKL+1
      IF(KKL.GE.200) KKL=200
```

```
      KKS=KKS+1
      IF(KKS.GE.50) KKS=50

      IF(KKS.LE.2) GO TO 10
      YLONG(KL)=ZIN
      YSHORT(KS)=ZIN
      YMIN1=ZIN
      YMIN2=ZIN

10    DO 100 I=1,KKL
      IF(YLONG(I).GT.YMIN1) GO TO 100
      YMIN1=YLONG(I)
100   CONTINUE

      DO 200 I=1,KKS
      IF(YSHORT(I).GT.YMIN2) GO TO 200
      YMIN2=YSHORT(I)
200   CONTINUE

      YMIN=YMIN1
      IF(YMIN2.LT.YMIN1) YMIN=YMIN2

      YMAVG=YMAVG+.004*(YMIN-YMAVG)

      RN=0.30*YMAVG
      IF(RN.LT.0.005) RN=0.005
      Z=1.1*(ZIN-2.4*YMAVG-.05)

      RETURN
      END
```

LIST OF REFERENCES

1. Watt, A.D., Coon, R.M., Maxwell, E.L., and Plush, R.W., "Performance of Some Radio Systems in the Presence of Thermal and Atmospheric Noise," Proc. IRE, Vol 46, Dec 1958.

2. Bell, E.L., Processing of the Manual Morse Signal Using Optimal Linear Filtering, Smoothing, and Decoding, EE Thesis, Naval Postgraduate School, Monterey, Calif., Sept. 1975.

3. Lane, George, "Signal-to-Noise Requirements for Various Types of Radio Telegraphy Service," US Army Communications-Electronics Engineering Installation Agency, Electromagnetics Engineering Division, August 1975.

4. Gallager, R.G., Information Theory and Reliable Communication, John Wiley and Sons, Inc., New York, 1968.

5. Abramson, N., Information Theory and Coding, McGraw Hill, New York, 1963.

6. Stein, S. and Jones, J., Modern Communication Principles, McGraw-Hill, New York, 1967.

7. Carliolaro, G., and Pierobon, G., "Stationary Symbol Sequences from Variable-Length Word Sequences," IEEE Trans. Inf. Thy, v. IT-23, No. 2, MAR 1977.

8. Lee, R.C.K., Optimal Estimation, Identification and Control, The M.I.T. Press, Cambridge, Mass. 1964.

9. Sims, F.L. and Lainiotis, D.G., "Recursive Algorithm for the Calculation of the Adaptive Filter Weighting Coefficients," IEEE Trans. Auto. Control, vol AC14, no. 2, April 1969.

10. Wenersson, A., "On Bayesian Estimators for Discrete-Time Linear Systems with Markovian Parameters," TRITA-MAT-1975-5, Dept. of Math., Royal Inst. of Technology, Stockholm, Sweden, Jan. 1975.

11. Yakowitz, S., Williams, T., and Williams, G., "Surveillance of Several Markov Targets," IEEE Trans, Inf. Thy., vol IT-22, no. 6, Nov. 1976.

12. Gold, B., "Machine Recognition of Hand-sent Morse Code," IRE Trans. Inf. Thy., March 1959.

13. Meisel, A., et. al., "Morse Laboratory Data Analysis Report," Technology Services Corporation Report, May 1976.

14. Howe, D., Decision-Directed Modified Quasi-Bayes Estimation and Tracking of the Nonstationary Stochastic Parameters of a Communication Signal, Ph.D. Dissertation, The Catholic University of America, Washington, D.C., 1976.

15. Jelinek, F., Bahl, L., and Mercer, R., "Design of a Linguistic Statistical Decoder for the Recognition of Continuous Speech," IEEE Trans. Inf. Thy., Vol IT-21, no. 3, May 1975.

16. Bahl, L. and Jelinek, F., "Decoding for Channels with Insertion, Deletions, and Substitutions with Applications to Speech Recognition," IEEE Trans. Inf. Thy., Vol IT-21, no. 4, July 1975.

17. Fung, L., and Fu, K., "Maximum-Likelihood Syntactic Decoding," IEEE Trans. Inf. Thy., Vol IT-21, no. 4, July 1975.

18. Gelb, A. (editor), Applied Optimal Estimation, The M.I.T. Press, Cambridge, Mass., 1974.

19. Skolnik, M., Introduction to Radar Systems, McGraw-Hill, New York, 1962.

20. Davenport, W., Probability and Random Processes, McGraw-Hill, New York, 1970.

21. Schwartz, S., "The Estimator-Correlator for Discrete-time Problems," IEEE Trans. Inf. Thy., Vol IT-23, no. 1, Jan 1977.

22. Haccoun, D., and Ferguson, M., "Generalized Stack Algorithm for Decoding Convolutional Codes," IEEE Trans. Inf. Thy., Vol IT-21, no. 6, Nov 1975.

23. Engineering Design Handbook, Experimental Statistics, AMC Pamphlet 706-110, Headquarters, U.S. Army Materiel Command, Dec 1969.

BIBLIOGRAPHY

1. Bailey, A., and McCann, T., "Application of Printing Telegraph to Long-Wave Radio Circuits," Bell System Technical Journal, Vol X, Oct. 1931.

2. Zadeh, L.A., "Optimum Nonlinear Filters," J. Appl. Physics, Vol 24, no. 4, April 1953.

3. Gonzales, C. and Vogler, R., "Automatic Radio Telegraph Translator and Transcriber," Ham Radio, Nov. 1971.

4. Althoff, W.A., An Automatic Radiotelegraph Translator and Transcriber for Manually Sent Morse, Master's Thesis, Naval Postgraduate School, Monterey, Ca., Dec 1973.

5. Forney, G.D., "The Viterbi Algorithm," Proc. IEEE, Vol. 61, no. 3., March 1973.

6. Neuhoff, D.L., "The Viterbi Algorithm as an Aid in Text Recognition," IEEE Trans. Inf. Thy., Vol IT-21, no. 2, March 1975.

7. Manzingo, R.A., "DIscrete Optimal Linear Smoothing for Systems with Uncertain Observations," IEEE Trans. Inf. Thy., Vol IT-21, no. 3, May 1975.

8. Clements, D. and Anderson, B.D.O., "A Nonlinear Fixed-Lag Smoother for Finite-State Markov Processes," IEEE Trans. Inf. Thy., Vol IT-21, July 1975.

9. Alspach, D.L. and Sorenson, H.W., "Nonlinear Bayesian Estimation using Gaussian Sum Approximations," IEEE Trans. Auto. Control, Vol AC17, no. 4, August 1972.

10. Gray, R.M., "Sliding-Block Source Coding," IEEE Trans. Inf. Thy., Vol IT-21, no. 4, July 1975.

11. Gray, R.M., "Time-Invariant Trellis Encoding of Ergodic Discrete-Time Sources with a Fidelity Criterion," IEEE Trans. Inf. Thy., Vol IT-23, no. 1, Jan 1977.

12. Shields, P.C. and Neuhoff, D.L., "Block and Sliding-Block Source Coding," IEEE Trans. Inf. Thy., Vol IT-23, no. 2, March 1977.

13. Lainotis, D.G. (Editor), Estimation Theory, American Elsevier Publishing Co., New York, 1974.

14. Meditch, J.S., Stochastic Optimal Linear Estimation and Control, McGraw-Hill, New York, 1969.

15. Sage, A.P. and Melsa, J.L., Estimation Theory with Applications to Communications and Control, McGraw-Hill, New York, 1971.

16. Nahi, N.E., Estimation Theory and Applications, John Wiley & Sons, Inc., New York 1969.

17. Jazwinski, A.H., Stochastic Processes and Filtering Theory, Academic Press, New York, 1970.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Va. 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, Ca. 93940	2
3.	Professor Donald Kirk Department Chairman Department of Electrical Engineering Naval Postgraduate School Monterey, Calif. 93940	2
4.	Associate Professor Stephen Jauregui, Jr. Coe 52Ja Department of Electrical Engineering Naval Postgraduate School Monterey, Ca. 93940	10
5.	Dr. Robert Fossum Dean of Research Naval Postgraduate School Monterey, CA. 93940	1
6.	Professor C. Comstock, Code 53Zk Department of Mathematics Naval Postgraduate School Monterey, Ca. 93940	1
7.	Professor J. Ohlson, Code 5201 Dept. of Elec Engr. Naval Postgraduate School Monterey, Ca. 93940	1
8.	Professor H. Titus, Code 52Ts Dept. of Elec Engr Naval Postgraduate School Monterey, Ca. 93940	1
9.	Dr. J. Friedhoffer, Code R6 National Security Agency Ft. George G. Meade, Md. 29755	1

10. Lt. Edison L. Bell, Code R6 1
National Security Agency
Ft. George G. Meade, Md. 20755

11. Commander, Naval Security Group Command 1
Naval Security Group Headquarters
3801 Nebraska Ave., N.W.
Washington, D.C. 20890
ATTN: LCDR Campbell, G80

12. Commander, Naval Electronics Systems Command 3
Naval Electronics Systems Command Headquarters
PME-107
Washington, D.C. 20360
ATTN: Mr. R. Lesage, Mr. F. Lebert, CAPT. H. Leavitt, USN

13. Commander, Naval Electronics Laboratory Center 1
San Diego, California 92152
ATTN: Mr. J. Griffin

14. Director, National Security Agency 4
Group R
Ft. George G. Meade, MD 20755
ATTN: Mr. H. Rosenbloom
Mr. I. McElvy
Mr. R. Ettinger
Mr. C. Wayne

15. Army Security Agency 1
Unit Hill Farms Station
Warenton, Va. 22186
ATTN: Dr. White

16. TRW, Inc. 1
Bldg 90
1 Space Park
Redondo Beach, Ca. 90278
ATTN: Dr. B. Whalen

17. Sylvania, EDL Systems West 1
P.O. Box 205
Mountain View, Ca. 94040
ATTN: D. Jarvis

18. Pickering Radio Company 1
Professional Plaza
Portsmouth, R.I. 02871

19. ESL, Inc. 1
495 Java Dr.
Sunnyvale, California 94086
ATTN: W. Phillips

20. Sanders Assoc. 1
95 Canal Street
Nashua, New Hampshire 03060
ATTN: W. Zandi

Thesis
B36125 Bell 171670
c.1
Optimal Bayesian estimation of the state of a probabilistically mapped memory-conditional Markov process with application to manual Morse decoding.