JPL PUBLICATION 79-22

Some Practical Universal Noiseless Coding Techniques

Robert F. Rice

(NASA-CR-158515) SOME PRACTICAL UNIVERSAL NOISELESS CODING TECHNIQUES (Jet Propulsion Lab.) 132 p HC A07/MF A01 CSCL 09B

N79-22805

Unclas
G3/63 25087

March 15, 1979

National Aeronautics and Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

JPL PUBLICATION 79-22

Some Practical Universal Noiseless Coding Techniques

Robert F. Rice

March 15, 1979

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

ACKNOWLEDGMENT

I wish to gratefully acknowledge the helpful suggestions and simulation support of Alan Schlutsmeyer. I would also like to thank Marie Curry and her associates for their competent assistance in preparing this report for publication.

The research described in this paper was carried out by the Information Systems Division of the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the National Aeronautics and Space Administration (NASA).

ABSTRACT

Discrete data sources arising from real problems are generally characterized by only partially known and varying statistics. This report provides the development and analysis of some practical adaptive techniques for the efficient noiseless coding of a broad class of such data sources.

Specifically, algorithms are developed for coding discrete memoryless sources which have a known symbol probability ordering but unknown probability values. A general applicability of these algorithms to solving practical problems is obtained because most real data sources can be simply transformed into this form by appropriate preprocessing.

These algorithms have exhibited performance only slightly above all entropy values when applied to real data with stationary characteristics over the measurement span. However, performance considerably under a measured average data entropy may be observed when data characteristics are changing over the measurement span. The latter observation is a result of the ability to adjust to both short term and long term variations in data characteristics.

These techniques are applicable to virtually any alphabet size arising in practice. A subset of these results is the specification and analysis of a large class of efficient adaptive coders

for a binary memoryless source which is characterized by unknown or varying statistic p_0 (probability of a zero). Again, performance will be slightly above the binary entropy function when p_0 is unchanging but will typically be well under a measured average binary entropy when p_0 is changing over the measurement span.

These techniques are both easy to use and amenable to practical high rate implementations. Functions of sums of data samples provide tight bounds to algorithm performance. Thus investigations of the effects of alternative algorithm or preprocessing configurations can be accomplished without the need for complete coder simulations. These same functions also serve to simplify internal decision making. Partially as a result of the unique cascading of variable length coding operations, the only implementation requirement for storage of code words is eight binary codewords of which the longest is five bits.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

I. INTRODUCTION

The basic problem we are addressing is one in which discrete data sources are to be coded into binary representations from which the original can be retrieved precisely. Thus these binary mappings are reversible. Standard binary representations of a fixed number of bits/sample is the most obvious and well known example of such mappings.

The statistical characteristics of most real data sources is such that certain things happen more often than others. Then, quite intuitively, it should be possible to reduce the average number of bits required by representing the frequently occurring events with fewer bits than the infrequent ones. Indeed this is the case.

The vast majority of practical attempts at coding of real sources to remove this inherent statistical redundancy have used a particular approach, with quite predictable results. Specifically, after reversible preprocessing to produce a near memoryless source (take differences from predicted values, run lengths, etc.) the usual approach to coding has been to determine a probability distribution and then use the famous Huffman algorithm to obtain an "optimum" variable length code. Unfortunately, this optimality is quite restrictive. The Huffman derived code would give the best average performance of any <u>single</u> prefix code (lowest bits/sample) on a source for which symbols <u>always</u> occurred according to the <u>assumed</u> probability distribution. But the statistical characteristics of almost any real data source change with time, sometimes dramatically. The assumed probability distribution used to derive and perhaps test the Huffman code might be utterly wrong part of the time when monitored over a very long sequence. At the same time it may simply be a distribution which is the result of averaging out many short term variations in data characteristics. The point here is that there is room for improvement by adapting the coding procedures to fit the changing data characteristics.

The theoreticians have only recently started attacking this problem in earnest under the name "universal noiseless coding."[1] A set of practical adaptive variable length coding procedures was developed some time ago[2], [3] for the specific task of providing efficient coding of spacecraft imaging data where exact reproduction was a requirement. These algorithms would now come under most people's definition of universal noiseless coding. However, because of the rather specialized nature of the spacecraft application the practical versatility of these algorithms to efficiently code a wide variety of data sources has often been overlooked. This report will reintroduce them in a way which will hopefully make their general applicability both obvious and easy to accomplish. Using these algorithms as building blocks, more sophisticated coding systems offering additional performance benefits will be developed. The latter results are currently used in an imaging data compression system called RM2[4]-[8].

PROBLEM DEFINITION AND BACKGROUND

This section provides further discussion of the practical problem that subsequent chapters will address as well as the preliminary notation necessary to proceed.

Some Basic Notation Conventions

Concatenation. If $\tilde{X}$ and $\tilde{Y}$ are two sequences of samples then we can form a new sequence $\tilde{Z}$ by running them back to back as

$$\tilde{Z} = \tilde{X} * \tilde{Y} \qquad (1\text{-}1)$$

using the asterisk as our basic indication of concatenation. However, we will occasionally omit the * where no confusion should result.

Length of a Sequence. Any sequence of non-binary samples can be represented by a sequence of bits using the familiar standard binary representations which use a fixed number of bits. Without any anticipated confusion operator $\mathscr{L}(\cdot)$ will be used to specify the length of such a sequence in samples or in bits (of its standard binary representation). Then if $\tilde{X}$ is a sequence of J samples

$$\mathscr{L}(\tilde{X}) = J \text{ samples} \tag{1-2}$$

and if the standard binary representation of $\tilde{X}$ required 6 bits/sample

$$\mathscr{L}(\tilde{X}) = 6J \text{ bits} \tag{1-3}$$

If $\tilde{X}$ is already a binary sequence (e.g., the result of coding) then $\mathscr{L}(\hat{X})$ will simply mean the length of $\tilde{X}$ in bits.

General Form of Reversible Operators

It is instructive to characterize the general form that reversible operators will take. Subsequent developments will seem much less abstract. Let $\tilde{Z}$ be some sequence of, possibly correlated, data, and let π be a priori or side information about $\tilde{Z}$. Then a reversible operator of $\tilde{Z}$ would take the form

$$\tilde{Z}' = F[\tilde{Z},\pi] = f_1[\tilde{Z},\pi] * f_2[\tilde{Z},\pi] * \cdots \tag{1-4}$$

Each of the $f_i[\cdots]$ represent mapping operations of $\tilde{Z}$ and π which individually may not be reversible, but if all the $f_i[\tilde{Z},\pi]$ and π are available then $\tilde{Z}$ may be recovered precisely. That is, $F[\cdots]$ has an inverse.

When

$$\mathscr{S}(\tilde{Z}') < \mathscr{S}(\tilde{Z}) \tag{1-5}$$

in bits then we have achieved a more efficient binary representation of $\tilde{Z}$. If this is true for many different $\tilde{Z}$, $F[\cdot\cdot\cdot]$ may be a useful <u>code operator</u>. Whether this is true or not, $\tilde{Z}'$ can always be viewed as a sequence of symbols to which reversible operators of the form in (1-4) can be applied. Thus we might generate

$$\tilde{Z}'' = F'[\tilde{Z}',\pi] = F'[F[\tilde{Z},\pi]] \tag{1-6}$$

or by relabeling

$$\tilde{Z}'' = F''[\tilde{Z},\pi] \tag{1-7}$$

Again, if $\mathscr{S}(\tilde{Z}'') < \mathscr{S}(\tilde{Z})$ for many different Z sequences, operator $F''[\cdot\cdot\cdot]$ may also be a useful code operator. We will make use of these observations in later chapters. Complex sequence code operators will be built up from simpler ones.

A Notational Convention

The code operator structures that we will develop and are interested in identifying generally have many possible alternative internal parameters. Quite obviously, carrying all these parameters within discussions and block diagrams would present an unwieldy, if not impossible, notation problem. To avoid this we will simply omit explicit reference to internal parameters when naming operators. We adopt the convention of subscripting and superscripting the symbol ψ (and occasional other symbols) to identify operators (structures) and implicitly assume that a detailed specification can be obtained by reference to a corresponding parameter string. The development of "the Basic Compressor" in Chapter II will be used as a vehicle for introducing this convention.

Simplifying the Problem

A great many practical coding problems which may look quite different on the surface can be transformed into coding problems with very similar characteristics. Consequently, a solution to a transformed coding problem can have general applicability. The form of this transformed coding problem will be developed in subsequent paragraphs.

Removing correlation. In real problems where samples of $\tilde{Z}$ are correlated with themselves or a priori information π, there is usually some simple transformation which results in new sequences $\tilde{Z}'$ such that the samples are approximately independent. More important, the uncertainty in what the sample values will be is usually greatly reduced. The less uncertainty there is the greater the potential for reducing the average bits required to code. This step is crucial in many practical problems but we will, for the most part, assume it is already accomplished. That is, data sequences will be assumed to be from approximately memoryless sources (no correlation). The user of algorithms to be developed here would then precede them with appropriate correlation removing transformations: operators of the form in (1-4).

Examples of correlation removing transformations abound. Taking differences between adjacent samples along a TV line results in approximately independent difference samples which tend to be tightly distributed about zero (less uncertainty). A priori information might be the preceding line; appropriate use of this additional information generally leads to a similar result but with samples more tightly distributed about zero (less uncertainty still). A sequence of samples might also be successive states of a Markov Source or run lengths from a run length coder.

Symbol labeling. Given q possible symbols from some source it is a simple matter to relabel them into the numbers 0, 1, 2, $\cdots$ q-1. Unless noted otherwise we will assume that such relabeling has already been done.

Symbol probability ordering. As part of the same problem, let $\tilde{P} = \{p_i\}$ be the probability distribution of symbols 0, 1, 2, $\cdots$ q-1. For a wide class of practical problems the probability ordering of symbols after correlation reducing operations is a priori known (or at least well approximated). In fact, this ordering tends to remain the same (or close) even as the actual $\tilde{P}$ may be changing quite dramatically (e.g., consider again the independent difference samples along a TV scan line). It is again a simple matter to relabel source symbols, if necessary, so that the following conditions are well approximated.

$$p_0 \geq p_1 \geq p_2 \quad \cdots \geq p_{q-1} \tag{1-8}$$

Preprocessing summary. Thus we will generally assume that data to be coded has been preceded by the reversible preprocessing operations summarized in Fig. 1-1.

Changing $\tilde{P}$

If the symbol probability distributions resulting from the preprocessing operations just described were always known and fixed then there would be little need to proceed any further. The standard procedure of using the Huffman algorithm[9] to derive an optimum variable length code for the known distribution $\tilde{P}$ would yield coding efficiency about as good as could be expected (unless of course there is [illegible]m for improvement of the preprocessing operations). However, most real world problems are best characterized by a "changing and poorly defined $\tilde{P}$."

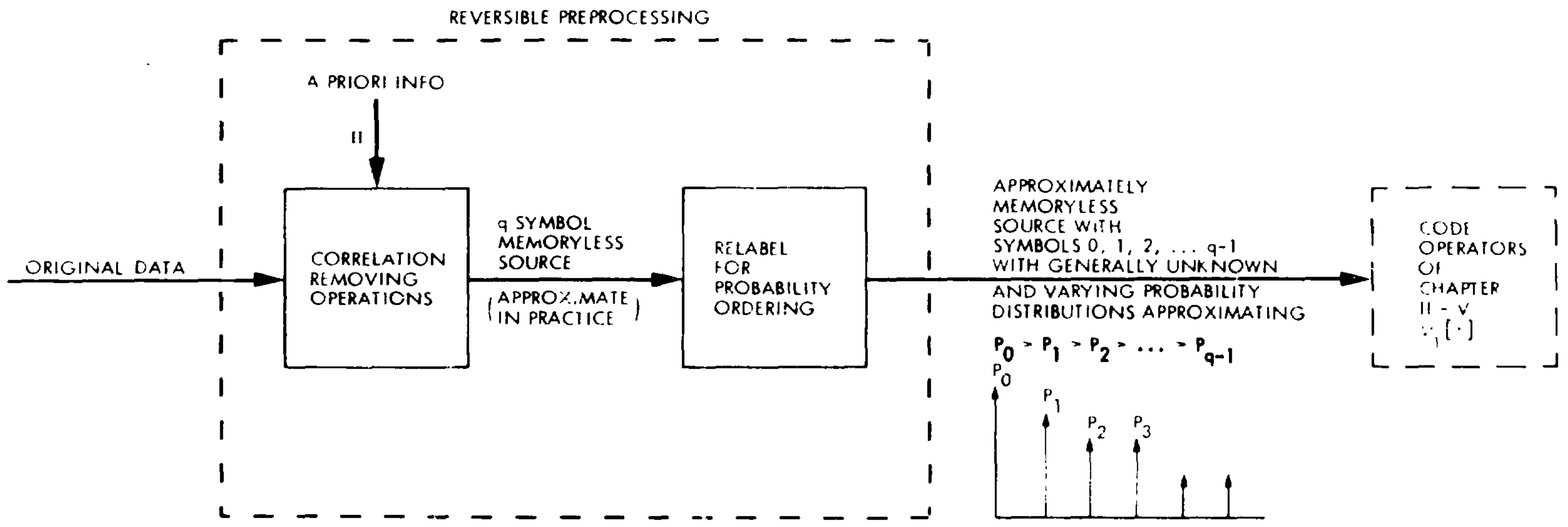

REVERSIBLE PREPROCESSING
A PRIORI INFO
II
ORIGINAL DATA
CORRELATION REMOVING OPERATIONS
q SYMBOL MEMORYLESS SOURCE
(APPROXIMATE IN PRACTICE)
RELABEL FOR PROBABILITY ORDERING
APPROXIMATELY MEMORYLESS SOURCE WITH SYMBOLS 0, 1, 2, ... q-1 WITH GENERALLY UNKNOWN
AND VARYING PROBABILITY DISTRIBUTIONS APPROXIMATING
P0 > P1 > P2 > ... > Pq-1
P0
P1
P2
P3
CODE OPERATORS OF CHAPTER II - V

Fig. 1-1. Reversible Preprocessing

Variations of $\tilde{P}$ in real problems appear in many different ways. $\tilde{P}$ may vary simply because separate short sequences are the result of preprocessing different data sources. There may be long and short term statistical variations in a single data source (e.g., picture to picture variations in image data caused by totally different scenes, different camera, lighting, etc. and local variations for similar reasons). Other than the approximate probability ordering, (1-8), $\tilde{P}$ may not be known at all. In any case a selected "optimum" Huffman code may perform quite poorly when the actual symbol distribution is different than the assumed. This is basically the problem we seek a practical remedy for in subsequent chapters.

<u>Practical Measures of Performance</u>

<u>Entropy definition.</u> Given the discrete symbol probability distribution $\tilde{P} = \{p_i\}$ the entropy $H(\tilde{P})$ is defined by

$$H(\tilde{P}) = -\sum_i p_i \log_2 p_i \text{ bits/sample} \qquad (1\text{-}9)$$

When properly used, $H(\tilde{P})$ can be a useful practical tool in assessing how well a particular coding algorithm performs.

<u>Interpretation of $H(\tilde{P})$.</u> If $\tilde{Z}$ is an infinite sequence of samples from a memoryless source with fixed and known symbol probability distribution $\tilde{P} = \{p_i\}$ then $H(\tilde{P})$ represents the minimum possible expected bits/sample required to represent $\tilde{Z}$ using any coding technique. But as we have just noted most practical problems which can be transformed by preprocessing into equivalent memoryless problems (Fig. 1-1) are characterized by changing or possibly unknown distributions. In practice it is generally difficult if not impossible to meaningfully model the way in which $\tilde{P}$ changes, although the fact

that it changes may be quite obvious. Consequently, the equivalent "bounds" for real data sources with changing $\tilde{P}$ are difficult to come by and we will not pretend to develop any here. Instead we will principally use (1-9) as a "practical measure of performance" rather than a bound on expected performance. The reader may consult the theoretical literature for performance bounds on idealized data sources.

Except where explicitly noted, the stated performance of a particular code operator $\psi_j[\cdot]$ will be based on measured performance on real data rather than statistical expectation based on some idealized model. We will generally use a span of K samples much greater than the length of sequence that $\psi_j[\cdot]$ operates on. Similarly, an average symbol probability distribution, $\overline{P}$, derived from a histogram of the same K samples can be used to provide the desired practical measure of performance $H(\overline{P})$.

If the real data has a somewhat uniform statistical character over the K samples then $H(\overline{P})$ represents a practical <u>bound</u> to average per sample performance of any $\psi_j[\cdot]$. An algorithm is performing efficiently if its measured average performance is close to $H(\overline{P})$. However, if data character changes significantly over the K samples then average per sample performance under the measured $H(\overline{P})$ may be possible by adapting the coding to suit the changes. $H(\overline{P})$ is still a useful guide in those cases and does in fact bound the best performance available with a single code (e.g., a Huffman code designed for $\overline{P}$).

SUMMARY OF RESULTS

The principal result of Chapter II is the development of a code operator called the Basic Compressor which provides measured performance close to $H(\overline{P})$ ($\overline{P}$ a priori unknown but stable over the measurement span) for values of $H(\overline{P})$ in the range of 0.7 to 4.0 bits/sample. This operator should have broad applicability and Chapters III - V are examples of using it to generate new operators with additional characteristics.

In Chapter III operators are developed which are capable of extending these results to much higher values of $H(\overline{P})$. Similarly, code operators are developed in Chapter IV which are capable of providing average performance close to $H(\overline{P})$ for $H(\overline{P}) \rightarrow 0$ as well as the higher entropies. An outgrowth of these developments is a class of binary memoryless coders capable of performing close to the binary entropy function as the (a priori unknown) probability of a binary zero or one varies between 0.0 and 1.0. These binary operators are described in Chapter V.

In all cases, performance under $H(\overline{P})$ has been observed for these operators when the data characteristics change substantially over the span of samples used for measurement.

An extremely useful practical characteristic of these algorithms is that in each case accurate estimates (actually bounds) of actual performance can be obtained basically as simple functions of the sum of input samples. This allows for accurate performance assessments without the need for elaborate simulations involving the generation of bit streams. This can greatly simplify the determination of various parameters (e.g., assessing a preprocessor) as well as aiding in the creation of new algorithms (which use the Basic Compressor as a basic tool). Additionally these same functions serve to simplify internal decision making.

II. THE BASIC COMPRESSOR

This chapter will provide the development of an algorithm for efficiently representing blocks of J preprocessed (see Fig. 1-1) samples when $\overline{P}$ is unknown and the measured average entropy $H(\overline{P})$ lies in the range of roughly 0.7 to 4 bits/sample. This Basic Compressor operator will be used as a basic tool in developing operators with additional characteristics in Chapters III - V.

As noted in Chapter I our primary means of identifying a large number of different code operators will be to subscript and superscript the symbol ψ. However, the first five operators $\psi_0[\cdot]$ - $\psi_4[\cdot]$ will receive dual names to avoid confusion to those readers familiar with the original description.[2], [3] The original names offer an additional benefit of being easy to remember.

Several coding examples are given at the end of this chapter.

FUNDAMENTAL SEQUENCE

Rather than seek a code which is optimum for some particular probability distribution, consider instead a code which is probably the simplest to implement and determine the range of $\tilde{P}$ over which it provides "good" performance.

Define the code word operator fs[$\cdot$] by

$$fs[i] \equiv \overbrace{000\ .\ .\ .\ .\ .\ 000}^{\text{i zeroes}}\ 1 \qquad (2\text{-}1)$$

where i is an input symbol. The length of codeword fs[i] is

$$\ell_i = \mathscr{L}\Big(fs[i]\Big) = i+1 \text{ bits} \qquad (2\text{-}2)$$

The coding of J preprocessed sample sequence $\tilde{X} = x_1 \ldots x_J$ using fs[$\cdot$] is given as

$$\psi_1[\tilde{X}] = FS[\tilde{X}] \equiv fs[x_1] * fs[x_2] * \ldots * fs[x_J] \qquad (2\text{-}3)$$

and will be called a fundamental sequence. We have thus defined our first code operator for sequences by $\psi_1[\cdot] = FS[\cdot]$.

The length of a fundamental sequence is

$$F = \mathcal{L}\left(FS[\tilde{X}]\right) = \sum_{j=1}^{J} \mathcal{L}\left(fs[x_j]\right) = J + \sum_{j=1}^{J} x_j \qquad (2\text{-}4)$$

or simply the block size plus the sum of the input samples. Note that when $\tilde{X}$ is the all zero sequence $FS[\tilde{X}]$ represents $\tilde{X}$ with J bits.

Observations

Because of the assumed probability ordering of input symbols in (1-8) and the codeword lengths in (2-2), shorter codewords will be used more often than longer ones. As noted in Chapter I, this condition can be well approximated for a wide variety of real problems by suitable preprocessing. The latter is necessary to make the most of a given variable length code such as (2-1).

A useful practical characteristic of the particular code in (2-1) is the fact that its definition does not depend on input alphabet size. This assures a wide applicability of the results to follow in later paragraphs.

FS Performance

A plot of the average per sample performance of code operator $FS[\cdot]$ is shown in Fig. 2-1 as a function of measured data entropy $H(\bar{P})$ over K samples where $K \gg J$.

The graph was derived from the results of preprocessing many forms of data such that condition (1-8) was well approximated. The fact that many distributions can yield the same entropy under these loose conditions is not of any practical significance. The graph is not intended to be that precise. The main point is that $FS[\cdot]$ performance tends to remain close to data entropy $H(\bar{P})$ in the range of roughly 1.5 to 3.0 bits/sample. If $H(\bar{P})$ were always restricted

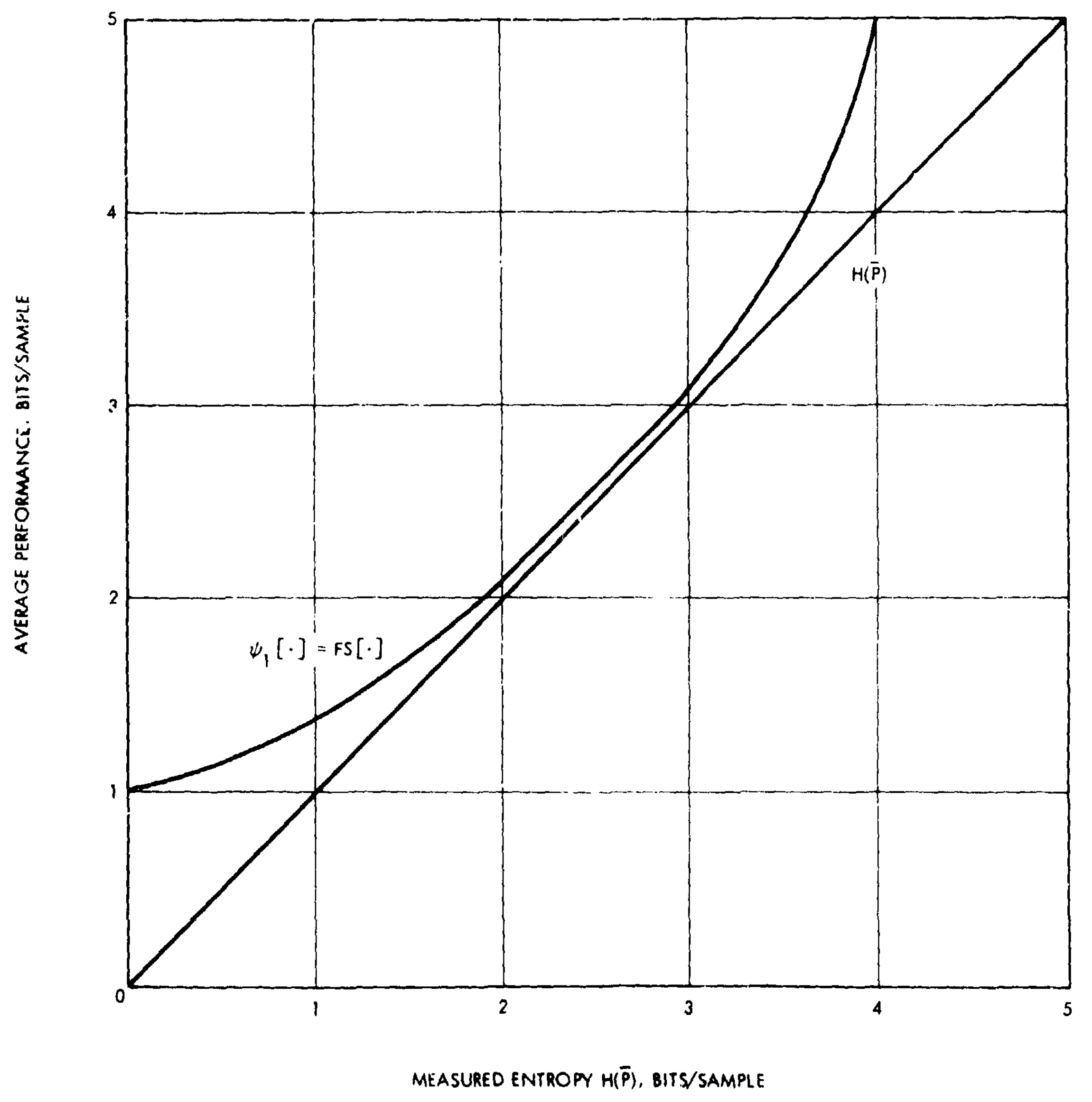


Fig. 2-1. Average FS[··] Performance

to this range then, considering the simplicity of fs[·], there would be little point in doing anything else. This is not always the case, however.

FOUR CODE OPTIONS

To extend good performance outside of this entropy range we could seek to find other codes which performed well for higher and lower entropies. But in addition we want such code operators to have the same versatility and simplicity as FS[·]. In particular we wish these operators to be applicable to any practical alphabet size, q, without substantially increasing complexity. Before defining these alternative code operators we need some additional definitions.

Additional Definitions[†]

Sequence extension. Let $\tilde{Y} = y_1 y_2 \ldots y_J$ be any J sample sequence. Given the positive integer $e \geq 1$ an extended sequence is formed by terminating $\tilde{Y}$ with enough dummy zeroes to make the resulting sequence length a multiple of e $\left(\text{i. e., } e\left\lceil \frac{J}{e} \right\rceil\right)$ as in

$$\tilde{Y}^e = \overbrace{y_1 y_2 \ldots y_J \underbrace{000 \ldots 0}_{\text{dummies}}}^{e\left\lceil \frac{J}{e} \right\rceil} \tag{2-5}$$

The eth extension of sequence $\tilde{Y}$ is then obtained by simply grouping the consecutive $J' = \left\lceil \frac{J}{e} \right\rceil$ e-tuples of $\tilde{Y}^e$ such as

$$\begin{aligned} \tilde{Y}' &= z_1 z_2 \cdots z_{J'} = \text{Ext}^e[\tilde{Y}] \\ &= (y_1 y_2 \cdots y_e) * (y_{e+1} y_{e+2} \cdots) * \cdots \\ &\cdots * (y_{J-1} y_J 00 \cdots 0) \end{aligned} \tag{2-6}$$

† $\lceil \alpha \rceil$ means the smallest integer greater than or equal to α.

If each sample of $\tilde{Y}$ could take on any of q values then each sample (except possib'y the last) of $\tilde{Y}'$ could take on any of q^e values. For our purposes we will use the reversible sequence operator $\mathrm{Ext}^c[\cdot]$ when $\tilde{Y}$ is binary so that samples of $\tilde{Y}'$ can take on 2^e values.

As an example, let $\tilde{Y}$ be the 34 sample binary sequence

$$\tilde{Y} = 1111100000111100000110000000000111011 \tag{2-7}$$

then the 3rd extension of $\tilde{Y}$ is given as

$$\tilde{Y}' = \mathrm{Ext}^3[\tilde{Y}] = (111) * (110) * (000) * (111) * (100)$$

$$* (001) * (100) * (000) * (000) * (011) * (101) * (100) \tag{2-8}$$

where we have added two dummy zeroes to complete the $\left\lceil \frac{34}{3} \right\rceil = 12$th sample of $\tilde{Y}'$.

Complementation. Given any binary sequence we let

$$\mathrm{COMP}[\cdot] \tag{2-9}$$

denote the operation of complementing each bit of the sequence (i. e., ones complement).

Coding FS[$\tilde{X}$]

Instead of seeking to code $\tilde{X}$ directly in a standard way we instead attempt to remove statistical redundancy that may be present in a fundamental sequence.

First let

$$\overline{\mathrm{FS}}[\tilde{X}] \triangleq \mathrm{COMP}\left[\mathrm{FS}[\tilde{X}]\right] \tag{2-10}$$

be the result of compl. nenting each bit of fundamental sequence FS$[\tilde{X}]$ and call it "FS bar." Now define

$$\tilde{a} = \mathrm{Ext}^3\left[FS[\tilde{X}]\right] = a_1\, a_2 \cdots \tag{2-11}$$

$$\tilde{b} = \mathrm{Ext}^3\left[\overline{FS}[\tilde{X}]\right] = b_1\, b_2 \cdots \tag{2-12}$$

where $\tilde{a}$ is the $\left\lceil \frac{J}{3} \right\rceil$ sample 3rd extension of a fundamental sequence, and $\tilde{b}$ is similarly the 3rd extension of its complement (see example in 2-7 and 2-8). By these operations we have simply lumped the fundamental sequence, and its bit by bit complement, into 3-tuples and added enough dummy zeroes (none, one or two) to complete the last 3-tuple.

<u>8-word code</u>. A simple 3-word variable length code i .efined in Table 2-1.

Table 2-1. 8-word code, cfs[·]

Input 3-tuple α	Output Codeword cfs$[\alpha]$
0 0 0	0
0 0 1	1 0 0
0 1 0	1 0 1
1 0 0	1 1 0
0 1 1	1 1 1 0 0
1 0 1	1 1 1 0 1
1 1 0	1 1 1 1 0
1 1 1	1 1 1 1 1

If we view the bits making up a fundamental sequence as approximately binary memoryless then 3-tuples with more zeroes will tend to occur more often when zeroes are more likely than ones (active data). Under these conditions the assignment of codeword cfs$[\alpha]$ to binary 3-tuple α in Table 2-1 assures that shorter codewords will be used more often than longer ones. Sequence $\tilde{a}$ in (2-11) is ready for a direct application of cfs[$\cdot$].

When binary ones are more likely than zeroes (inactive data) the situation is reversed; 3-tuples with more ones are more likely. In this case the assignment of shorter codewords in Table 2-1 to the more likely 3-tuples can be accomplished by flipping the right-hand column over or by simply complementing all input 3-tuples. Then sequence $\tilde{b}$ in (2-12) represents a preprocessed fundamental sequence which is ready for a direct application of cfs[$\cdot$].

We are now ready to define two additional code operators called "code fs" and "code fs bar". These are defined as follows

$$\psi_2[\tilde{X}] = \mathrm{CFS}[\tilde{X}] \equiv \mathrm{cfs}[a_1] * \mathrm{cfs}[a_2] * \ldots \qquad (2\text{-}13)$$

and

$$\psi_0[\tilde{X}] = \mathrm{C\overline{FS}}[\tilde{X}] = \mathrm{cfs}[b_1] * \mathrm{cfs}[b_2] * \ldots \qquad (2\text{-}14)$$

where the a_i and b_i are defined in (2-11) and (2-12).

As will be seen shortly, the fact that a binary memoryless model is not a perfect match for a fundamental sequence is of no practical significance.

Block diagram. Block diagrams describing code operators CFS[$\cdot$] and C$\overline{\text{FS}}$[$\cdot$] are shown in Fig. 2-2. We maintain the dual notation of $\psi_0[\cdot]$ and $\psi_2[\cdot]$ for later use.

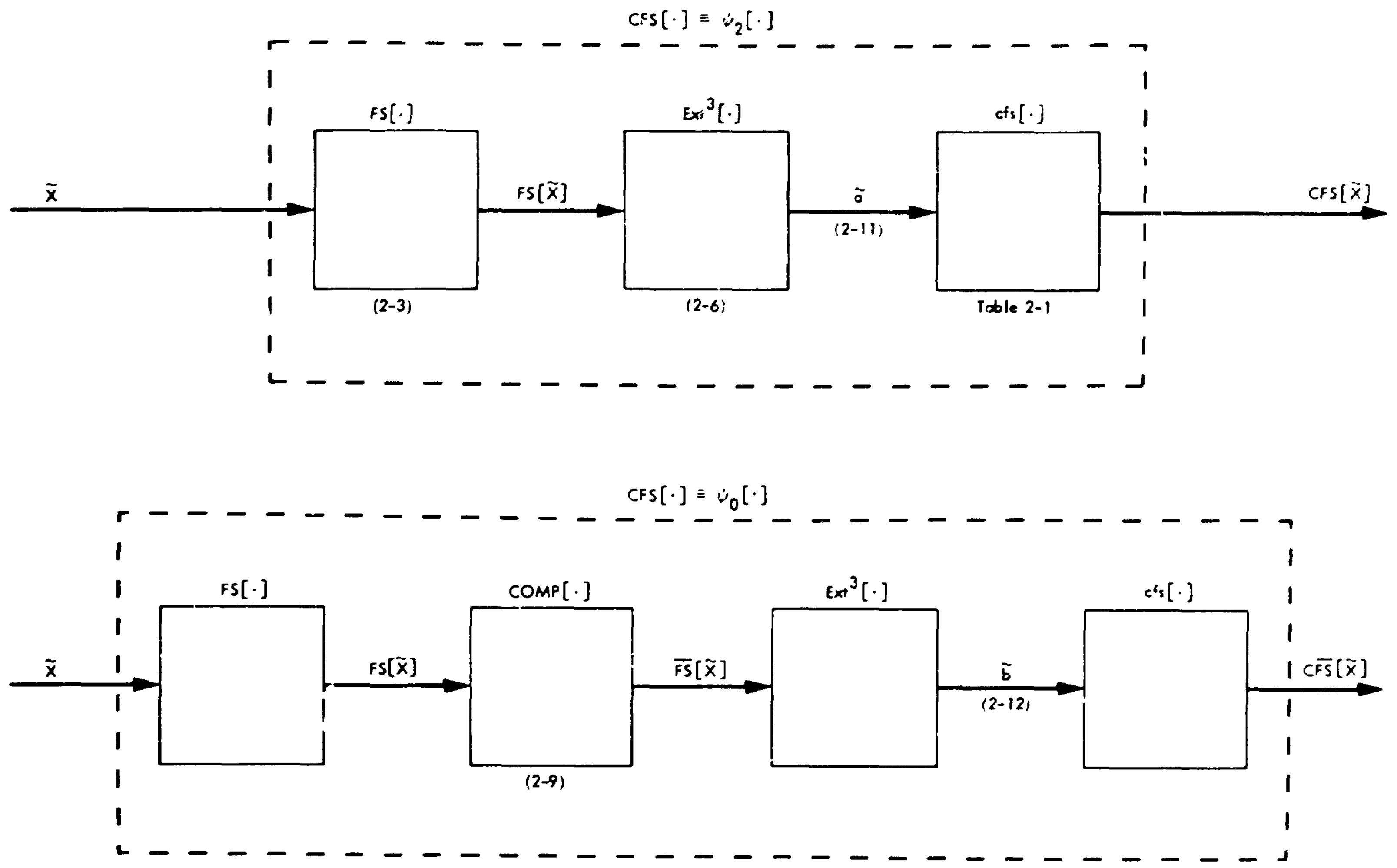


Fig. 2-2. Block diagrams for $CFS[\cdot]$ and $C\overline{FS}[\cdot]$

All zero sequence. In the special case when $\tilde{X}$ is the J sample all zero sequence we have

$$\mathscr{L}\left(CFS[\tilde{X}]\right) = 5\lceil J/3 \rceil \text{ bits} \tag{2-15}$$

and

$$\mathscr{L}\left(C\overline{FS}[\tilde{X}]\right) = \lceil J/3 \rceil \text{ bits} \tag{2-16}$$

Recall that these numbers compare with J bits required by operator FS[·].

Unity Code Operator

We can trivially add a fourth code option to the possibilities by defining

$$\psi_3[\tilde{X}] \tag{2-17}$$

as any fixed length binary representation of $\tilde{X}$. In the simplest case we can take $\psi_3[\tilde{X}]$ as $\tilde{X}$ itself.

$$\psi_3[\tilde{X}] = \tilde{X} \tag{2-18}$$

However, recall that in many applications the $\tilde{X}$ sequence we are coding is the result of reversible preprocessing operations (see Fig. 1-1). The function of such operations is to remove correlation and by relabeling, produce a symbol stream for which the desired probability ordering in (1-8) is well approximated. However, these operations may also effectively increase the alphabet size. For example, taking differences between adjacent samples increases the number of possible sample values by two. Thus a direct fixed length binary representation of a preprocessed $\tilde{X}$ sequence may actually require more bits than a direct fixed length representation before preprocessing. In this case it would be more advantageous to interpret $\psi_3[\tilde{X}]$ as a fixed length binary representation before reversible preprocessing.

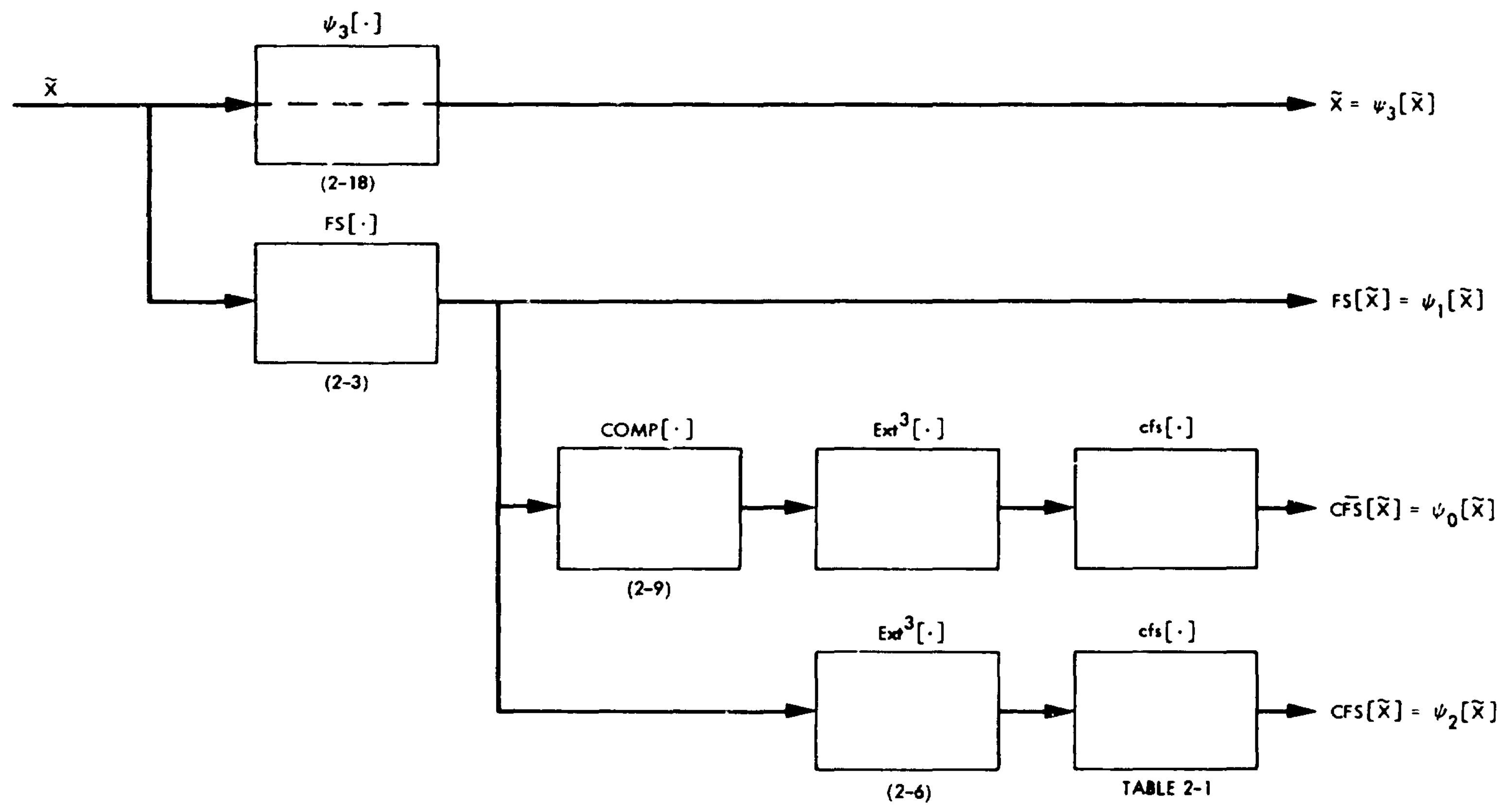
Keeping in mind this more general interpretation of $\psi_3[\cdot]$ we will for the most part assume the special case in (2-18).

<u>Average Performance</u>

A block diagram showing the four code operators $\psi_i[\cdot]$ discussed thus far are shown in Fig. 2-3. Measured average performance for these operators is shown in Fig. 2-4. The graph for $FS[\cdot] = \psi_1[\cdot]$ has been transferred from Fig. 2-1. Again, performance has been measured over spans of K samples much greater than the J sample length of $\tilde{X}$. Other than making q or J too small to be meaningful, the values of these parameters has little influence on the location of these curves. However, an input alphabet size of $q=2^5$ was chosen to show the fixed position of the $\psi_3[\tilde{X}] = \tilde{X}$ curve.

The three curves for $C\overline{FS}[\cdot]$, $FS[\cdot]$ and $CFS[\cdot]$ are almost a perfect match in the sense that when one starts performing poorly (away from the 45° entropy line) another starts performing well. This should not be surprising since $C\overline{FS}[\cdot]$ and $CFS[\cdot]$ obtain improvements over $FS[\cdot]$ by coding redundancy left in 3-tuples of $FS[\tilde{X}]$. If $FS[\cdot]$ is performing close to $H(\overline{P})$ then there can be little redundancy left. Otherwise $C\overline{FS}[\cdot]$ or $CFS[\cdot]$ would be performing under the entropy. For individual fixed coding algorithms this is impossible.

The main observation is that these three code operators offer options which can provide efficient average performance when data entropies lie, roughly, in the range of 0.7 bits/sample to 4 bits/sample. Operator definitions do not depend on alphabet size, q. The additional unity operator, $\psi_3[\cdot]$, is available free of charge in most digital systems. Thus we should be able to provide a simple yet efficient adaptive coder (universal coder) by selecting between these four options.



$\tilde{X}$
$\psi_3[\cdot]$
(2-18)
$\tilde{X} = \psi_3[\tilde{X}]$
$FS[\cdot]$
(2-3)
$FS[\tilde{X}] = \psi_1[\tilde{X}]$
$COMP[\cdot]$
(2-9)
$Ext^3[\cdot]$
$cfs[\cdot]$
$C\bar{F}S[\tilde{X}] = \psi_0[\tilde{X}]$
$Ext^3[\cdot]$
(2-6)
$cfs[\cdot]$
TABLE 2-1
$CFS[\tilde{X}] = \psi_2[\tilde{X}]$

Fig. 2-3. Four Code Options: Block Diagram

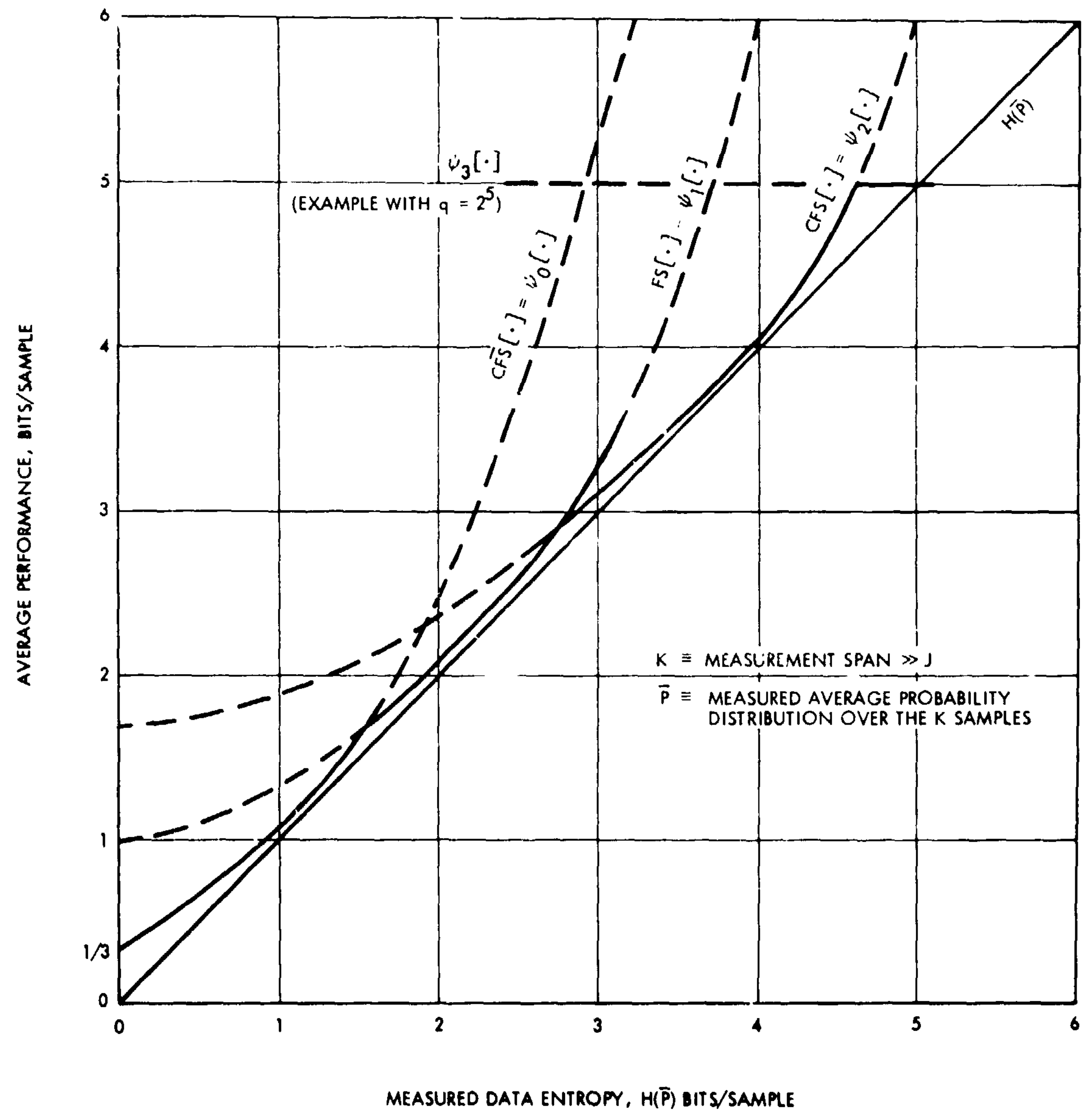


Fig. 2-4. Average Performance, $C\overline{FS}[\cdot]$, $FS[\cdot]$, $CFS[\cdot]$, $\psi_3[\cdot]$

ADAPTIVE CODER

We are now ready to define an adaptive coder which selects a code option from the four choices $\psi_0[\cdot] = C\overline{FS}[\cdot]$, $\psi_1[\cdot] = FS[\cdot]$, $\psi_2[\cdot] = CFS[\cdot]$ and $\psi_3[\cdot]$. Letting ID be the selected code operator for a given J sample input block a "Basic Compressor" output takes the form

$$\psi_4[\tilde{X}] = BC[\tilde{X}] \equiv ID * \psi_{ID}[\tilde{X}] \qquad (2\text{-}19)$$

where the concatenated ID is assumed to be a 2-bit binary number whereas, as a subscript to ψ it takes on the values 0, 1, 2 or 3.

Observe that ID is really a function of $\tilde{X}$ which partitions the space of all input sequences into four decision regions and takes on the corresponding values 0, 1, 2 or 3. (Carrying the full ID$[\tilde{X}]$ would obviously cause notational problems). It remains to specify this decision rule to complete the definition of $\psi_4[\cdot] = BC[\cdot]$ in (2-19).

Optimum Decision

The most straightforward, and in fact optimum, selection procedure is to simply choose the code operator output sequence which is the shortest. While this might be considered brute force, the simplicity of the code options makes this approach quite feasible in many applications. We will provide a simpler procedure later. The optimum decision rule can be stated simply as

Choose ID such that

$$\mathscr{L}\left(\psi_{ID}[\tilde{X}]\right) = \min_j \left\{\mathscr{L}\left(\psi_j[\tilde{X}]\right)\right\} \qquad (2\text{-}20)$$

The Basic Compressor code operator would then require

$$\mathscr{S}\Big(\text{BC}[\tilde{X}]\Big)/J = 2/J + \mathscr{S}\Big(\psi_{1D}[\tilde{X}]\Big)/J \qquad (2\text{-}21)$$

bits/sample to code an input data block $\tilde{X}$, where the second part is assumed to be the smallest of four possibilities.

The overhead cost in identifying the code choice is

$$2/J \ \text{bits/sample} \qquad (2\text{-}22)$$

which could obviously be diminished to insignificance by increasing J. However, other considerations guide the choice of block size J.

A measured average of the second term in (2-21) will tend to decrease as J is decreased because of the ability to switch codes (adapt) more frequently. This effect will more than compensate for increasing overhead until eventually overhead dominates. Thus there should be a best block size. Runs on various forms of data by the author and Spencer and May[10] suggests that this best block size lies in the range of 16 to 25.† The main observation is that J is not a critical performance parameter and can be chosen primarily for implementation considerations. We will emphasize a block size of 16 for these reasons. Later we will have reason to consider variable block sizes.

Average Performance

A graph of the measured average performance of Basic Compressor operator $\psi_4[\cdot] = \text{BC}[\cdot]$ of (2-19) using either the optimum decision criterion in (2-20) or a simplified rule to be introduced later is shown in Fig. 2-5. As in earlier graphs $H(\overline{P})$ is the average measured data entropy over a span of K

†Test cases originated from image data.

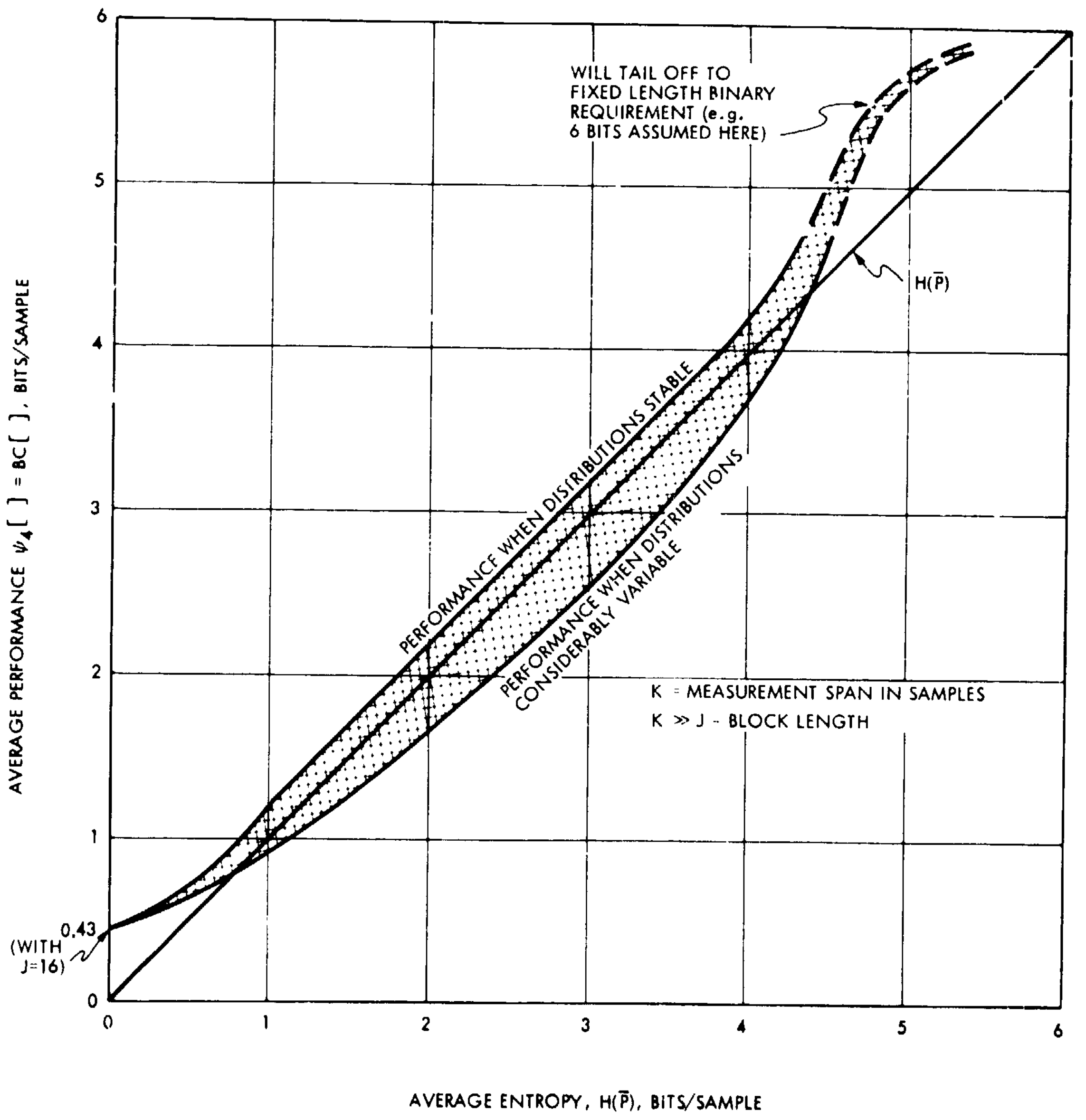


Fig. 2-5. Typical Average Performance of Basic Compressor $\psi_4[\cdot] = BC[\cdot]$

input samples, where $K \gg J$. A range of possible results is shown within the crosshatched area, depending on the variability of data statistics over the measurement span of K samples. When distributions are stable then average performance slightly above $H(\overline{P})$ is typical (e.g., 0.25 bits/sample) throughout the range of $0.7 \leq H(\overline{P}) \leq 4$. However, when data distributions are quite variable over the K samples then performance considerably under $H(\overline{P})$ is possible (we are assuming that probability ordering in (1-8) remains well approximated as distributions vary).

Comments. Note again that J is not a critical parameter. The rough performance description in Fig. 2-5 would generally be unaffected provided J is not so small that overhead becomes dominant or so large that the advantage of adapting disappears. If alphabet size q is very small then a performance description up to 4 bits/sample is not meaningful since a fixed length binary representation, $\psi_3[\cdot]$, can do better. However, Hilbert[11] has obtained good results using the Basic Compressor on processed data which has small q (e.g., $q = 3, 4, 5, \cdots$).

The main point should not be missed: the Basic Compressor can be expected to give efficient performance over a wide range of data entropies with no prior knowledge of distribution $\tilde{P}$ (ordering in (1-8) excluded). In later chapters we will extend efficient performance first to higher data entropies and subsequently to very low entropies near zero. In these cases and others the code operators that result are essentially generated by preprocessing data into forms which can make effective use of the Basic Compressor.

USEFUL BOUNDS AND ESTIMATES[†]

There are some situations where it is desirable to minimize the computation and memory requirements needed to make Basic Compressor decisions and to estimate performance. The relationships between $\psi_0[\cdot]$, $\psi_1[\cdot]$ and $\psi_2[\cdot]$ may be used to accomplish this yielding the functions of input $\tilde{X}$ γ_0, γ_1, γ_2 and γ_3 where[††]

$$\gamma_0(\tilde{X}) \equiv \left\lceil \frac{F}{3} \right\rceil + 2(F - J) \geq \mathscr{L}\left(\psi_0[\tilde{X}]\right) \tag{2-23}$$

$$\gamma_1(\tilde{X}) \equiv F = \mathscr{L}\left(\psi_1[\tilde{X}]\right) \text{ (See Eq. 2-4)} \tag{2-24}$$

$$\gamma_2(\tilde{X}) \equiv \left\lceil \frac{F}{3} \right\rceil + 2J \geq \mathscr{L}\left(\psi_2[\tilde{X}]\right) \tag{2-25}$$

and

$$\gamma_3(\tilde{X}) \equiv \text{constant} \tag{2-26}$$

These are <u>all simple functions of F</u> which is itself just the sum of data samples plus the Basic Compressor block length (2-4).

† $\lceil \alpha \rceil$ means the smallest integer greater than or equal to α.

†† The functions γ_1 and γ_3 are trivial and γ_0 and γ_2 are easily derived. Suppose $\overline{FS}$ is a string of all zeroes, then by Table 2-1 each 3-tuple is coded into one bit so that $\mathscr{L}\left(\psi_0[\tilde{X}]\right) = \left\lceil \frac{F}{3} \right\rceil$. Now start changing zeroes of $\overline{FS}$ into ones. Each change will increment $\mathscr{L}\left(\psi_0[\tilde{X}]\right)$ by at most 2 bits. The increase will be less than this only if an all ones 3-tuple is created. Since $\left\lceil \frac{F}{3} \right\rceil$ is the number of 3-tuples and F-J the number of ones in $\overline{FS}$ we get $\mathscr{L}\left(\psi_0[\tilde{X}]\right) \leq \left\lceil \frac{F}{3} \right\rceil + 2(F-J)$. Using the same argument on FS, using J in place of (F-J), yields γ_2 in (2-25).

The relationship of the $\gamma_i(\tilde{X})$ is shown more clearly by the straightline approximation in Fig. 2-6. The lower envelope of these curves, shown with heavy lines, is essentially the simplified decision rule we are seeking.

Simplified Decision Rule

Instead of comparing the actual bit counts for the four options as in (2-20) we could instead use the following rule: Since the $\gamma_j(\cdot)$ are really functions of F we choose ID such that

$$\gamma_{ID}(F) = \min_{J} \left\{ \gamma_j(\tilde{X}) \right\} \tag{2-27}$$

This rule simplifies further to the rule in Table 2-2 where we assume that fixed length coding of $\tilde{X}$ by $\psi_3[\cdot]$ requires m bits, and $m > 3J$.

The expression $3(m-2J)$ is the approximate result of solving $\gamma_2(\tilde{X})$ in (2-25) for the F which gives $\gamma_2(\tilde{X}) \geq m$.

Using these rules the Basic Compressor operator, $\psi_4[\cdot]$ in (2-19), could require no more than $2 + \gamma_{ID}(F)$ bits to code $\tilde{X}$. But then certainly a coder which used the optimum criteria in (2-20) could require no more than $\gamma_{ID}(F)$ bits either. Thus $\gamma_{ID}(F)$ may be used to bound the performance of either system. In particular, we have

$$\mathscr{L}\left(\psi_4[\tilde{X}]\right) = \mathscr{L}\left(BC[\tilde{X}]\right) \leq \gamma_4(\tilde{X}) \equiv 2 + \gamma_{ID}(F) \tag{2-28}$$

We can now see that $\gamma_{ID}(F)$ is the lower envelope to the curves in Fig. 2-5.

Note that there are two critical points on the graph. Below $F \approx \frac{3}{2}J$ $\psi_0[\cdot] = C\overline{FS}[\cdot]$ will always perform better than $FS[\cdot] = \psi_1[\cdot]$ while above $F = 3J$ (and less than $3(m-2J)$) $\psi_2[\cdot] = CFS[\cdot]$ will always perform better than $FS[\cdot]$. In these cases the simplified decision criterion in (2-29) would make the same decisions as the optimum criterion in (2-19). Between these two operating points there

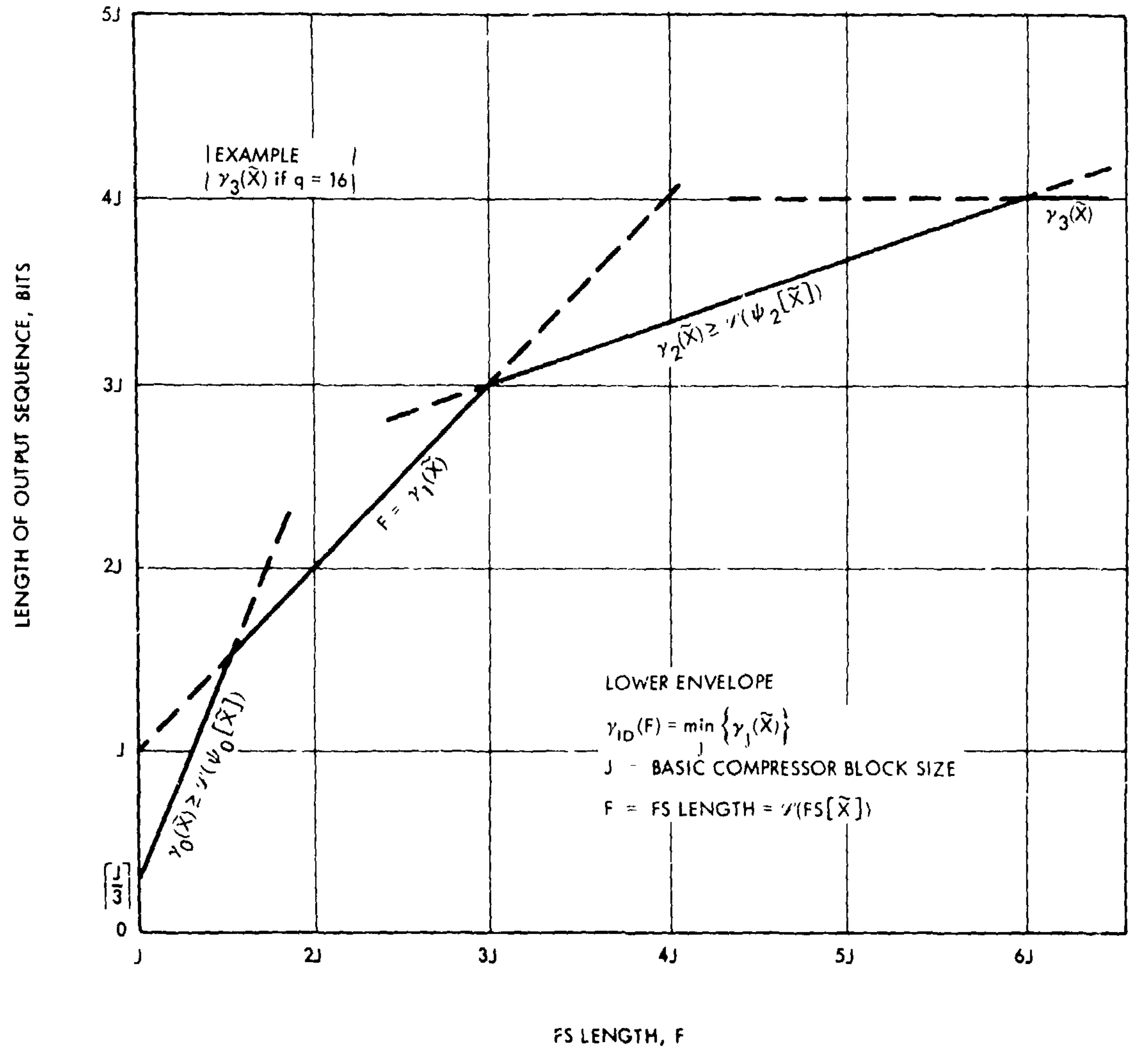


Fig. 2-6. Plot of $\gamma_{ID}(F)$ vs F

Table 2-2. Another Form of Simplified Decision Rule.[†]

Operator Decision	Condition on Fundamental Sequence Length
$\psi_0[\cdot]$	$F \le 3\lfloor J/2 \rfloor$
$\psi_1[\cdot]$	$3\lfloor J/2 \rfloor < F \le 3J$
$\psi_2[\cdot]$	$3J < F < 3(m-2J)$
$\psi_3[\cdot]$	$F \ge 3(m-2J)$

is some possibility that either $\psi_0[\cdot]$ or $\psi_2[\cdot]$ might perform better on a given $\tilde{X}$ than operator $FS[\cdot] = \psi_1[\cdot]$ which would be chosen by the simplified rule. But experience on real data has shown that the vast majority of the time $FS[\cdot]$ is indeed the best choice. There is also a remote possibility that $\psi_2[\cdot]$ could perform better than $\psi_3[\cdot]$ when $F > 3(m-2J)$. Again, experience indicates that $\psi_2[\cdot]$ is usually the best choice. Thus the difference in performance between a Basic Compressor using the optimal decision rule and one which uses the simplified rule seems to be insignificant from a statistical point of view.

$\underline{\gamma_{ID}(F)\text{ as an Estimate}}$

The measured performance of the Basic Compressor operator $\psi_4[\cdot]$ shown in Fig. 2-5 is a plot of the long term average of Eq. (2-21). We have just noted that the choice of decision rule would have a negligible impact on these results. In addition we have the following useful observation that a long term average of

$$\frac{2}{J} + \frac{\gamma_{ID}(F)}{J} \tag{2-29}$$

† $\lfloor \alpha \rfloor$ means the greatest integer less than or equal to α.

will tend to yield very nearly the same results. That is in a statistical sense we have, for either decision rule[†]

$$E\{\mathscr{V}(\psi_4[\tilde{X}])\} \leq 2 + E\{\gamma_{ID}(F)\} \qquad (2\text{-}30)$$

<u>and</u>

$$E\{\mathscr{V}(\psi_4[\tilde{X}])\} \approx 2 + E\{\gamma_{ID}(F)\} \qquad (2\text{-}31)$$

Thus the bound in (2-28) is statistically tight.

This is an extremely useful result. In simple terms it means that Basic Compressor performance can be estimated (and bounded) quite closely by adding up input samples (Eq. 2-4) and then computing $\gamma_{ID}(F)$ using the simple expressions in (2-23) through (2-26). The actual coder need not be implemented to determine performance.

<u>Looser Bounds</u>

Note from Fig. 2-6 that the function $\gamma_{ID}(F)$ is convex $\cap$. Thus we have

$$E\{\gamma_{ID}(F)\} \leq \gamma_{ID}(\bar{F}) \qquad (2\text{-}32)$$

where

$$\bar{F} = E\{F\} \qquad (2\text{-}33)$$

Direct substitution in (2-30) yields a looser bound. Whereas $E\{\gamma_{ID}(F)\}$ takes into account the ability to switch codes each J samples, $E\{\gamma_{ID}(\bar{F})\}$ assumes only the ability to pick the best of four codes once.

[†]$E\{\cdot\}$ denotes expectation.

A practical application of (2-32) to bounding the performance of the Basic Compressor over a long sequence $\tilde{Y} = y_1 y_2 \cdots y_K$ where $K \gg J$, is quite straightforward. Determine $\overline{F}$ as[†]

$$\overline{F} = \frac{J}{K}\left(K + \sum_{i=1}^{K} y_i\right) \tag{2-34}$$

and then bound average per sample performance by

$$\frac{2}{J} + \frac{\gamma_{ID}(\overline{F})}{J} \tag{2-35}$$

<u>Variable J.</u> In all cases so far we have assumed that block size J was fixed. However, we will later have reason to consider variable block sizes. Here we derive a useful result for later use. First supplement the function $\gamma_{ID}(F)$ with J as in $\gamma_{ID}(F, J)$. If we fix F at F^* and plot $\gamma_{ID}(F^*, J)$ as a function of J we would find it is also convex $\cap$. Then we have

$$E\left\{\gamma_{ID}(F, J) \mid F = F^*\right\} \le \gamma_{ID}(F^*, \overline{J}) \tag{2-36}$$

where

$$\overline{J} = E\{J\} \tag{2-37}$$

But then if both F and J vary

$$\begin{aligned} E\{\gamma_{ID}(F, J)\} &= E\left\{E\{\gamma_{ID}(F, J) \mid J = J^*\right\} \\ &\le E\left\{\gamma_{ID}(\overline{F}, J)\right\} && (2\text{-}38) \\ &\le \gamma_{ID}(\overline{F}, \overline{J}) && (2\text{-}39) \end{aligned}$$

[†] K is assumed to be a multiple of J.

We recognize (2-38) as the previous result in (2-32). Thus (2-39) provides a slightly weaker bound to Basic Compressor performance.

BLOCK DIAGRAM

A block diagram of the Basic Compressor using the simplified decision rule is shown in Fig. 2-7.

EXAMPLES

Example 1

Let the input sequence block size be J=16 and the alphabet size q=16. Then suppose

$$\tilde{X}_1 = 0, 0\ 0, 0, 0, 4, 0, 0, 0, 4, 0, 9\ 0, 0, 1, 0 \qquad (2\text{-}40)$$

Fixed length, $\psi_3[\tilde{X}_1]$. A standard binary representation of $\tilde{X}_1$ is easily obtained as a sequence of sixteen 4-bit codewords. Then, obviously

$$\mathscr{L}\left(\psi_3[\tilde{X}_1]\right) = 64 \qquad (2\text{-}41)$$

Fundamental Sequence, $\psi_1[\tilde{X}_1]$. A fundamental sequence is obtained using (2-1) and (2-3) yielding

$$FS[\tilde{X}_1] = \psi_1[\tilde{X}_1] = 1, 1, 1, 1, 1, 00001, 1, 1, 1, 00001, 1, 0000000001, 1, 1, 01, 1$$

(2-42)

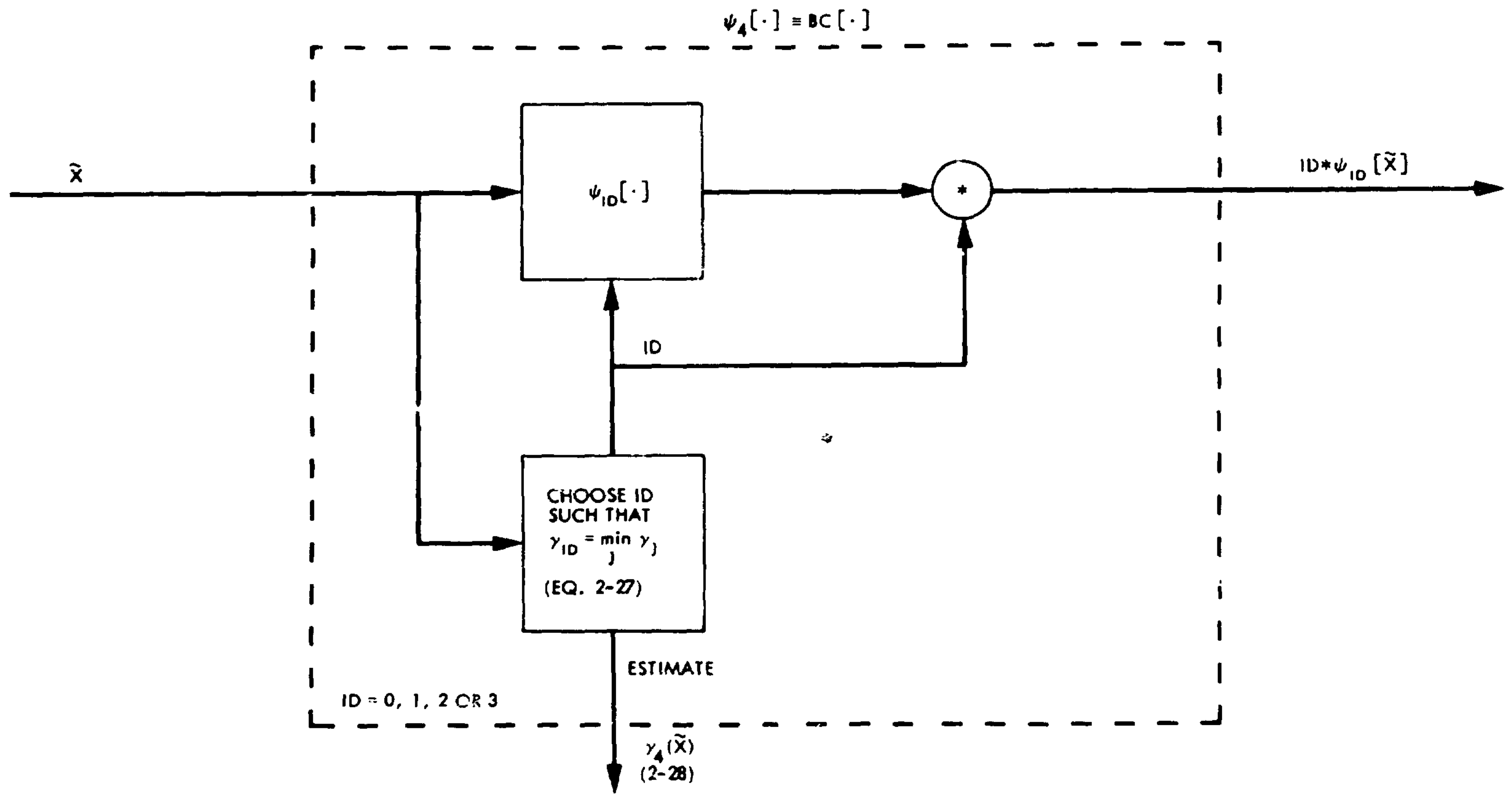

$\psi_4[\cdot] \equiv BC[\cdot]$
$\tilde{X}$
$\psi_{ID}[\cdot]$
$*$
$ID * \psi_{ID}[\tilde{X}]$
ID
CHOOSE ID SUCH THAT $\gamma_{ID} = \min_j \gamma_j$ (EQ. 2-27)
ESTIMATE
ID = 0, 1, 2 OR 3
$\gamma_4(\tilde{X})$ (2-28)

Fig. 2-7. Basic Compressor, $\psi_4[\cdot]$.

where we have separated individual codewords by commas. The length of this "FS" sequence can be obtained by counting or by using (2-4)

$$F = \mathcal{L}\left(\psi_1[\tilde{X}_1]\right) = J + \sum_{i=1}^{J} x_i = 34 \tag{2-43}$$

$\overline{FS}[\tilde{X}_1]$ is obtained from (2-42) by complementing each bit. Sequences $\tilde{a}$ and $\tilde{b}$ in (2-11) and (2-12) are obtained by grouping the bits of $FS[\tilde{X}_1]$ and $\overline{FS}[\tilde{X}_1]$ into 3-tuples and adding enough dummy zeroes at the end to complete the last 3-tuple.

$$\begin{aligned}\tilde{a} &= \mathrm{Ext}^3\left[FS[\tilde{X}_1]\right] \\ &= 111, 110, 000, 111, 100, 001, 100, 000, 000, 011, 101, 100\end{aligned} \tag{2-44}$$

$$\begin{aligned}\tilde{b} &= \mathrm{Ext}^3\left[\overline{FS}[\tilde{X}_1]\right] \\ &= 000, 001, 111, 000, 011, 110, 011, 111, 111, 100, 010, 000\end{aligned} \tag{2-45}$$

Now applying the code cfs[$\cdot$] in Table 2-1 to sequences $\tilde{a}$ and $\tilde{b}$ as in (2-13) and (2-14) respectively we get

$$\begin{aligned}\psi_2[\tilde{X}_1] &= CFS[\tilde{X}_1] \\ &= 11111, 11110, 0, 11111, 110, 100, 110, 0, 0, 11100, 11101, 110\end{aligned} \tag{2-46}$$

and

$$\psi_0[\tilde{X}_1] = C\overline{FS}[\tilde{X}_1]$$

$$= 0, 100, 11111, 0, 11100, 11110, 11100, 11111, 11111, 110, 101, 0 \quad (2\text{-}47)$$

By adding bits in (4-46) and (4-47) or by adding codeword lengths when using Table 2-1 we get

$$\begin{aligned} \mathcal{L}\left(\psi_2[\tilde{X}_1]\right) &= 40 \\ \mathcal{L}\left(\psi_0[\tilde{X}_1]\right) &= 42 \end{aligned} \quad (2\text{-}48)$$

Thus the optimum decision criterion of (2-20) would select $\psi_1[\tilde{X}_1] = FS[\tilde{X}_1]$ as the coded output sequence to use. $\psi_1[\tilde{X}_1]$ would be preceded by a two-bit identifier for a total of 36 bits.

Using the simplified rule in (2-27) would have yielded the same results.

Example 2

As another example take $J = 20$, $q \geq 16$ and

$$\tilde{X}_2 = 1, 0, 1, 1, 5, 3, 0, 1, 3, 7, 1, 2, 2, 0, 1, 2, 2, 2, 8, 6 \quad (2\text{-}49)$$

Then generating an FS and breaking it into 3-tuples as before we get

$$\begin{aligned} \tilde{a} = \text{Ext}^3\left[FS[\tilde{X}_2]\right] = {} & 011, 010, 100, 000, 100, 011, 010, \\ & 001, 000, 000, 010, 100, 100, 110, \\ & 100, 100, 100, 100, 000, 000, 100, \\ & 000, 010 \end{aligned} \quad (2\text{-}50)$$

Again using Table 2-1 we could generate $\psi_2[\tilde{X}_2] = CFS[\tilde{X}_2]$. Instead we put down the corresponding codeword lengths, $\mathcal{L}\left(cfs[x_i]\right)$, the sum of which equals $\mathcal{L}\left(CFS[\tilde{X}_2]\right)$

$$5, 3, 3, 1, 3, 5, 3, 3, 1, 1, 3, 3, 3, 5, 3, 3, 3, 3, 1, 1, 3, 1, 3 \tag{2-51}$$

From (2-50) and (2-51)

$$F = \mathcal{L}\left(FS[\tilde{X}_2]\right) = 68 \tag{2-52}$$

and

$$\mathcal{L}\left(CFS[\tilde{X}_2]\right) = 63 \tag{2-53}$$

$\mathcal{L}\left(\psi_3[\tilde{X}_2]\right) \geq 80$ and $\mathcal{L}\left(\psi_0[\tilde{X}_2]\right) > 68$ so that the optimum decision is $\psi_2[\cdot]$.

Simplified test. The same decision results from the use of Table 2-2 where

$$60 < F < 120 \Rightarrow \psi_2[\cdot] \tag{2-54}$$

Estimate. Using $Y_2(\tilde{X}_2)$ in (2-25) we have

$$\psi_2(\tilde{X}_2) = \lceil F/3 \rceil + 2J = 23 + 40 = 63 \tag{2-55}$$

Using the result in (2-55) we see that the bound in (2-25) is achieved, $\mathcal{L}\left(\psi_2[\tilde{X}_2]\right) = Y_2(\tilde{X}_2)$.

Example 3

Now take J=20 again but suppose $\tilde{X}_3$ is a sequence of zeroes

$$\tilde{X}_3 = 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 \tag{2-56}$$

Using (2-4), F=20 and

$$\gamma_{ID}(F) = \gamma_0(\tilde{X}_3) = \left\lceil \frac{20}{3} \right\rceil = 7 \qquad (2\text{-}57)$$

Actually going through the coding procedures for $\psi_0[\cdot]$ yields the seven bit binary sequence

$$\gamma_0[\tilde{X}_3] = 0000000 \qquad (2\text{-}58)$$

Adding two bits for identification of the code option gives from (2-19)

$$\psi_4[\tilde{X}_3] = BC[\tilde{X}_3] = 00{*}0000000 \qquad (2\text{-}59)$$

Example 4

Now suppose we have the sequence

$$\tilde{Y} = \tilde{X}_2 * \tilde{X}_3 \qquad (2\text{-}60)$$

where $\tilde{X}_2$ is the sequence of example 2 and $\tilde{X}_3$ is the all zero sequence of example 3.

Letting $\psi_5[\cdot]$ denote the operation of using the Basic Compressor on each $\tilde{X}_i$ separately we have

$$\psi_5[\tilde{Y}] = \psi_4[\tilde{X}_2] * \psi_4[\tilde{X}_3] \qquad (2\text{-}61)$$

Using the previous results we have

$$\mathscr{L}\left(\psi_5[\tilde{Y}]\right) = 4 + 63 + 7 = 74 \text{ bits} \qquad (2\text{-}62)$$

Recall that either an exact bit count or use of the estimator $\gamma_{ID}(\cdot)$ yields this result.

Now compute the average FS length from (2-54) as

$$\overline{F} = \frac{1}{2}(88) = 44 \tag{2-63}$$

Then using $\overline{F}$ in (2-23) to (2-26), the minimum yields

$$\gamma_{ID}(\overline{F}) = \overline{F} \tag{2-64}$$

Average performance of $\tilde{Y}$ is bounded by (2-35)

$$\frac{2}{20} + \frac{44}{20} = 2.3 \text{ bits/sample} \tag{2-65}$$

This compares with actual average performance derived from (2-62)

$$\frac{74}{40} = 1.85 \text{ bits/sample} \tag{2-66}$$

The reader can check that he would obtain essentially the same result by coding $\tilde{Y}$ directly using the Basic Compressor $\left(\text{i. e.,}\ \psi_4[\tilde{Y}]\right)$. The overhead term, 2/20, in (2-65) would be reduced to 2/40. The key observation is that using $\psi_4[\cdot]$ on all of $\tilde{Y}$ gives up the ability to adjust the coding to variations in data character, which in the case of $\tilde{X}_2$ and $\tilde{X}_3$ are quite extreme. Here, the advantages of adapting are much more significant than the slight increase in overhead.

OTHER OPERATOR DEFINITIONS

Operator $\psi_5[\cdot]$

Let $\tilde{Y}$ be an N sample sequence of samples which is a priori partitioned into η smaller blocks, $\tilde{Y}_i$, so that

$$\tilde{Y} = \tilde{Y}_1 * \tilde{Y}_2 * \ldots * \tilde{Y}_\eta \tag{2-67}$$

composed of J_i samples each where

$$N = \sum_{i=1}^{\eta} J_i \tag{2-68}$$

Then we define the block by block Basic Compressor coding of $\tilde{Y}$ by the operator $\psi_5[\cdot]$ where

$$\psi_5[\tilde{Y}] = \psi_4[\tilde{Y}_1] * \psi_4[\tilde{Y}_2] * \ldots * \psi_4[\tilde{Y}_\eta] \tag{2-69}$$

An example of $\psi_5[\cdot]$ was given in (2-61).

By defining $\psi_5[\cdot]$ we have lumped all the possible lengths of $\tilde{Y}$ and all the possible ways of partitioning $\tilde{Y}$ into the $\tilde{Y}_i$. In most cases N and a specific partitioning would be fixed for a given application. The most obvious and practical situation is when N is a multiple of some fixed Basic Compressor block size J.

Variable N. In Chapters IV and V we introduce an application for which N is a priori unknown for each $\tilde{Y}$ sequence but is available at a decoder for decoding purposes (i.e., it is transmitted separately). Then we define

$$\psi_6[\cdot] \tag{2-70}$$

as a code operator which codes $\tilde{Y}$ by first choosing a preselected block partitioning assigned to N and then applies the corresponding form of $\psi_5[\cdot]$ to $\tilde{Y}$. Whereas there are many possible rules for partitioning, the following seems to be quite suitable and practical:

Partition $\tilde{Y}$ into η blocks, where

$$\eta \doteq \begin{cases} \lfloor N/J \rfloor & \text{for } N \geq J \\ 1 & \text{otherwise} \end{cases} \tag{2-71}$$

and the first η-1 blocks have J samples and the last, N-(η - 1)J samples. That is

$$J_i = \begin{cases} J \text{ for } i < \eta \\ N-(\eta-1)J & \text{otherwise} \end{cases} \tag{2-72}$$

Quite obviously, $\psi_5[\cdot]$ is a special case of $\psi_6[\cdot]$ for which the length of $\tilde{Y}$ is predetermined and fixed.

Performance Bounds and Estimates

The function $\gamma_4(\cdot)$ in (2-28) provides a useful bound and estimate of the performance of Basic Compressor operator $\psi_4[\cdot]$. It is equally desirable, and a simple matter, to assign similar functions to more complex operators as they are developed. That is

$$\mathscr{S}\left(\psi_j[\tilde{Y}]\right) \leq \gamma_j(\tilde{Y}) \text{ and } \mathscr{S}\left(\psi_j[\tilde{Y}]\right) \approx \gamma_j(\tilde{Y}) \tag{2-73}$$

for each $\psi_j[\cdot]$. As an example and a result for future use we have from the developments above, for j = 5 or 6,

$$\mathscr{S}\left(\psi_j[\tilde{Y}]\right) \leq \gamma_j(\tilde{Y}) = \sum_{i=1}^{\eta} \gamma_4(\tilde{Y}_i) \tag{2-74}$$

and can expect that typically

$$\mathscr{S}\left(\psi_j[\tilde{Y}]\right) \approx \gamma_j(\tilde{Y}) \tag{2-75}$$

III. EXTENDING PERFORMANCE TO HIGH DATA ENTROPIES

Using the Basic Compressor to directly code long sequences drawn from average distributions $\overline{P}$ which have entropies greater than 4 bits/sample results in relatively inefficient performance as noted in Fig. 2-5. That is, the curve for average performance will move away from $H(\overline{P})$. This chapter addresses that problem and provides a simple means for achieving efficient performance at the higher entropies.

SPLIT SAMPLES

Let $\tilde{M}^n$ be some sequence of N preprocessed samples for which the probability ordering of (1-8) is satisfied. The symbol n signifies that the standard binary representation for $\tilde{M}^n$ requires n bits/sample. Define the "split sample" operator $SS^m[\cdot]$ by

$$SS^m[\tilde{M}^n] \equiv \tilde{M}^m * \tilde{L}^k, \; m+k=n \tag{3-1}$$

where

$$\tilde{L}^k \equiv \begin{cases} \text{N sample sequence consisting of the k least significant} \\ \text{bits of each sample of } \tilde{M}^n \end{cases} \tag{3-2}$$

and

$$\tilde{M}^m \equiv \begin{cases} \text{N sample sequence consisting of the m = n-k most} \\ \text{significant bits of each sample of } \tilde{M}^n \end{cases} \tag{3-3}$$

and $SS^n[\tilde{M}^n] = \tilde{M}^n$.

Clearly $SS^m[\cdot]$ is reversible since each sample of $\tilde{M}^n$ can be reconstructed by combining the corresponding samples of $\tilde{M}^m$ and $\tilde{L}^k$. We can therefore concentrate on the efficient coding of $\tilde{M}^m$ and $\tilde{L}^k$, from which $\tilde{M}^n$ can be retrieved.

A sample block diagram showing $SS^m[\cdot]$ is provided in Fig. 3-1.

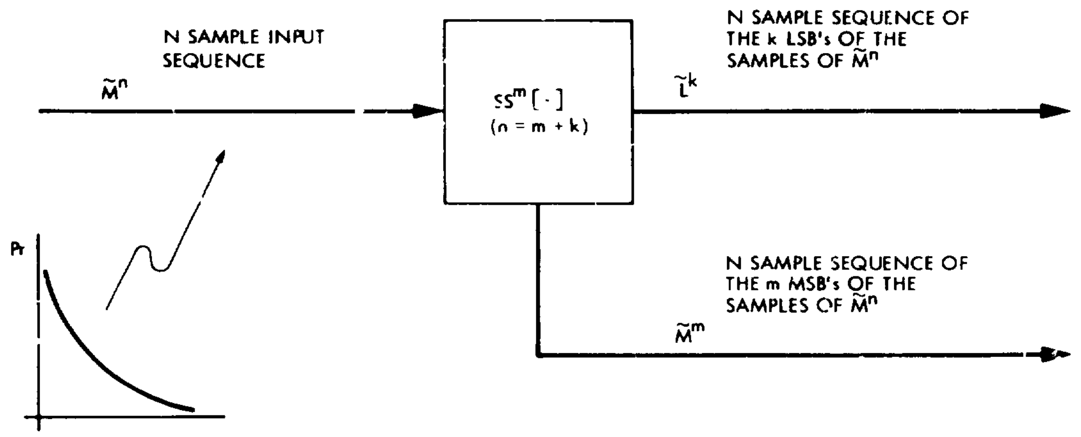

N SAMPLE INPUT SEQUENCE
$\tilde{M}^n$
$SS^m[\cdot]$ $(n = m + k)$
$\tilde{L}^k$
N SAMPLE SEQUENCE OF THE k LSB's OF THE SAMPLES OF $\tilde{M}^n$
$\tilde{M}^m$
N SAMPLE SEQUENCE OF THE m MSB's OF THE SAMPLES OF $\tilde{M}^n$
Pr

Fig. 3-1. Split Sample Operator, $SS^m[\cdot]$

ENTROPY CONSIDERATIONS

Definitions

Let $\overline{P}_m$ be the average distribution of the N m-bit samples from $\tilde{M}^m$.

Let $\beta(m)$ denote the per sample performance of the Basic Compressor operator applied to sequence $\tilde{M}^m$. That is

$$\beta(m) = \frac{\mathcal{T}\left(\psi_\ell[\tilde{M}^m]\right)}{N} \tag{3-4}$$

where ℓ = 5 or 6 (see 2-71 and 2-72).

Observations

If the distribution $\overline{P}_n$ on the original input samples $\tilde{M}^n$ approximates the desired ordering in (1-8) then so does each $\overline{P}_m$ on the m most significant bit samples. In addition as m decreases (fewer significant bits in each sample) the distributions $\overline{P}_m$ become more peaked around zero and the entropies $H(\overline{P}_m)$ decrease. For our purposes here we note from experimental observation that if $H(\overline{P}_m)$ exceeds approximately 4 bits/sample

$$H(\overline{P}_{m-1}) \approx H(\overline{P}_m) - 1 \tag{3-5}$$

That is, one less bit of quantization reduces the sample entropy by approximately one bit. This provides the key to obtaining efficient performance at higher entropies.

Example

Suppose $H(\overline{P}_n) = 5.5$, then we know that a direct application of the Basic Compressor using $\psi_5[\cdot]$ will not produce efficient performance since $H(\overline{P}_n) > 4$. That is, $\beta(n)$ is not close to $H(\overline{P}_n)$.

Applying $\psi_5[\cdot]$ to $\tilde{M}^{n-1}$ would still yield inefficient performance on the n-1 most significant bits since by (3-5) $H(\overline{P}_{n-1}) \approx 4.5 > 4$ (see Fig. 2-5).

However, $H(\overline{P}_{n-2}) \approx 3.5 < 4$ so that we can expect that the Basic Compressor, via operator $\psi_5[\cdot]$, will yield

$$\beta(n-2) \approx H(\overline{P}_{n-2}) \tag{3-6}$$

Then using (3-5) twice we have

$$\beta(n-2) + 2 \approx H(\overline{P}_n) \tag{3-7}$$

This suggests that we can obtain efficient coding of the original input data $\tilde{M}^n$ by coding $\tilde{M}^{n-2}$ with the Basic Compressor and transmitting all the least significant bits, $\tilde{L}^2$, separately.

SPLIT SAMPLE MODES

Definition

We define the set of operators $\psi_7^m[\cdot]$ by

$$\psi_7^m[\tilde{M}^n] = \psi_\ell[\tilde{M}^m] * \tilde{L}^k \quad , \ \ell = 5 \text{ or } 6 \tag{3-8}$$

where $\tilde{M}^m$ and $\tilde{L}^k$ are defined in (3-1) - (3-3).

Block diagram. A block diagram of $\psi_7^m[\cdot]$ is given in Fig. 3-2.

Average Performance

The per sample performance of operator $\psi_7^m[\cdot]$ over the N samples is given as

$$\alpha(m) = \frac{\mathscr{I}\left(\psi_7^m[\tilde{M}^n]\right)}{N} = \beta(m) + k \tag{3-9}$$

Actual measurements of the $\alpha(m)$ are shown in Fig. 3-3 for n=8 bit input samples and N large.

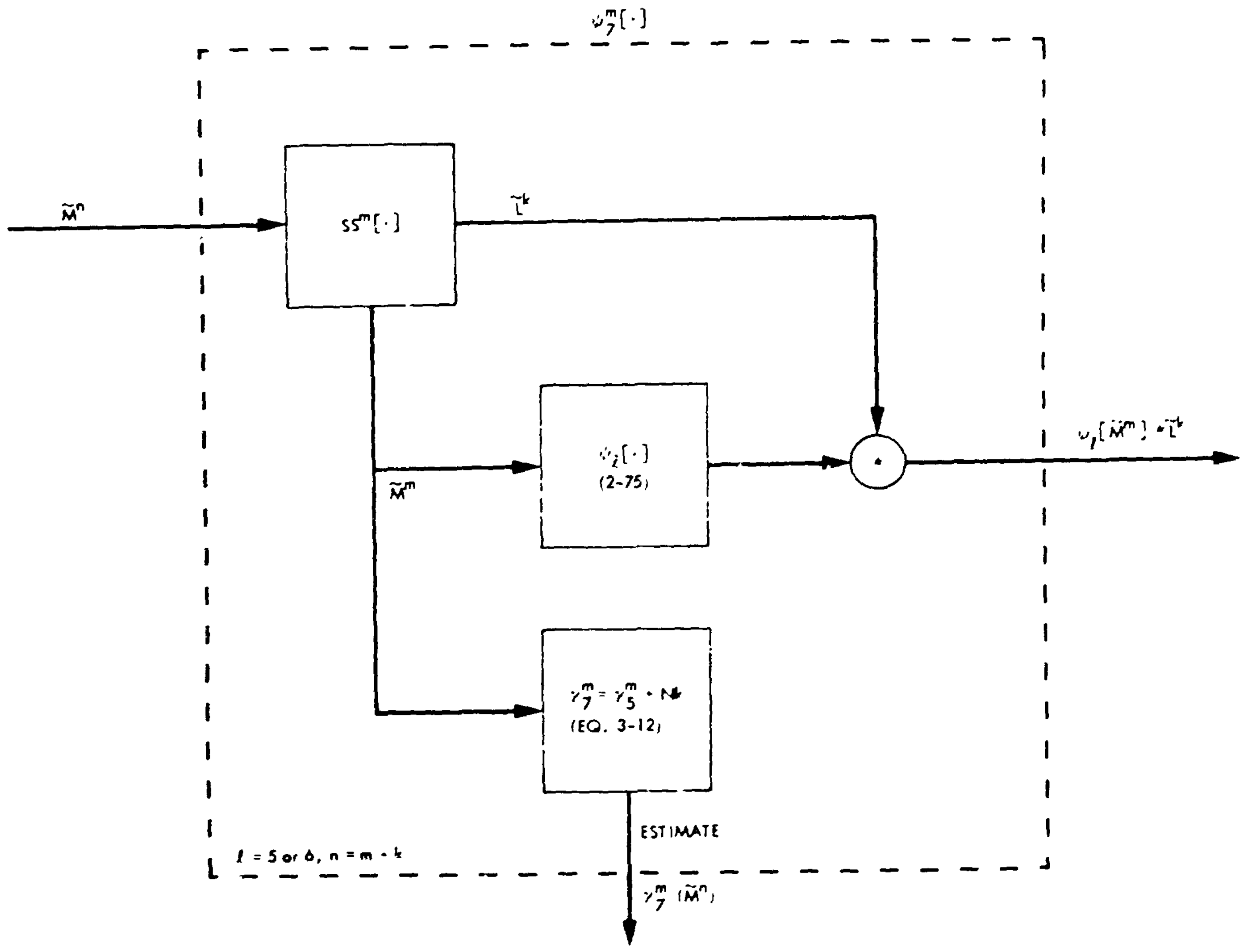


Fig. 3-? Block Diagram Operator $\psi_7^m[\cdot]$

ADAPTIVE OPERATOR, $\psi_8[\cdot]$

We observe from Fig. 3-3 that at least one operator has average performance that remains close to the average entropy line $H(\overline{P}_n)$. Thus we need only select the proper one to use. Define operator $\psi_8[\cdot]$ by

$$\psi_8[\tilde{M}^n] = m' * \psi_7^{m'}[\tilde{M}^n] \tag{3-10}$$

where m' is a selected value of m (the concatenated m' being interpreted as a binary number).

Observe that m' (just like ID in (2-19) is really a function of $\tilde{M}^n$ which partitions the space of all input sequences into decision regions and takes on the possible values of m. It remains to specify this decision rule to complete the definition of $\psi_8[\cdot]$.

Optimum Decision

Using the optimum decision criterion (counting bits) we choose m' such that

$$\mathscr{L}\left(\psi_7^{m'}[\tilde{M}^n]\right) = \min_m \mathscr{L}\left(\psi_7^{m}[\tilde{M}^n]\right) \tag{3-11}$$

Simplified Rule

Proceeding as we did in Chapter II we can bound and approximate the performance of each $\psi_7^m[\tilde{M}^n]$ by using (2-74) and (2-75). We have from (3-8) for $\ell = 5$ or 6

$$\mathscr{L}\left(\psi_7^{m}[\tilde{M}^n]\right) \doteq Y_7^m(\tilde{M}^n) = Y_\ell(\tilde{M}^n) + Nk \tag{3-12}$$

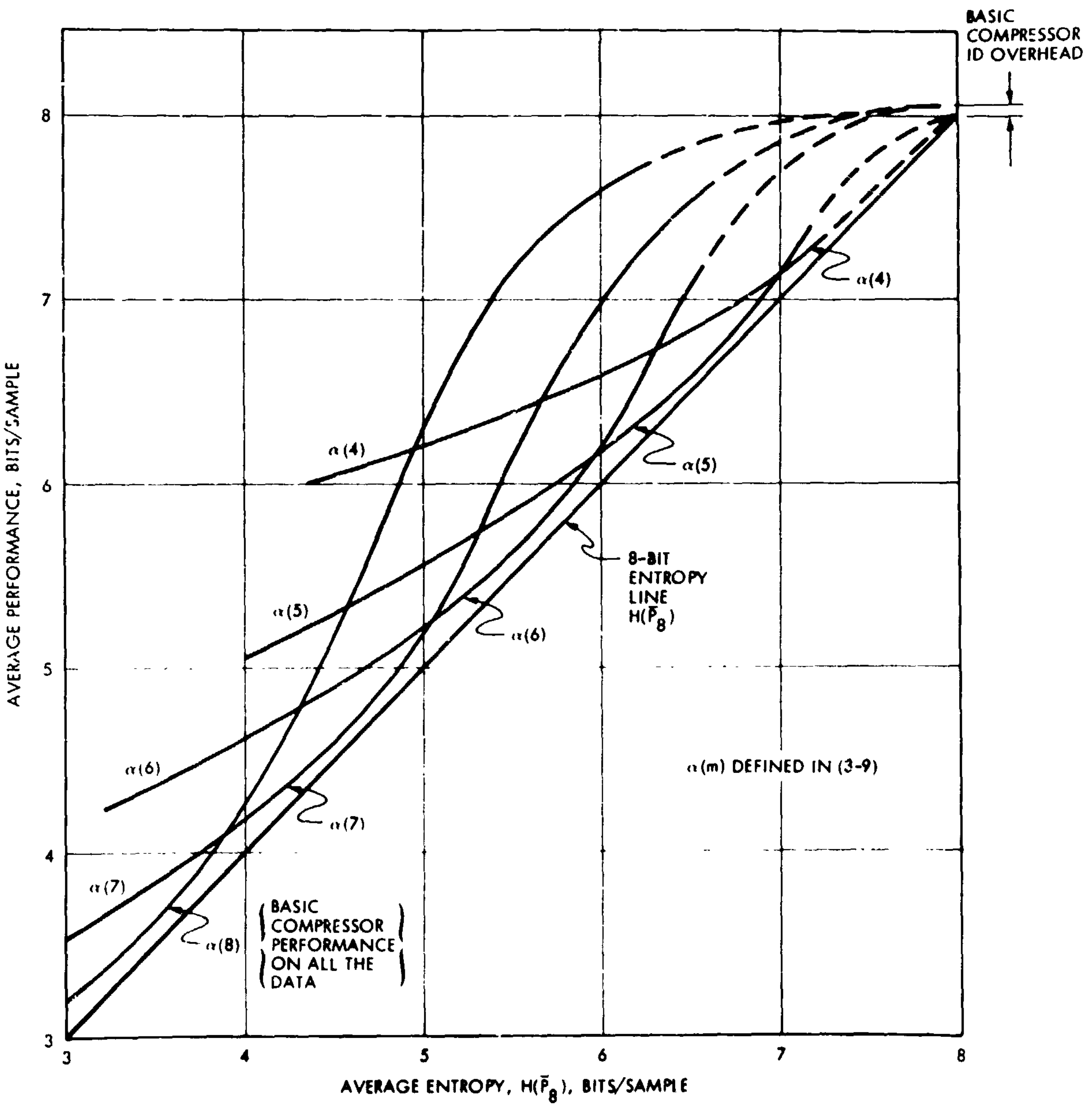


Fig. 3-3. Split-Sample Modes, Individual Average Performance

where $\mathcal{Y}_\ell(\tilde{M}^m)$ is a bound on the performance of operator $\psi_\ell[\cdot]$ applied to the m MSB's of $\tilde{M}^n$ and Nk is a count of all the k LSB's of input sequence $\tilde{M}^n$.[†] By the observations in (2-33) we also have typically

$$\mathcal{I}\left(\psi_7^m[\tilde{M}^n]\right) \approx \mathcal{Y}_7^m(\tilde{M}^n) \tag{3-13}$$

We can now specify a simplified rule for determining the choice of m.

Choose m' such that

$$\mathcal{Y}_7^{m'}(\tilde{M}^n) = \min_m \mathcal{Y}_7^m(\tilde{M}^n) \tag{3-14}$$

Letting $\mathcal{I}(m')$ denote the number of bits required to represent a decision (i.e., m') we have

$$\mathcal{I}\left(\psi_8[\tilde{M}^n]\right) \leq \mathcal{Y}_8(\tilde{M}^n) = \mathcal{I}(m') + \mathcal{Y}_7^{m'}(\tilde{M}^n) \tag{3-15}$$

and where typically

$$\mathcal{I}\left(\psi_8[\tilde{M}^n]\right) \approx \mathcal{Y}_8(\tilde{M}^n) \tag{3-16}$$

Block Diagram, $\psi_8[\cdot]$

A block diagram of operator $\psi_8[\cdot]$ using the simplified decision rule of (3-14) is shown in Fig. 3-3.

Block Size of $\tilde{M}^n$

We assumed that the length of sequence $\tilde{M}^n$ was large to obtain the statistical performance results for individual split sample modes in Fig. 3-3. However,

[†] Note that the length of a fundamental sequence generated by using m MSB's can be related to an FS length generated using fewer MSB's by the sum of the additional split LSB's.

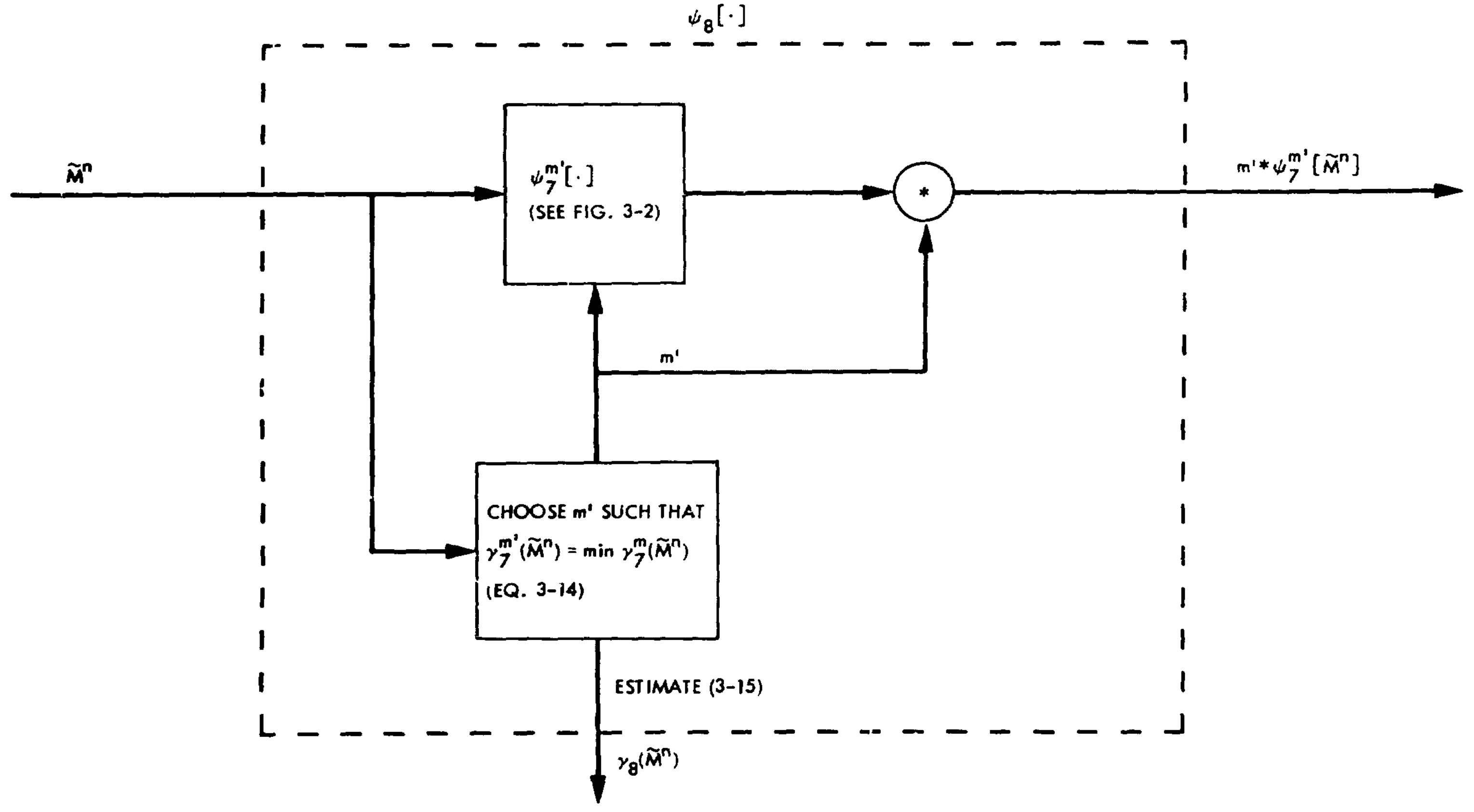

$\psi_8[\cdot]$
$\widetilde{M}^n$
$\psi_7^{m'}[\cdot]$ (SEE FIG. 3-2)
*
$m' * \psi_7^{m'}[\widetilde{M}^n]$
m'
CHOOSE m' SUCH THAT $\gamma_7^{m'}(\widetilde{M}^n) = \min \gamma_7^m(\widetilde{M}^n)$ (EQ. 3-14)
ESTIMATE (3-15)
$\gamma_8(\widetilde{M}^n)$

Fig. 3-4. Block Diagram Operator $\psi_8[\cdot]$.

there is no fundamental reason that N be large. In fact, the tradeoff is much like that required in choosing a good Basic Compressor block size in Chapter II. The smaller N is, the more rapidly operator $\psi_8[\cdot]$ can change to accommodate variations in data statistics. At the same time per sample cost in overhead to identify the selected split sample mode, $\mathscr{Y}(m')$, increases. For most typical applications the choice of N is not critical and one simply chooses N to be some convenient value which is large enough to make $\mathscr{Y}(m')$ negligible. For example if there are four split-sample modes included as options and the data character is slowly varying then a convenient block size of $N = 64$ (e.g., four Basic Compressor blocks using $J = 16$) only adds the negligible overhead of 1/32 bits/sample. In these situations the average per sample performance of operator $\psi_8[\cdot]$, using the optimum or simplified decision rule, will generally ride the lower envelope of the curves in Fig. 3-3 (where the computation of average performance and entropy is assumed to be determined over some $N' \gg N$).

When the variation of data statistics is more rapid, the average per sample performance of $\psi_8[\cdot]$ may be less than the average entropy. In special cases it may be advantageous to choose N as small as the length of one Basic Compressor block J (e.g., 16). That is, the additional adaptivity may more than compensate for the increase in overhead.† In fact additional simplifications turn up once N is constrained to equal J.

† Note that it is a simple matter to investigate the effects of changing parameters (N,J) using estimators $\gamma_8(\cdot)$, $\gamma_7^j(\cdot)$, etc. Complete simulations are replaced primarily by counting operations.

Operator $\psi_8^J[\cdot]$. With N=J we are led to simplify $\psi_8[\cdot]$ to the following

$$\psi_8^J[\tilde{M}^n] = \begin{cases} ss * \psi_4[\tilde{M}^n] & \text{if } ss = 0 \\ \\ ss * m' * \psi_2[\tilde{M}^{m'}] * \tilde{L}^{k'} & \text{if } ss = 1 \end{cases} \qquad (3\text{-}17)$$

The binary term ss indicates whether the input data is to be split or not. If ss=0 the unmodified input data is coded di ctly using the Basic Compressor operator $\psi_4[\cdot]$, whereas ss=1 designates a split with m' indicating what the split is as for $\psi_8[\cdot]$. However, unlike $\psi_8[\cdot]$, the split MSB samples making up $\tilde{M}^{m'}$ are coded using code operator $\psi_2[\cdot]$ = CFS[·] only, rather than the four options of $\psi_4[\cdot]$. This eliminates the need for the two ID bits associated with $\psi_4[\cdot]$. It also simplifies the decision rules since there is only one code option for each split mode.

Based on observed performance using high entropy image data, $\psi_8[\cdot]$ and $\psi_8^J[\cdot]$ perform essentially the same with N = J = 16. This is because a $\psi_4[\cdot]$ decision is almost always to use $\psi_2[\cdot]$ when entropies are high. A slight improvement in average performance can be obtained by including $\psi_1[\cdot]$ = FS[·] as an additional option when ss = 1 in (3-17).

Further Simplification Notes

In some applications decision making can be further simplified with little loss in performance. For example, the correct split-sample modes can be accurately predicted from the results of coding previous blocks in the application described in Refs. 2 and 3. In other cases the computation associated with determining fundamental sequence lengths for each $\tilde{M}^m$ (adding the split MSB and LSB samples) can be reduced. This result is obtained by noting that when entropies are high, a fundamental sequence length associated with the m MSB's of a sequence will (typically) be roughly half that for the m+1 MSB's.

Moving the Preprocessing Operations

All the discussions thus far in this chapter have assumed that input data has been previously preprocessed as described in Fig. 1-1. In some applications the split-sample operations can be performed before this preprocessing with precisely the same average effects. An important example includes the differencing of adjacent image samples as described in Refs. 2, 3 and 10. Although the same performance can be expected, it is more difficult to provide a parallel implementation of the code estimators when there are several split sample modes.

EXAMPLE

The following example should help make the previous discussions seem less abstract. Let

$$\tilde{M}^n = \tilde{Y}_1^n * \tilde{Y}_2^n * \tilde{Y}_3^n \qquad (3\text{-}18)$$

be an $N=52$ sample preprocessed input sequence partitioned into three (Basic Compressor) blocks of $J_1=16$, $J_2=16$ and $J_3=20$ samples respectively where

$$\tilde{Y}_1^n \equiv 3, 5, 1, 0, 2, 1, 2, 2, 7, 10, 10, 22, 14, 7, 0, 14$$

$$\tilde{Y}_2^n \equiv 22, 22, 3, 0, 1, 5, 3, 0, 21, 17, 5, 4, 5, 1, 7, 13$$

$$\tilde{Y}_3^n \equiv 11, 2, 5, 7, 2, 1, 17, 3, 6, 6, 2, 1, 1, 5, 6, 6, 6, 22, 16, 0$$

(3-19)

We assume that the data originates from a source with alphabet size $q = 32$ so that $n = 5$ and we wish to code $\tilde{M}^n$ using operator $\psi_8[\cdot]$, assuming two split-sample modes with $m = n = 5$ and $m = 4$.

Simplified Decision

Referring to Fig. 3-3, we first need to determine which split-sample mode to use by comparing the estimates γ_7^5 and γ_7^4.

m = 5. With $m = 5$ we have from (3-12) $\gamma_7^5(\tilde{M}^5) = \gamma_5(\tilde{M}^5)$ which by (2-74) is simply the sum of the Basic Compressor estimates on each of the input blocks, $\gamma_4(\tilde{Y}_i^5)$. Performing the required computations using the methods of Chapter II we get

$$\gamma_7^5(\tilde{M}^n) = 73 + 82 + 91 = 246 \qquad (3\text{-}20)$$

m = 4. Representing the samples of $\tilde{M}^5$ as five bit numbers and applying split-sample operator $SS^4[\cdot]$ (described in (3-1)-(3-3)) we obtain, with an obvious extension of notation

$$\tilde{M}^4 = \tilde{M}_1^4 * \tilde{M}_2^4 * \tilde{M}_3^4 \;, \quad \tilde{L}^1 = \tilde{L}_1^1 * \tilde{L}_2^1 * \tilde{L}_3^1 \qquad (3\text{-}21)$$

where

$$\begin{aligned} \tilde{M}_1^4 &= 1, 2, 0, 0, 1, 0, 1, 1, 3, 5, 5, 11, 7, 3, 0, 7 \\ \tilde{M}_2^4 &= 11, 11, 1, 0, 0, 2, 1, 0, 10, 8, 2, 2, 2, 0, 3, 6 \\ \tilde{M}_3^4 &= 5, 1, 2, 3, 1, 0, 8, 1, 3, 3, 1, 0, 0, 2, 3, 3, 3, 11, 8, 0 \end{aligned} \qquad (3\text{-}22)$$

and the least significant bits are

$$\tilde{L}_1^1 = 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0$$

$$\tilde{L}_2^1 = 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1$$

$$\tilde{L}_3^1 = 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0$$

(3-23)

Then

$$\gamma_5(\tilde{M}^4) = 55 + 59 + 68 = 182$$

using Basic Compressor operator $\psi_2[\cdot] = \text{CFS}[\cdot]$ on each $\tilde{M}_i^4$. Then by (3-12)

$$\gamma_7^4(\tilde{M}^5) = \gamma_5(\tilde{M}^4) + 52(1) = 234 \qquad (3\text{-}24)$$

Decision. The decision rule in Fig. 3-3 would lead us to select m = 4 since $\gamma_7^4(\tilde{M}^5) < \gamma_7^5(\tilde{M}^5)$. Since there are only two possible split sample modes we have from (3-24) and (3-15)

$$\mathscr{L}\left(\psi_8[\tilde{M}^5]\right) \leq 235 \qquad (3\text{-}25)$$

Actual Coding

The actual coding of $\tilde{M}^n$ takes the form

$$\psi_8[\tilde{M}^n] = m' * \psi_7^{m'}[\tilde{M}^n]$$

$$= m' * \psi_5[\tilde{M}^{m'}] * \tilde{L}^k \qquad (3\text{-}26)$$

By previous results we have decided to use m' = 4 which can be arbitrarily represented with a binary 1. Then (3-26) can be expanded to

$$\psi_8[\tilde{M}^n] = 1 * \psi_4[M_1^4] * \psi_4[\tilde{M}_2^4] * \psi_4[\tilde{M}_3^4] * \tilde{L}^1 \qquad (3\text{-}27)$$

Noting that the individual Basic Compressor decisions leading to $\gamma_5(\tilde{M}^4)$ were $\psi_2[\cdot] = \text{CFS}[\cdot]$ for each block $\tilde{M}_i^4$, (3-27) further breaks down to

$$\psi_8[\tilde{M}^n] = 1 * 10 * \psi_2[\tilde{M}_1^4] * 10 * \psi_2[\tilde{M}_2^4] * 10 * \psi_2[\tilde{M}_3^4] * \tilde{L}^1 \qquad (3\text{-}28)$$

The reader may verify that[†]

$$\psi_2[\tilde{M}_1^4] = 1011110011101111011011000100010000010000010110011001000 \qquad (3\text{-}29)$$

$\psi_2[\tilde{M}_1^4] =$ 10111100111011110110110001000100
000100001011001100100 (3-29)

$\psi_2[\tilde{M}_2^4] =$ 00010000001001110011011101110001 0
0001001001001001101010100 (3-30)

$\psi_2[\tilde{M}_3^4] =$ 01001011011001110000100101100011
101111101011000110101000101001110 0

(3-31)

and that equality is obtained in (3-25)

$$\mathcal{S}\left(\psi_8[\tilde{M}^n]\right) = 235 \qquad (3\text{-}32)$$

[†] Note that successive use of operator $\psi_8^J[\cdot]$ in (3-17) would require fewer bits for this example.

Observation. Note that the actual location of the LSB sequences $\tilde{L}_i^k$ need not be as prescribed in the development of $\psi_8[\cdot]$. For example, they could all be located before the Basic Compressor blocks, or individually after each Basic Compressor block they correspond to. Such variations to $\psi_8[\cdot]$ have no effect on performance but may be a useful implementation consideration.

IV. EXTENDING EFFICIENT PERFORMANCE TO VERY LOW ENTROPIES FOR NON-BINARY SOURCES

As noted from Fig. 2-5 if the variations in data entropy result in values much below 1 bit/sample the efficiency of the Basic Compressor operator $\psi_4[\cdot]$ suffers. This chapter develops a code operator structure for extending good performance over this lower range of $H(\overline{P})$ when $\overline{P}$ of non-binary sources are a priori unknown except for the usual preprocessing assumptions in Chapter I (see Fig. 1-1). A basic requirement in the definition of this structure is the existence of a separate code operator which is capable of providing efficient coding of binary memoryless sources with a priori unknown (and varying) probabilities. Since the latter subject is of general interest by itself it is given a separate treatment in Chapter V which provides the development of a class of such binary code operators. Appropriate substitution of these results into the operator structure of Chapter IV completes the definition of a class of non-binary code operators which will maintain efficient performance when $H(\overline{P})$ is very low.

REWRITING THE ENTROPY EQUATION

Data Model

We can obtain motivation for developing an adaptive code operator by first investigating an idealized data model. Let

$$\tilde{Z} = z_1 z_2 \cdots\cdot z_T \tag{4-1}$$

be a T sample sequence from a discrete memoryless source with known and fixed probability distribution $\tilde{P}$ with the usual symbols $0, 1, 2, \cdots q-1$ and the probability ordering of (1-8) (i.e., the idealized output of preprocessing operations in Fig. 1-1).

Relati nship with $\overline{P}$

As in earlier chapters, $\overline{P}$ denotes a measured average distribution of samples. If these samples are from the ideal memoryless source with distribution $\tilde{P}$ then $\overline{P}$ will equal $\tilde{P}$ if we make the measurement span long enough.

Spiit.ing the Source

When $H(\tilde{P}) = H(\overline{P})$ drops much below 1 bit/sample the sample distributions start taking the form shown in Fig. 4-1.

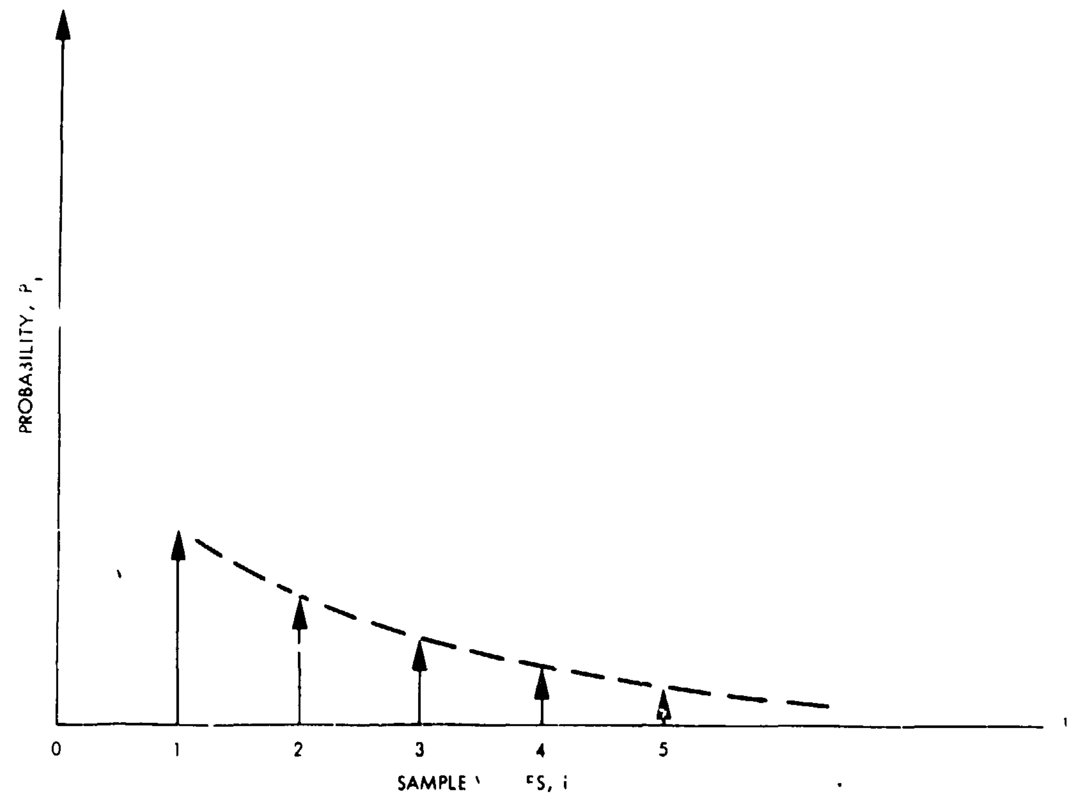


Fig. 4-1. Sample Distributions When $H(\tilde{P}) \rightarrow 0$

The important characteristic to notice is that the probability spike at zero seems out of place from the remainder of the distribution. The coding philosophy we have followed so far suggests that it may be advantageous to treat these distinctly different parts of the distribution separately.

Indeed the source characterized by distributions in Fig. 4-1 can be viewed as a mixture of two sources, one with the symbol zero and the other with the remaining symbols 1, 2, $\cdots$ q-1. More simply we can get the desired motivation by manipulating the basic entropy equation in (1-9) rewritten here as

$$H(\tilde{P}) = -p_0 \log_2 p_0 - \frac{(1-p_0)}{(1-p_0)} \sum_{i \neq 0} p_i \log_2 p_i \tag{4-2}$$

After some manipulation this yields

$$H(\tilde{P}) = H_\beta(p_0) + (1-p_0) H_\theta \tag{4-3}$$

where

$$H_\beta(p_0) = -p_0 \log_2 p_0 - (1-p_0) \log_2 (1-p_0) \tag{4-4}$$

and

$$H_\theta = \sum_{i \neq 0} \left(\frac{p_i}{1-p_0}\right) \log_2 \left(\frac{p_i}{1-p_0}\right) \tag{4-5}$$

We recognize the first term, $H_\beta(p_0)$ as the entropy of a binary memoryless source with the probability of a zero equal to p_0. More specifically, let

$$\tilde{D} \tag{4-6}$$

T be the sample binary sequence which identifies whether a symbol of $\tilde{Z}$ is a zero or not. Then $H_\beta(p_0)$ can be viewed as the minimum average bits/sample required to code $\tilde{D}$.

Since $1 = \Sigma p_i/(1-p_0)$, H_θ is the entropy of a discrete memoryless source with symbols $i = 1, 2, \cdots q-1$ and probabilities $p_i/(1-p_0)$. The latter terms are simply the conditional probabilities

$$\Pr[\text{symbol} = i | i \neq 0] = p_i/(1-p_0) \tag{4-7}$$

of the original source and which, with the exception of the missing zero, also satisfy (1-8) if the p_i do. Then H_θ can be viewed as the minimum average bits/sample required to code all the non-zero samples of $\tilde{Z}$ (where T large) and $(1-p_0)$ represents the fraction of all the original symbols which would typically be included in this sequence.

More specifically let[†]

$$\tilde{\Theta} \tag{4-8}$$

be the sequence of all the non-zero samples of Z. Then H_θ can be viewed as the minimum average bits/sample required to code $\tilde{\Theta}$ and $(1-p_0)$ represents the fraction of all the samples of Z included in $\tilde{\Theta}$. That is

$$E\{\mathcal{V}(\tilde{\Theta})\} = (1-p_0)T \tag{4-9}$$

Then by (4-3), the coding of long sequences from the original source close to $H(\tilde{P})$ can be achieved by "splitting" the sequence into two new sequences $\tilde{D}$

[†] We will later relabel the $\tilde{\Theta}$ symbols to $0, 1, 2, \cdots q-2$. This has no effect on the entropy arguments.

and $\tilde{\Theta}$, as described above and then coding each of them close to their corresponding entropies $H_\beta(p_0)$ and H_θ.

The remainder of this chapter and Chapter V will seek to develop adaptive code operators which take advantage of these concepts while recognizing that real world problems are not quite so idealized. While the memoryless assumption and probability ordering can usually be approximated well in practice the real $\tilde{P}$ is generally a priori unknown and varying. Measured distributions, $\overline{P}$ and $\overline{p}_0$ may even average out variations in data characteristics. In these practical situations, as in previous chapters, the entropy expressions in (4-3) - (4-5) serve as practical guides to performance but may no longer be bounds.

OPERATOR SPLIT[·]

Using the above discussions as a guide we define the reversible SPLIT[·] operator by

$$\text{SPLIT}[\tilde{Z}] \equiv \tilde{D} * \tilde{\Theta} \tag{4-10}$$

where $\tilde{Z}$ is the T sample sequence in (4-1), $\tilde{D}$ is a T sample binary sequence "identifying" which samples of $\tilde{Z}$ are non-zero and, $\tilde{\Theta}$ is a sequence of all the non-zero samples of $\tilde{Z}$ reduced by 1.[†] Specifically

$$\tilde{D} = d_1 d_2 \cdots\cdot d_T \tag{4-11}$$

where

$$d_i = \begin{cases} 0 \text{ if } z_i = 0 \\ 1 \text{ otherwise} \end{cases} \tag{4-12}$$

[†]This amounts to a relabeling of the original $\tilde{\Theta}$ in (4-9).

and the

$$N = \sum_{i=1}^{T} d_i \qquad (4\text{-}13)$$

samples of

$$\tilde{\Theta} = \theta_1 \ \theta_2 \ \theta_3 \cdots \theta_N \qquad (4\text{-}14)$$

can be generated by testing successive samples of $\tilde{Z}$ and creating new samples of $\tilde{\Theta}$ from $\tilde{Z}$ by the rule

$$\text{If } z_j > 0 \quad \begin{array}{c} \text{Create next} \\ \text{sample of } \tilde{\Theta} \\ \theta = z_j - 1 \end{array} \qquad (4\text{-}15)$$

Observe that by subtracting 1 we have essentially done the relabeling described in Fig. 1-1. By (4-7) the relabeled $\tilde{\Theta}$ symbols 0, 1, 2, $\cdots$ q-2 will satisfy the desired probability ordering in (1-8) if the input z_i do.

Reconstructing $\tilde{Z}$ from $\tilde{D}$ and $\tilde{\Theta}$ is obtained simply by testing successive samples of $\tilde{D}$ and generating the corresponding samples of $\tilde{Z}$ from $\tilde{\Theta}$ by the rule:

$$\begin{array}{ll} \text{If } d_i = 0 & z_i = 0 \\ & \\ \text{If } d_i = 1 & \begin{array}{c} z_i = \theta + 1 \\ \text{where } \theta \text{ is next} \\ \text{sample of } \tilde{\Theta} \end{array} \end{array} \qquad (4\text{-}16)$$

CODE OPERATOR

The SPLIT[·] operator just defined allows us to specify the general form of a new code operator $\psi_9[\cdot]$ by

$$\psi_9[\tilde{Z}] = \psi_\beta^j[\tilde{D}] * \psi_\theta[\tilde{\Theta}] \qquad (4\text{-}17)$$

where $\psi_\theta[\cdot]$ means any code operator for coding the variable length $\tilde{\Theta}$ sequences and $\psi_\beta^j[\cdot]$ is any binary code operator for coding the $\tilde{D}$ sequences. A block diagram is shown in Fig. 4-2. Included are the usual performance estimates (bounds) for later use.

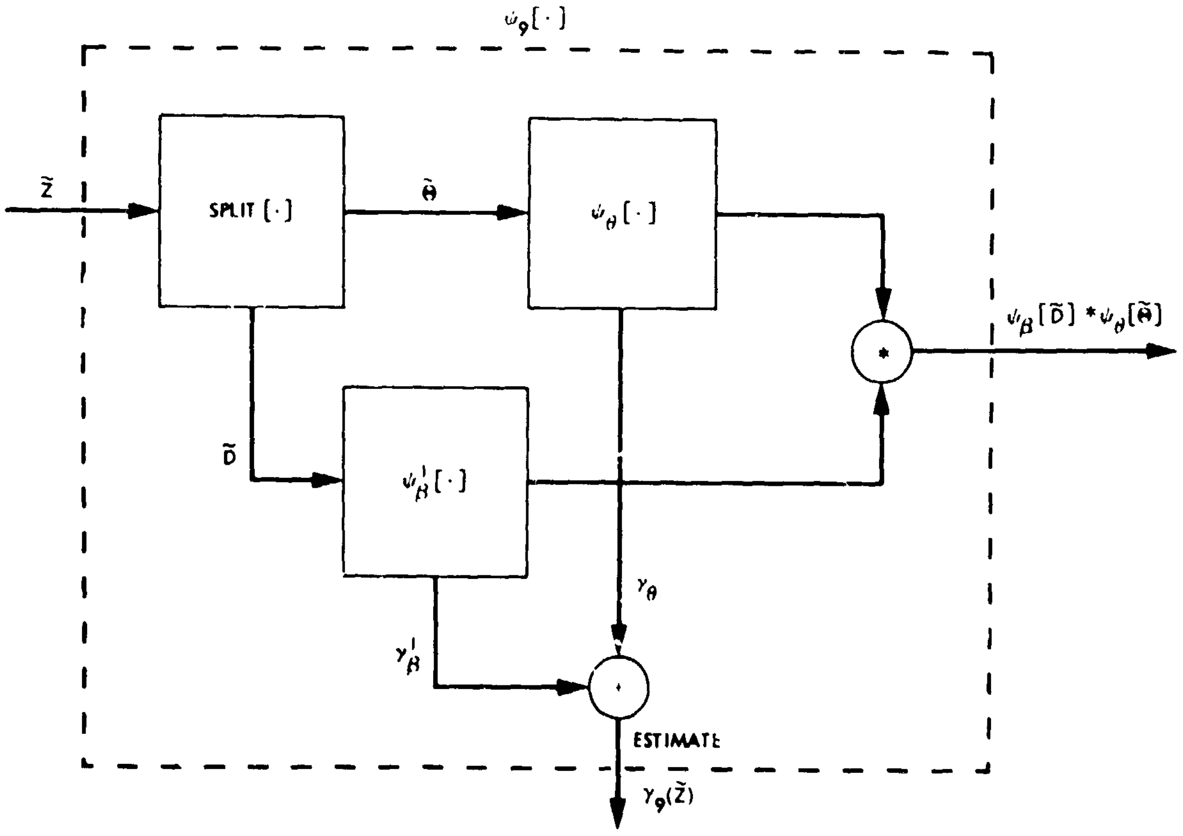


Fig. 4-2. General Form, Operator $\psi_9[\cdot]$

As noted in earlier discussions, operator $\psi_9[\cdot]$ will be an efficient operator for $\tilde{Z}$ sequences if $\psi_\beta^j[\cdot]$ and $\psi_\theta[\cdot]$ are efficient operators for $\tilde{D}$ and $\tilde{\Theta}$ sequences respectively. Since distributions of $\tilde{Z}$ are in practice a priori unknown and varying, so too are the corresponding distributions of samples of $\tilde{D}$ and $\tilde{\Theta}$. For $\psi_9[\cdot]$ to be an effective operator $\psi_\beta^j[\cdot]$ and $\psi_\theta[\cdot]$ must be able to adapt to variations in the statistical character of $\tilde{D}$ and $\tilde{\Theta}$.

<u>Coding of $\tilde{\Theta}$, $\psi_\theta[\cdot]$</u>

Of course any algorithm for coding $\tilde{\Theta}$ could be substituted for $\psi_\theta[\cdot]$. For our purposes we will assume an operator form utilizing the Basic Compressor, such as $\psi_6[\cdot]$ in (2-70) or some equivalent variable length form of $\psi_8[\cdot]$ or $\psi_8^J[\cdot]$ utilizing split sample modes (see Fig. 3-3). The specific algorithm details would of course depend on the particular application. However, based on the results of previous chapters an appropriate choice of such operators should provide a broad range of efficient performance as entropy H_θ in (4-5) varies.

<u>Estimate of $\mathscr{V}\left(\psi_\theta[\tilde{\Theta}]\right)$.</u> Since we have specified that $\psi_\theta[\cdot]$ be an operator form utilizing the Basic Compressor, estimates of performance of the type in (2-75) are easily obtained. Following earlier notation, we let $\gamma_\theta(\tilde{\Theta})$ represent this estimate.

<u>Coding of $\tilde{D}$, $\psi_\beta^j[\cdot]$</u>

Again, any algorithm for coding the binary $\tilde{D}$ sequences could be substituted for $\psi_\beta^j[\cdot]$ in (4-17). We will provide an efficient class of such binary code operators in Chapter V.

<u>Estimate of $\mathscr{V}\left(\psi_\beta^j[\tilde{D}]\right)$.</u> As we did for $\psi_\theta[\cdot]$, we let $\gamma_\beta^j(\tilde{D})$ represent an estimate of $\mathscr{V}\left(\psi_\beta^j[\tilde{D}]\right)$ of the form in (2-73). Then as in Fig. 4-2 we also have the equivalent form for $\psi_9[\cdot]$

$$\gamma_9(\tilde{Z}) = \gamma_\beta^j(\tilde{D}) + \gamma_\theta(\tilde{\Theta}) \tag{4-18}$$

Operator $\psi_{10}[\cdot]$

The design of operator $\psi_9[\cdot]$ is motivated by the distinct spike at zero in symbol probability distributions when $H(\overline{P}) \rightarrow 0$. However, by (4-3) operator $\psi_9[\cdot]$ will be efficient for any $H(\overline{P})$ provided both operators $\psi_\beta^j[\cdot]$ and $\psi_\theta[\cdot]$ are. Unfortunately requiring $\psi_9[\cdot]$ to be efficient at high entropies in addition to low entropies places unnecessary demands on the design of $\psi_\beta^j[\cdot]$. For example, when $H(\overline{P})$ is very low $\overline{p}_0 \rightarrow 1$ whereas when $H(\overline{P})$ becomes large and the distributions flatten, $\overline{p}_0 \rightarrow 0$. Thus $\psi_\beta^j[\cdot]$ is required to be efficient for unknown and varying $\overline{p}_0$ in the range from 0 to 1. This additional requirement can be avoided in most applications by instead specifying another operator, $\psi_{10}[\cdot]$, which selects between $\psi_9[\cdot]$ and some operator which can efficiently code $\tilde{Z}$ at the higher entropies. Such operators were the subject of previous chapters. The most general form was $\psi_8[\cdot]$ in Fig. 3-3 for which special cases include operator $\psi_5[\cdot]$ or $\psi_4[\cdot]$ when there is no need for split-sample modes. In this case the 4 or 5 replaces 8 in the following discussions.

It is a simple matter to define $\psi_{10}[\cdot]$, following the same procedures as in earlier chapters

$$\psi_{10}[\tilde{Z}] = \lambda * \psi_\lambda[\tilde{Z}] \tag{4-19}$$

where λ is the selected code operator number 8 or 9 (the concatenated λ being interpreted as a binary zero or one). It remains to specify the decision rule to complete the definition of $\psi_{10}[\cdot]$.

Optimum rule. Again the optimum rule is simply to count bits, choosing λ such that

$$\mathscr{L}\left(\psi_\lambda[\tilde{Z}]\right) = \min_{i=8,9} \mathscr{L}\left(\psi_i[\tilde{Z}]\right) \tag{4-20}$$

However, use of the Basic Compressor estimates can result in considerable simplification.

Simplified rule. Instead choose λ such that

$$\gamma_\lambda(\tilde{Z}) = \min_{i=8,9} \gamma_i(\tilde{Z}) \qquad (4\text{-}21)$$

Noting that λ requires one bit for identification we also have

$$\mathscr{L}\left(\psi_{10}[\tilde{Z}]\right) \le \gamma_{10}(\tilde{Z}) \equiv 1 + \gamma_\lambda(\tilde{Z}) \qquad (4\text{-}22)$$

and where typically

$$\mathscr{L}\left(\psi_{10}[\tilde{Z}]\right) \approx \gamma_{10}(\tilde{Z}) \qquad (4\text{-}23)$$

With $\psi_8[\cdot]$ available as an option which performs efficiently when $H(\overline{P})$ is high, a less sophisticated form of $\psi_9[\cdot]$ (which works well only at very low entropies) can be assumed. In this case, the decision rules in (4-20) and (4-21) would always choose $\lambda = 8$ when $H(\overline{P})$ was high.

Further simplifications. To further take advantage of the existence of the option $\psi_8[\cdot]$, note that in general a $\tilde{\theta}$ buffer equal in size to the length of $\tilde{Z}$ is required. But this buffer will tend to fill up only when $H(\overline{P})$ is high and $\overline{p}_0$ is low. But then $\psi_8[\cdot]$ is the code choice. Consequently the required buffer size can be reduced. Let

$$B_\theta \qquad (4\text{-}24)$$

be the length of this buffer and supplement the simplified decision rule by adding

$$\text{Set } \lambda = 8 \text{ if } \mathscr{L}(\tilde{\theta}) \ge B_\theta \qquad (4\text{-}25)$$

to override the decision in (4-21). B_0 can be experimentally chosen so that with high probability there is negligible loss in performance.

Block Diagram, $\psi_{10}[\cdot]$

A block diagram describing $\psi_{10}[\cdot]$ using 4-21 and 4-25 appears in Fig. 4-3.

PERFORMANCE

Extensive tests of a sophisticated $\psi_9[\cdot]$, using appropriate substitutions for $\psi_\beta^j[\cdot]$ from Chapter V, indicate average performance which remained close to $H(\overline{P})$ for any $H(\overline{P})$ in the range of 0 to 8 bits/sample, where $\overline{P}$ was a priori unknown and not changing significantly over the measurement span. This is shown in Fig. 4-4.

Average performance considerably under $H(\overline{P})$ was observed in situations where $\overline{P}$ changed significantly over the measurement span.[†]

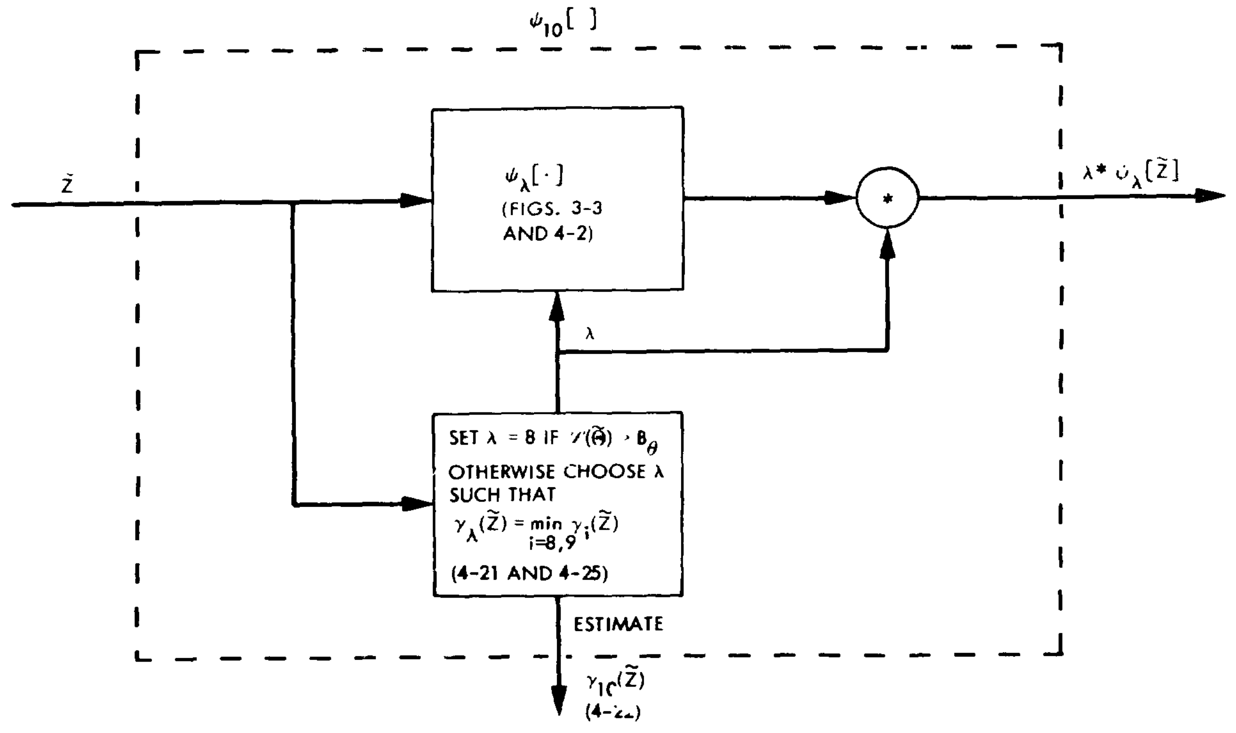
EXAMPLE

An example of the use of SPLIT[$\cdot$] is given in Fig. 4-5 for a T = 256 sample $\tilde{Z}$ sequence. Observe that the $\tilde{\Theta}$ that results is the same as $\tilde{M}^n$ in the example of (3-18) and (3-19). Thus the coding of $\tilde{\Theta}$ for operator $\psi_9[\cdot]$ in (4-17) has already been described as a special case of $\psi_0[\cdot]$ with

$$\mathscr{L}\left(\psi_\theta[\tilde{\Theta}]\right) = 235 \qquad (4\text{-}26)$$

This example will be continued in Chapter V after first developing appropriate binary code operators for $\tilde{D}$ (see 5-47).

[†]These tests were run on transform coefficients of the RM2 image compressor algorithm.[5]



$\psi_{10}[\ ]$
$\tilde{Z}$
$\psi_\lambda[\cdot]$ (FIGS. 3-3 AND 4-2)
$*$
$\lambda * \psi_\lambda[\tilde{Z}]$
λ
SET λ = 8 IF $\gamma(\tilde{\Theta}) > B_\theta$ OTHERWISE CHOOSE λ SUCH THAT $\gamma_\lambda(\tilde{Z}) = \min_{i=8,9} \gamma_i(\tilde{Z})$ (4-21 AND 4-25)
ESTIMATE
$\gamma_{10}(\tilde{Z})$ (4-22)

Fig. 4-3. Block Diagram Operator $\psi_{10}[\cdot]$

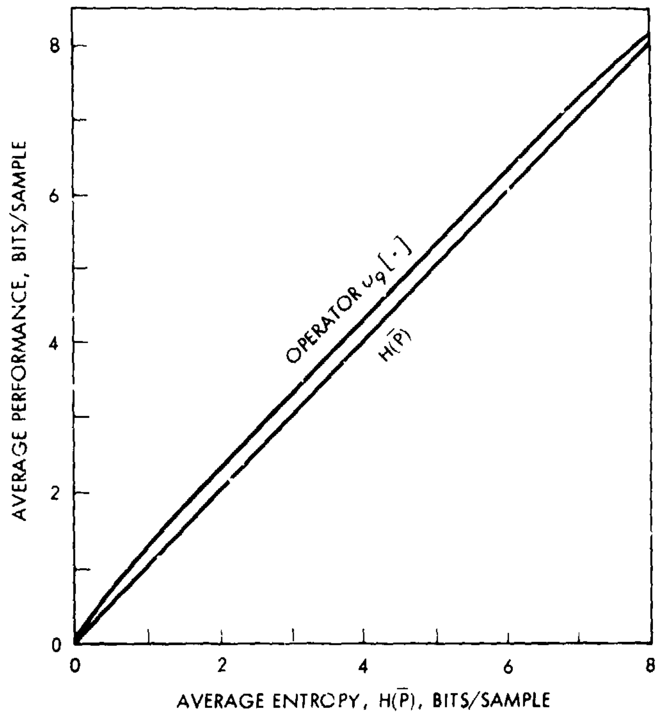


Fig. 4-4. Measured Average Performance, $\psi_9[\cdot]$

Z · $\overbrace{0\,0\ldots0\,0}^{68}$, 4, 0, 0, 0, 0, 6, 2, 1, 0, 0, 0, 3, 2, 0, 0, 0, 0, 0, 0, 3, 0, 3, 8, 0, 0, 11, 0, 0, 0, 0,

0, 11, $\overbrace{0\,0\ldots\mathbf{0\,0}}^{44}$, **23, 0,** 0, 0, 0, 0, 15, 8, 1, 15, 23, 0, 23, $\overbrace{0\,0\ldots\mathbf{0\,0}}^{8}$, **4, 0,** 1 2, 6, 0, 0, 0, **4,**

0, 0, 0, 1, 0, 0, 0, 22, 18, 0, 0, 0. 0, 6, 0, 5, $\overbrace{0\,0\ldots\mathbf{0\,0}}^{8}$, 6, 2, 0, 0, 0, 8, 0, 0, 14, 12, 0, 3, 0, 6,

0, 8, 0, 3, 2, 18, 4, 0, 0, 7, 7, 3, 0, 0, 2, $\overbrace{0\,0\ldots\mathbf{0\,0}}^{9}$, 2, 0, 6, 7, 7, 7, 0, 0, 23, 17, 1, $\overbrace{0\,0\ldots0\,0}^{9}$

U $\overbrace{0\,0\ldots\mathbf{0\,0}}^{68}$, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0,

0, 1, $\overbrace{0\,0\ldots\mathbf{0\,0}}^{44}$, **1,** 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, $\overbrace{0\,0\ldots\mathbf{0\,0}}^{8}$, **1,** 0, 1, 1, 1, 0, 0, 0, 1,

0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, $\overbrace{0\,0\ldots0\,0}^{8}$, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1,

0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, $\overbrace{0\,0\ldots0\,0}^{9}$, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, $\overbrace{0\,0\ldots0\,0}^{9}$

$\tilde{\Theta}$ · 3, 5, 1, 0, 2, 1, 2, 2, 7, 10, 10, 22, 14, 7, 0, 14, 22, 22, 3, 0, 1, 5, 3, 0, 21, 17, 5, 4, 5, 1, 7, 13, 11, 2,

5, 7, 2, 1, 17, 3, 6, 6, 2, 1, 1, 5, 6, 6, 6, 22, 16, 0

Fig. 4-5. Example SPLIT[·]

V. CODING FOR BINARY MEMORYLESS SOURCES WITH UNKNOWN STATISTICS

This chapter provides a class of efficient code operators for real binary sources which can be realistically modeled as memoryless with a priori unknown and varying probabilities.

Several of these binary operators, denoted $\psi_{\beta}^{j}[\cdot]$ and $\psi_{\beta'}^{j}[\cdot]$ are intimately related to the non-binary operators, $\psi_9[\cdot]$ and $\psi_{10}[\cdot]$, developed in Chapter IV. It will be advantageous for the reader to become familiar with the structure of the latter operators before proceeding into the details here.

PRACTICAL ASSUMPTIONS

Data Model

Let

$$\tilde{D} = d_1 d_2 \dots d_T \tag{5-1}$$

be a T sample sequence where the d_i are the output of a binary memoryless source with probability of a zero, p_0. To reflect real world variations in p_0 we will generally assume that the p_0 for each $\tilde{D}$ is a priori unknown and where p_0 can lie in the range $0 \le p_0 \le 1$.

Binary Entropy

From (4-4) the binary entropy function is given as

$$H_{\beta}(p_0) = - p_0 \log_2 p_0 - (1 - p_0) \log_2 (1 - p_0) \tag{5-2}$$

and is shown in Fig. 5-1 for $0 \le p_0 \le 1$.

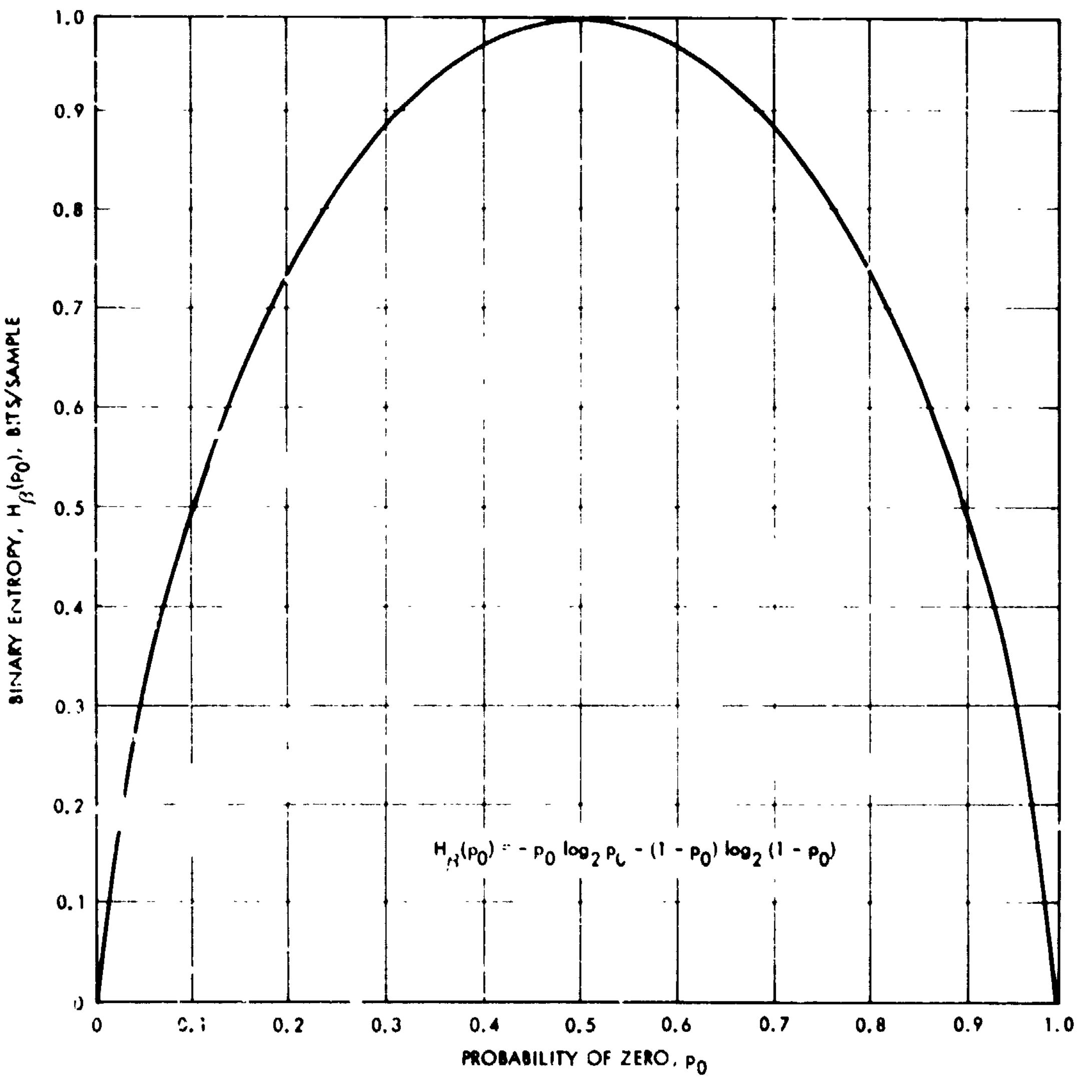


Fig. 5-1. Binary Entropy, $H_\beta(\cdot)$

Quite unsurprisingly, $H_\beta(p_0)$ is symmetrical about $p_0 = 1/2$ where $H_\beta(p_0)$ reaches its maximum value of 1. At this point zeroes and ones are totally random.

As noted from previous discussions the interpretation of entropy in practical problems usually requires some caution. The use of $H_\beta(\cdot)$ generally requires the measurement of an average probability, $\bar{p}_0$, determined over a span of samples which may be much larger than the length of $\tilde{D}$. When the real p_0 is slowly varying over this measurement span then average code performance above but close to $H_\beta(\bar{p}_0)$ can be viewed as "efficient." $H_\beta(p_0)$ acts as an approximate bound in this case because the data behaves like an ideal memoryless source. However, if data statistics are significantly changing over the measurement span, p_0 will average out the changes. Since $H_\beta(\bar{p}_0)$ cannot account for the possibility of adapting a coder to these variations, average performance below $H_\beta(\bar{p}_0)$ may be possible.

The remainder of this chapter will seek to develop binary code operators which exhibit efficient performance characteristics in the sense described above.

PREPROCESSING OF $\tilde{D}$

The samples of $\tilde{D}$ are not in a form suitable for a direct application of previously developed code operators. This section will provide the necessary preprocessing of $\tilde{D}$. In so doing we will restrict the choice of various parameters to simplify discussion and potential practical implementations. However, it is felt that the chosen parameters are a good choice from a performance standpoint also. More general investigations are left for further study.

eth Extension

We first restrict the length of $\tilde{D}$ to be multiple of e so that

$$\mathscr{L}(\tilde{D}) = T = \tau e \tag{5-3}$$

Using operator $\text{Ext}^e[\cdot]$ from (2-6) yields the e^{th} extension of $\tilde{D}$

$$\tilde{D}' = \text{Ext}^e[\tilde{D}] - d_1' \, d_2' \, d_3' \, \dots \, d_\tau' \tag{5-4}$$

where the τ samples of D' take on the values 0, 1, 2, ... 2^e-1 determined by the corresponding binary e-tuples of $\tilde{D}$. That is, the standard binary representation of d_i' is simply the i^{th} consecutive e-tuple of $\tilde{D}$. We will principally rely on the context of a discussion to identify whether a d_i' refers to its non-binary value or its binary representation.

e-tuple Probability. If the binary digits representing any d_i' are the result of a binary memoryless source then the probability that these digits will form a particular e-tuple with j ones is easily given as

$$\Pr\begin{bmatrix}\text{Any e-tuple with} \\ \text{j ones}\end{bmatrix} = p_0^{e-j}(1 - p_0)^j \tag{5-5}$$

where p_0 is the usual probability that an individual binary digit will be zero. There are

$$\binom{e}{j} = \frac{e!}{j!(e-j)!} \tag{5-6}$$

such e-tuples. Then, provided $p_0 \geq 1/2$ we must have

$$\Pr\begin{bmatrix}\text{Any e-tuple with} \\ \text{j ones}\end{bmatrix} \geq \Pr\begin{bmatrix}\text{Any e-tuple with} \\ \text{i}' > \text{j ones}\end{bmatrix} \tag{5-7}$$

Conversely, if $p_0 \leq 1/2$ the inequality sign in (5-7) reverses.

Thus while $\tilde{D}'$ is a sequence of (approximately) independent samples taking on the values 0, 1, 2, ... 2^e-1 these values do not occur with the desired probability ordering of (1-8). For example, by the above arguments

$$\Pr\,[d'_i = 4] = \Pr\begin{bmatrix} \text{e-tuple} \\ 000\ldots0100 \end{bmatrix} > \Pr\,[d'_i = 3] = \Pr\begin{bmatrix} \text{e-tuple} \\ 00\ldots011 \end{bmatrix} \tag{5-8}$$

where $p_0 > 1/2$. Thus $\tilde{D}'$ must be preprocessed further before the results of earlier chapters can be used.

Ordering Probabilities

We propose two reversible mappings of d'_i samples given by $f_0[\cdot]$ and $f_1[\cdot]$. The functional specification of these mappings can be described by the following:

$$\Delta_i^0 = f_0[d'_i] \tag{5-9}$$

and

$$\Delta_i^1 = f_1[d'_i] \tag{5-10}$$

where Δ_i^0 and Δ_i^1 take on the values 0, 1, 2, ... 2^e-1 (non-binary interpretation) and where

$$\Pr[\Delta_i^\zeta = 0] \geq \Pr[\Delta_i^\zeta = 1] \geq \ldots \geq \Pr[\Delta_i^\zeta = 2^e-1] \tag{5-11}$$

is satisfied for $\zeta = 0$ when $p_0 \geq 1/2$ and for $\zeta = 1$ when $p_0 \leq 1/2$ (provided the binary memoryless source model holds for the digits of $\tilde{D}$).

Extending our notation slightly, the application of $f_\zeta[\cdot]$ to all the τ samples of $\tilde{D}'$ in (5-4) yields the sequence $\tilde{\Delta}_\zeta$ given by

$$\begin{aligned} \tilde{\Delta}_\zeta = f_\zeta[\tilde{D}'] &= f_\zeta[d'_1] \ast f_\zeta[d'_2] \ast \ldots f_\zeta[d'_\tau] \\ &= \Delta_1^\zeta \ast \Delta_2^\zeta \ldots \Delta_\tau^\zeta \end{aligned} \tag{5-12}$$

where $\zeta = 0$ or 1.

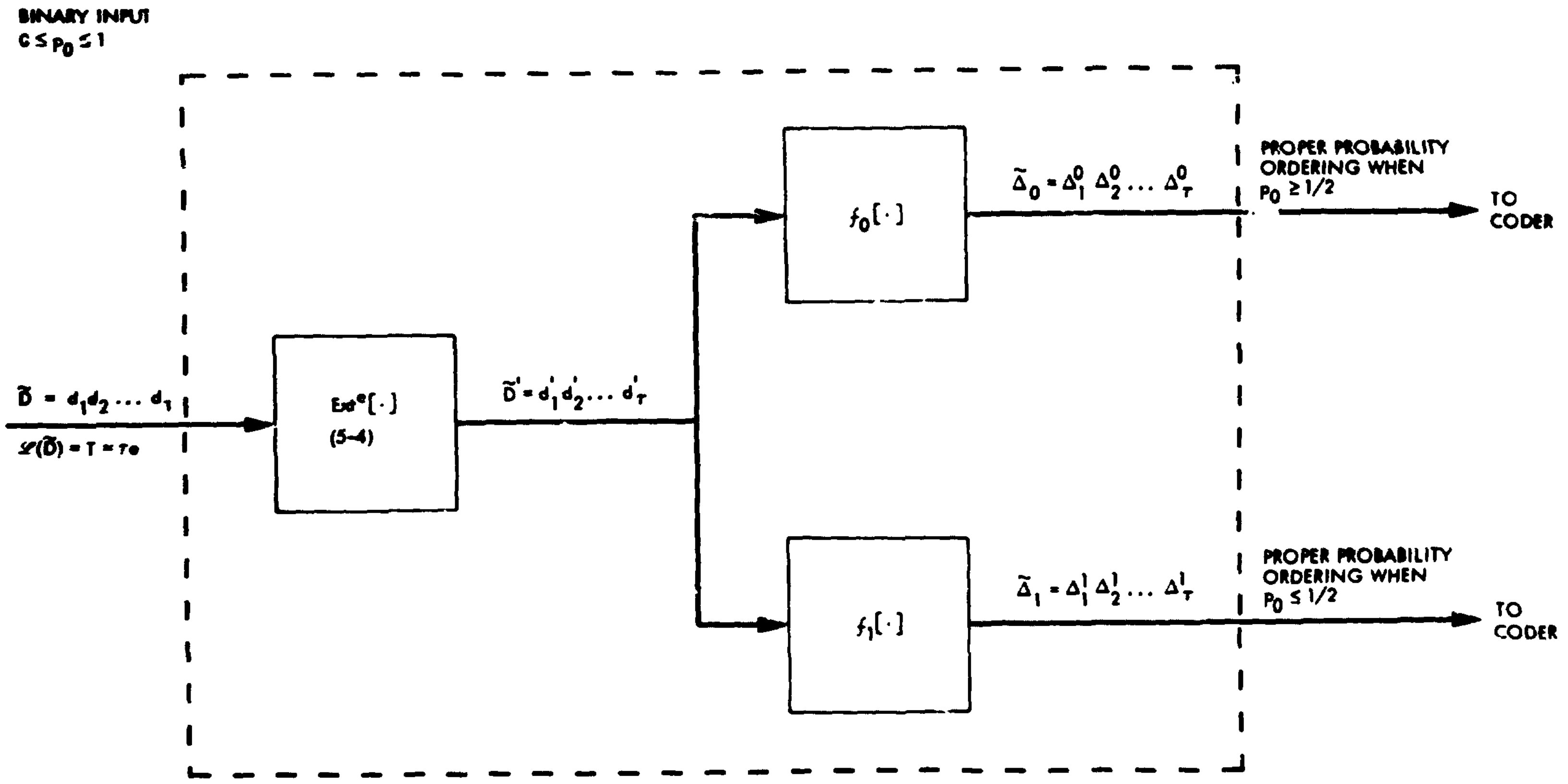
Then by the above assumptions for the $f_\zeta[\cdot]$, either $\tilde{\Delta}_0$ or $\tilde{\Delta}_1$ should meet the desired requirements for a preprocessed data sequence in Fig. 1-1 when $0 \le p_0 \le 1$. Further, since the operations of $\mathrm{Ext}^e[\cdot]$ and $f_\zeta[\cdot]$ are reversible the per sample entropy of $\tilde{\Delta}_\zeta$ should be increased (by a factor of e) into the efficient operating range of the Basic Compressor. The results of preceding chapters should be directly applicable. A block diagram summarizing the discussion and notation thus far is given in Fig. 5-2.

Derivation and Implementation of $f_\zeta[\cdot]$

From (5-7) we see that to obtain the ordering in (5-11) for $p_0 \ge 1/2$, $f_0[\cdot]$ need assign the number zero to the all zero e-tuple, the consecutive numbers 1 to $\binom{e}{1}$ to e-tuples with a single one, the numbers $\binom{e}{1} + 1$ up to $\binom{e}{1} + \binom{e}{2}$ to e-tuples with two ones and so on. Without any additional constraints the particular assignment of numbers to e-tuples with the same number of ones is arbitrary.

An $f_1[\cdot]$ mapping which results in the desired ordering in (5-11) when $p_0 \le 1/2$ can be described in exactly the same way by reversing the roles of zeroes and ones. We will investigate this further.

Joint Implementation of $f_0[\cdot]$ and $f_1[\cdot]$. Let ν be any binary e-tuple and $\bar{\nu}$ its bit by bit complement. We note that if ν is one of the $\binom{e}{j}$ possible e-tuples with j ones then $\bar{\nu}$ is one of the $\binom{e}{e-j} = \binom{e}{j}$ possible e-tuples with j zeroes. We can then implement $f_0[\cdot]$ and $f_1[\cdot]$ as a table lookup by arranging all possible e-tuples so that e-tuples with fewer ones appear at a higher position (closer to the top) and that if ν is in position k-1 from the top (counting zero), $\bar{\nu}$ appears in position k-1 from the bottom. Then $f_0[\nu]$ is the position of ν from the bottom. This is shown in Table 5-1.



BINARY INPUT
$0 \le p_0 \le 1$
$\tilde{D} = d_1 d_2 \ldots d_T$
$\mathscr{L}(\tilde{D}) = T = \tau e$
$Ext^e[\cdot]$
(5-4)
$\tilde{D}' = d'_1 d'_2 \ldots d'_T$
$f_0[\cdot]$
$f_1[\cdot]$
$\tilde{\Delta}_0 = \Delta^0_1 \Delta^0_2 \ldots \Delta^0_T$
$\tilde{\Delta}_1 = \Delta^1_1 \Delta^1_2 \ldots \Delta^1_T$
PROPER PROBABILITY ORDERING WHEN $p_0 \ge 1/2$
PROPER PROBABILITY ORDERING WHEN $p_0 \le 1/2$
TO CODER
TO CODER

Fig. 5-2. Basic Preprocessing of $\tilde{D}$

Table 5-1. Arrangement of $f_\gamma[\cdot]$ for Table Lookup

Position from Top $f_0[\cdot]$	Input e-tuple	Position from Bottom $f_1[\cdot]$
0	all zeroes	$2^e - 1$
1	0 0 . . . . 0 0 1	$2^e - 2$
2	0 0 . . . . 0 1 0	$2^e - 3$
k - 1	ν (has j ones)	$2^e - k$
$2^e - k$	$\bar{\nu}$ (has j zeroes)	k - 1
$2^e - 1$	all ones	0

Then

$$f_1[\bar{\nu}] = f_0[\nu]$$

and similarly (5-13)

$$f_0[\bar{\nu}] = f_1[\nu]$$

This means that $f_1[\nu]$ can be obtained by first complementing an input e-tuple and using $f_0[\cdot]$ as shown in Fig. 5-3. However, if both $f_0[\nu]$ and $f_1[\nu]$ are simultaneously desired, this approach requires two table lookups. This can be avoided.

Simply note that while $f_0[\nu]$ is the position of from the top of the table, $f_1[\nu]$ is the position from the bottom so that

$$f_1[\nu] = 2^e - 1 - f_0[\nu] \tag{5-14}$$

But (5-14) is the same as

$$f_1[\nu] = \bar{f}_0[\nu] \tag{5-15}$$

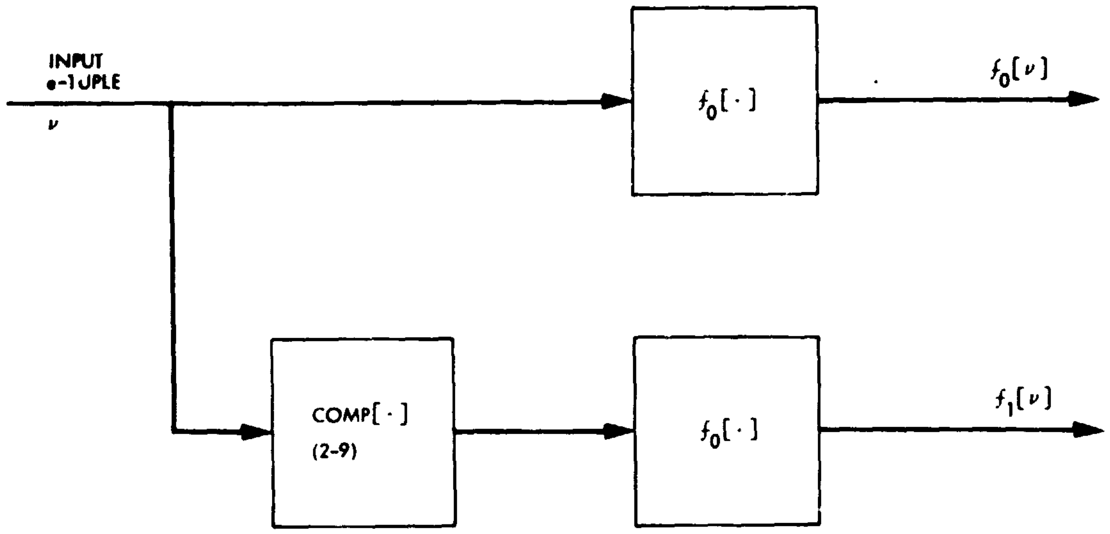


Fig. 5-3. Implementing $f_0[\cdot]$ and $f_1[\cdot]$, Method 1

where

$$\bar{f}_0[\nu] = \mathrm{COMP}[f_0[\nu]]$$

is the bit by bit complement of $f_0[\nu]$ interpreted as a binary e-tuple. Thus $f_1[\cdot]$ and $f_0[\cdot]$ can be implemented as a single table lookup as shown in Fig. 5-4.

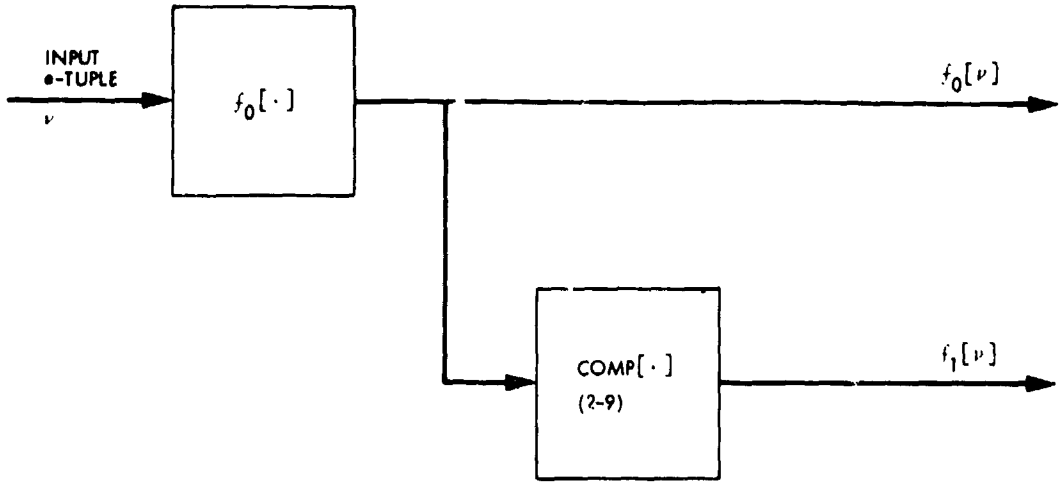


Fig. 5-4. Simplified Implementation of $f_0[\cdot]$ and $f_1[\cdot]$

$\underline{f_0[\cdot] \text{ and } f_1[\cdot] \text{ for } e = 4}$

We will henceforth restrict the parameter e to be four. This choice has some practical advantages with no obvious disadvantages as far as coding performance is concerned. Quite without surprise, we will seek to make use of the code operators developed in previous chapters. If we restrict e to be 4, the preprocessed samples of $\tilde{\Delta}_\zeta$ in (5-12) will have entropies in the range of 0 to 4 bits/sample, quite suitable for a direct application of the Basic Compressor operator developed in Chapter II. Higher entropies might necessitate split-sample modes as described in Chapter III. This would be an unnecessary complication for this application.

A complete table defining both $f_0[\cdot]$ and $f_1[\cdot]$ for e = 4 appears in Table 5-2.

GENERAL CODE OPERATOR STRUCTURE

Since by (5-11) and the preceding developments, $\tilde{\Delta}_0$ and $\tilde{\Delta}_1$ satisfy the desired requirements for preprocessed data sequences (Fig. 1-1) the results of previous chapters should be directly applicable. The most obvious example is to use the Basic Compressor operator, $\psi_4[\cdot]$, to code $\tilde{D}$ by treating $\tilde{\Delta}_0$ and $\tilde{\Delta}_1$ as single J = T/4 sample Basic Compressor blocks or to partition $\tilde{\Delta}_0$ and $\tilde{\Delta}_1$ into several smaller blocks and then apply $\psi_5[\cdot]$ from (2-71).† More generally, any operator, say $\psi_0[\cdot]$, can be used to code $\tilde{\Delta}_0$ and $\tilde{\Delta}_1$ in the same manner. To this end we define

†Note that when using $\psi_5[\cdot]$ there is some obvious practical advantage to restricting the length of the $\tilde{\Delta}_i$ (and hence $\tilde{D}$) to be a multiple of some convenient Basic Compressor block size J (e.g., 16). Then from (5-3) we have

$$\mathscr{I}(\tilde{\Delta}_i) = \tau = \tau' J \tag{5-16}$$

and

$$\mathscr{I}(\tilde{D}) = 4\tau' J \tag{5-17}$$

Table 5-2. $f_0[\cdot]$ and $f_1[\cdot]$ for e = 4

Input 4-Tuple ν		$f_0[\nu]$		$f_1[\nu]$	
Non Binary	As 4-Tuple	Non Binary	As 4-Tuple	Non Binary	As 4-Tuple
0	0 0 0 0	0	0 0 0 0	15	1 1 1 1
1	0 0 0 1	1	0 0 0 1	14	1 1 1 0
2	0 0 1 0	2	0 0 1 0	13	1 1 0 1
4	0 1 0 0	3	0 0 1 1	12	1 1 0 0
8	1 0 0 0	4	0 1 0 0	11	1 0 1 1
3	0 0 1 1	5	0 1 0 1	10	1 0 1 0
6	0 1 1 0	6	0 1 1 0	9	1 0 0 1
5	0 1 0 1	7	0 1 1 1	8	1 0 0 0
10	1 0 1 0	8	1 0 0 0	7	0 1 1 1
9	1 0 0 1	9	1 0 0 1	6	0 1 1 0
12	1 1 0 0	10	1 0 1 0	5	0 1 0 1
7	0 1 1 1	11	1 0 1 1	4	0 1 0 0
11	1 0 1 1	12	1 1 0 0	3	0 0 1 1
13	1 1 0 1	13	1 1 0 1	2	0 0 1 0
14	1 1 1 0	14	1 1 1 0	1	0 0 0 1
15	1 1 1 1	15	1 1 1 1	0	0 0 0 0

$$\psi_\beta^j[\tilde{D}] = \zeta * \psi_j[\tilde{\Delta}_\zeta] \tag{5-18}$$

where ζ identifies the selected choice of preprocessed sequence $\tilde{\Delta}_0$ or $\tilde{\Delta}_1$ (the concatenated ζ being interpreted as a binary zero or one) and j is a priori fixed (e.g., 4 or 5).

Decision Rules

Optimum. By counting bits ζ is chosen such that

$$\mathscr{L}(\psi_j[\tilde{\Delta}_\zeta]) = \min_{i=0,1} \mathscr{L}(\psi_j[\tilde{\Delta}_i]) \tag{5-19}$$

Simplified rule. By using the Basic Compressor performance estimates (bounds), ζ is chosen such that

$$\gamma_j(\tilde{\Delta}_\zeta) = \min_{i=0,1} \gamma_j(\tilde{\Delta}_i) \tag{5-20}$$

Since ζ requires one bit for specification we have the bound and estimate for $\psi_\beta^j[\tilde{D}]$ given by

$$\mathscr{L}(\psi_\beta^j[\tilde{D}]) \leq \gamma_\beta^j(\tilde{D}) \equiv 1 + \gamma_j(\tilde{\Delta}_\zeta) \tag{5-21}$$

Majority rule. In some applications it may be acceptable to simply choose ζ by the majority rule

$$\zeta = \begin{cases} 0 & \text{if } \sum_{i=1}^{T} d_i < \frac{T}{2} \\ 1 & \text{otherwise} \end{cases} \tag{5-22}$$

The applicability of this rule is most suitable when p_0 is slowly varying and the length of $\tilde{D}$ is large or when an operator is expected to ope·ate only for very low or very high values of p_0.

Note that (5-21) is still a useful estimate when (5-22) is used instead of (5-20).

Block Diagram

A block diagram describing $\psi_\beta^j[\cdot]$ is shown in Fig. 5-5 assuming the simplified rule in (5-20).

Restricted Range of p_0, $\psi_\beta^j[\cdot]$

In some applications the range of p_0 may be limited to either $0 \le p_0 \le 1/2$ or $1/2 \le p_0 \le 1$. Quite obviously under these conditions, there is no need to choose between $\tilde{\Delta}_0$ or $\tilde{\Delta}_1$ for coding purposes, and correspondingly no need for the ζ identifier in (5-18). It is useful to define two simplified operators which fit these conditions. Define $\psi_{\beta'}^{\zeta}[\cdot]$ for $\zeta = 0$ and 1 by

$$\psi_{\beta'}^{j}[\tilde{D}] = \psi_j[\tilde{\Delta}_\zeta] \qquad (5\text{-}23)$$

where we a priori choose $\zeta = 0$ if $p_0 \ge 1/2$ and $\zeta = 1$, otherwise. The block diagram for $\psi_\beta^j[\cdot]$ in Fig. 5-5 reduces to that shown in Fig. 5-6.

THE BASIC BINARY OPERATORS

A direct application of the Basic Compressor to the coding of the $\tilde{\Delta}_\zeta$ in Figs. 5-5 and 5-6 yields "the Basic Binary Operators" $\psi_\beta^4[\cdot]$, $\psi_\beta^5[\cdot]$ and $\psi_{\beta'}^4[\cdot]$, $\psi_{\beta'}^5[\cdot]$. We will investigate these operators further in this section. Before proceeding, note that $\psi_\beta^4[\cdot]$ is really a special case of $\psi_\beta^5[\cdot]$ in which only one Basic Compressor block is used. Similarly, the $\psi_{\beta'}^j[\cdot]$ are really simpler versions of $\psi_\beta^j[\cdot]$ designed to work over half the range of p_0. Thus we can concentrate on $\psi_\beta^5[\cdot]$ without loss in generality.

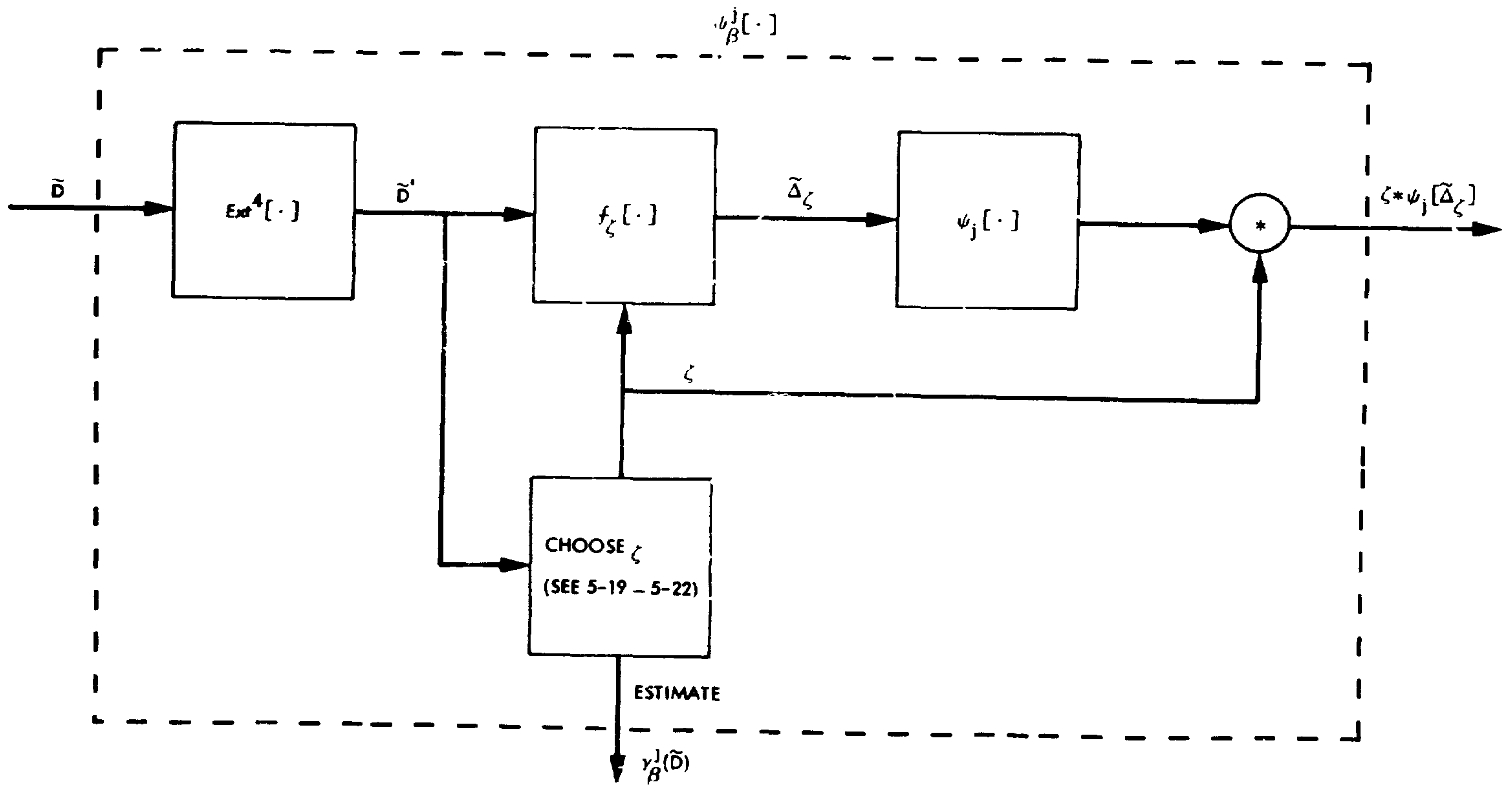


Fig. 5-5. Block Diagram, $\psi_\beta^j[\cdot]$

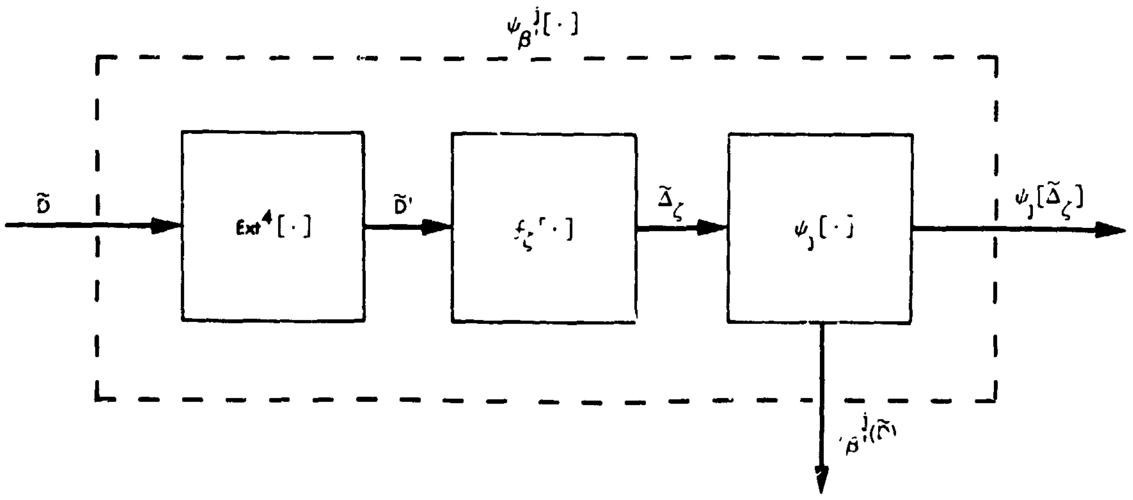


Fig 5-6. Operator $\psi_{\beta'}^{j}[\cdot]$ for Limited Range of p_0

Implementation of Simplified Rule

Observe from the developments in Chapter II that an essential element in the generation of Basic Compressor estimates $\gamma_4(\cdot)$ or $\gamma_5(\cdot)$ is the computation of "fundamental sequence length" F. By (2-4), this amounts to adding the samples making up a J-Sample Basic Compressor block. The test in (5-20) and Fig. 5-5 suggests that these same computations are required for each block making up both $\tilde{\Delta}_0$ and $\tilde{\Delta}_1$. However, the structure of Tables 5-1 and 5-2 can be used to avoid a requirement to add the samples of $\tilde{\Delta}_1$. The results provide an additional practical argument for the use of a block size of J = 16.

Two's Complement. Assuming the same partitioning of $\tilde{\Delta}_0$ and $\tilde{\Delta}_1$ into J = 16 Sample Basic Compressor blocks let

$$\tilde{X}^i = x_1^i \, x_2^i \ldots x_{16}^i \qquad (5\text{-}24)$$

be any such block from $\tilde{\Delta}_i$ with FS length given by (2-4) as

$$F^i = 16 + \sum_{k=1}^{16} x_k^i \qquad (5\text{-}25)$$

where i = 0, 1.

By (5-14) we have

$$x_k^1 = 15 - x_k^0 \qquad (5\text{-}26)$$

We can then write F^1 as

$$F^1 = 16 + \sum_{k=1}^{16} (15 - x_k^0)$$

$$= 16 + \left\{256 - F^0\right\} \qquad (5\text{-}27)$$

or more simply

$$F^1 = 16 + \text{Two's Complement } [F^0] \qquad (5\text{-}28)$$

Expected Performance

Under the rather ideal assumption that all the T samples of $\tilde{D}$ are the result of a binary memoryless source with a priori unknown but constant p_0, the expected performance of $\psi_\beta^4[\cdot]$ and $\psi_\beta^5[\cdot]$ can be bounded. The results of Appendix A provide a result of the form[†]

$$\frac{1}{T} E\left\{\mathscr{L}(\psi_\beta^j[D])\right\} \le \Lambda_\beta^j (p_0, T) \text{ bits/sample} \tag{5-29}$$

for j = 4 and 5, and $0 \le p_0 \le 1$.

A plot of $\Lambda_\beta^4(p_0, 256)$ is given in Fig. 5-7.

Observe from Eq. A-2 that because p_0 is assumed constant over the length of $\tilde{D}$ additional Basic Compressor blocks of $\psi_\beta^5[\cdot]$ actually degrade performance by increasing the overhead. In this situation there is no advantage to adapting since the data characterization is the same for all of $\tilde{D}$. However, in many practical problems p_0 will change over the length of $\tilde{D}$ and the added flexibility to change code options may more than make up for the additional overhead. Average performance under $H_\beta(\bar{p}_0)$, where $\bar{p}_0$ is the usual measured average of p_0, is a typical result.

<u>Performance of $\psi_\beta^j[\cdot]$</u>. With no elaboration necessary, we have

$$\frac{1}{T} E\left\{\mathscr{L}(\psi_{\beta'}^j[D])\right\} \le \Lambda_\beta^j(p_0, T) - \frac{1}{T} \tag{5-30}$$

for j = 4 or 5, provided p_0 is limited to either $p_0 \ge 1/2$ or $p_0 \le 1/2$.

<u>Example of $\psi_\beta^5[\cdot]$</u>

Let $\tilde{D}$ be the T = 256 sample binary sequence of Fig. 4-5. Following Fig. 5-5 we obtain $\tilde{D}'$ as

[†]We have taken the liberty to leave out the additional parameter of η for the number of Basic Compressor blocks in $\psi_\beta^5[\cdot]$.

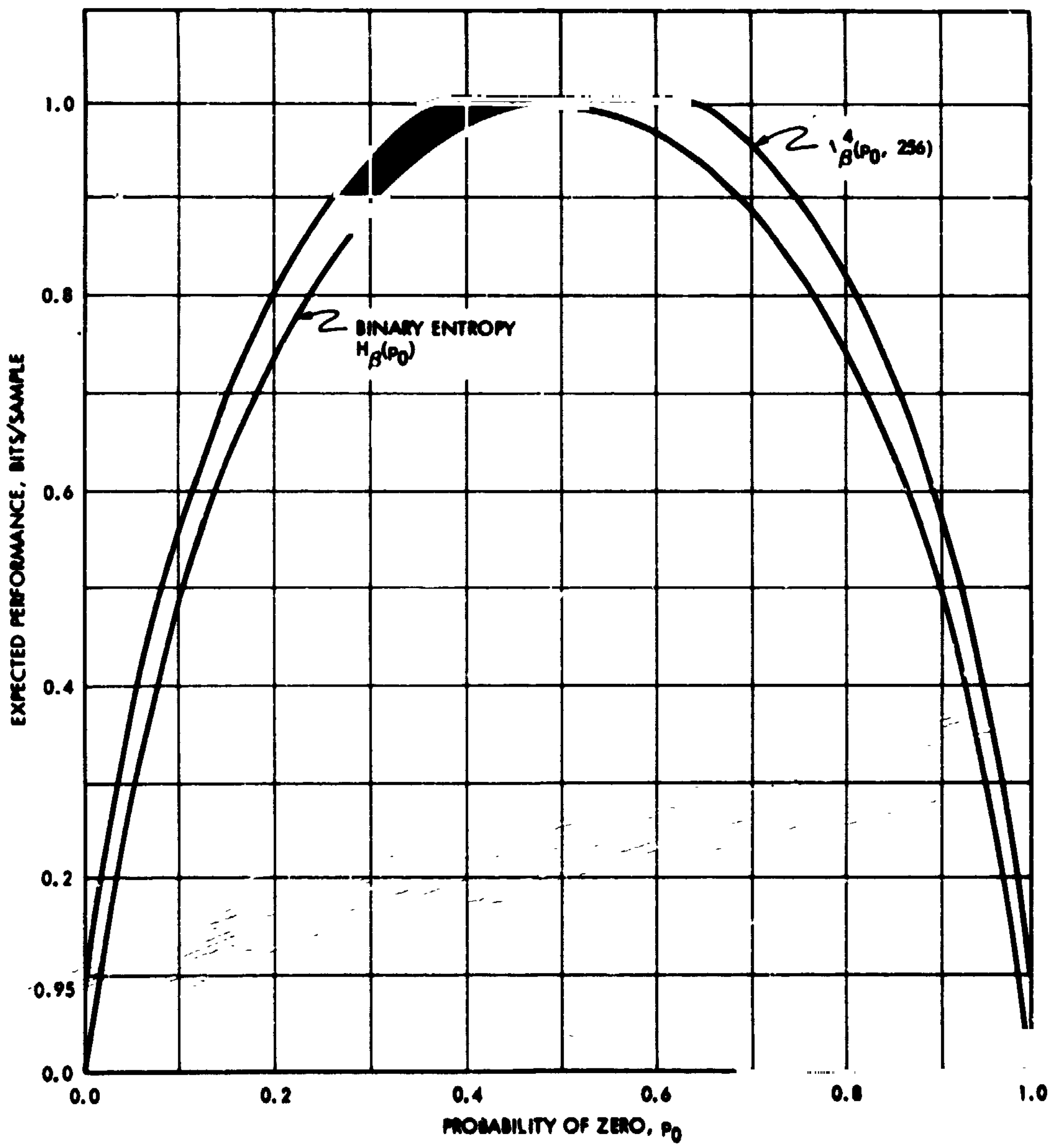


Fig. 5-7. Expected Performance of $\psi_\beta^4[\cdot]$ on Ideal Memoryless Source, Unknown but Constant p_0

$$\tilde{D}' = \text{Ext}^4[\tilde{D}] = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)*$$
$$(0, 8, 7, 1, 8, 1, 6, 4, 1, 0, 0, 0, 0, 0, 0, 0)*$$
$$(0, 0, 0, 0, 8, 3, 14, 8, 0, 5, 12, 4, 4, 6, 1, 4)*$$
$$(0, 3, 1, 3, 5, 5, 14, 7, 2, 0, 0, 11, 12, 14, 0, 0) \tag{5-31}$$

where we have conveniently split $\tilde{D}'$ into four 16 sample blocks. Now applying $f_0[\cdot]$ of Table 5-2 to each sample of $\tilde{D}'$ yields

$$\tilde{\Delta}_0 = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)*$$
$$(0, 4, 11, 1, 4, 1, 6, 3, 1, 0, 0, 0, 0, 0, 0, 0)*$$
$$(0, 0, 0, 0, 4, 5, 14, 4, 0, 7, 10, 3, 3, 6, 1, 3)*$$
$$(0, 5, 1, 5, 7, 7, 14, 11, 2, 0, 0, 12, 10, 14, 0, 0) \tag{5-32}$$

Applying $f_1[\cdot]$ to obtain $\tilde{\Delta}_1$ can be done in several ways. The simplest is to use Table 5-2 again, yielding

$$\tilde{\Delta}_1 = (15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15)*$$
$$(15, 11, 4, 14, 11, 14, 9, 12, 14, 15, 15, 15, 15, 15, 15, 15)*$$
$$(15, 15, 15, 15, 11, 10, 1, 11, 15, 8, 5, 12, 12, 9, 14, 12)*$$
$$(15, 10, 14, 10, 8, 8, 1, 4, 13, 15, 15, 3, 5, 1, 15, 15) \tag{5-33}$$

We could also obtain $\tilde{\Delta}_1$ by using (5-13) and (5-15). For example, take sample 19 of $\tilde{D}'$. Its non-binary form is seven = 0111 = ν. Complementing each bit we get v = 1000 = eight. Then using Table 5-2 for $f_0(v)$ we get $f_0(8)$ = 0100 = four = $f_1(7)$.

Following the procedure in (5-15) we first obtain $f_0(7)$ = eleven = 1011. Complementing each bit we obtain $f_0(7)$ = 0100 = four. The same result.

Code Estimates. $\tilde{\Delta}_0$ and $\tilde{\Delta}_1$ have already been partitioned into 16 sample blocks to accommodate application of the Basic Compressor in the form of $\psi_5[\cdot]$ in (2-69). To facilitate notation, let

$$\tilde{\Delta}_i = \tilde{\Delta}_i(1) * \tilde{\Delta}_i(2) * \tilde{\Delta}_i(3) * \tilde{\Delta}_i(4) \qquad (5\text{-}34)$$

where the $\tilde{\Delta}_i(\ell)$ corresponds to the partitioned blocks in (5-32) and (5-33) and let

$$F_\ell^i \qquad (5\text{-}35)$$

be the fundamental sequence length corresponding to $\tilde{\Delta}_i(\ell)$. Then we have by (2-74) and (2-28)

$$\gamma_5(\tilde{\Delta}_i) = \sum_{\ell=1}^{4} \gamma_4(\tilde{\Delta}_i(\ell)) = 8 + \sum_{\ell=1}^{4} \gamma_{ID}(F_\ell^i) \qquad (5\text{-}36)$$

Making use of the procedures developed in Chapter II for Basic Compressor estimates leads to the results for $\tilde{\Delta}_0$ shown in Table 5-3.

Table 5-3. Estimates for $\tilde{\Delta}_0$

F_ℓ^0	16	47	76	104
ID	0	1	2	3
γ_{ID}	6	47	58	64

Using Table 5-3, $\gamma_5(\tilde{\Delta}_0)$ becomes

$$\gamma_5(\tilde{\Delta}_0) = 183 \tag{5-37}$$

Making use of (5-27) or (5-28) we see that

$$\gamma_5(\tilde{\Delta}_0) \ll \gamma_5(\tilde{\Delta}_1) \tag{5-38}$$

so that the simplified decision rule of (5-20) provides the choice of

$$\zeta = 0 \tag{5-39}$$

(the optimum rule of (5-19) also yields this decision).

Then by (5-21) we have

$$\mathscr{L}(\psi_\beta^5[\tilde{D}]) \le 184 \tag{5-40}$$

Coding $\tilde{\Delta}_0$. Making use of the Basic Compressor code decisions in Table 5-3 the coding of $\tilde{D}$ using $\psi_\beta^5[\cdot]$ takes the form

$$\psi_\beta^5[\tilde{D}] = 0*00*\psi_0[\tilde{\Delta}_0(1)]*01*\psi_1[\tilde{\Delta}_0(2)]*10*\psi_2[\tilde{\Delta}_0(3)]*11*\psi_3[\tilde{\Delta}_0(4)] \tag{5-41}$$

The resulting coded Basic Compressor blocks are shown in Table 5-4. From the table we note that $\mathscr{L}(\psi_2[\tilde{\Delta}_0(3)]) = 56$ bits, two less than the estimate shown in Table 5-3. Thus

$$\mathscr{L}(\psi_\beta^5[\tilde{D}]) = 182 \text{ bits} \tag{5-42}$$

Entropy. From $\tilde{D}$ we obtain the relative frequency of zeroes in $\tilde{D}$ as

$$\bar{p}_0 = \frac{204}{256} = 0.797 \tag{5-43}$$

Table 5-4. Example: Coded Blocks of $\tilde{\Delta}_0$

		Length (Bits)
$\tilde{\Delta}_0(1)$	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	
$\psi_0[\tilde{\Delta}_0(1)]$	000000	6
$\tilde{\Delta}_0(2)$	0,4,11,1,4,1,6,3,1,0,0,0,0,0,0,0	
$\psi_1[\tilde{\Delta}_0(2)]$	10000100000000000001010000101000000010001011111111	47
$\tilde{\Delta}_0(3)$	0,0,0,0,4,5,14,4,0,7,10,3,3,6,1,3	
$FS[\tilde{\Delta}_0(3)]$	111100001000001000000000000000100001100000001000000000001000100010000001010001	
$\psi_2[\tilde{\Delta}_0(3)]$	11111, 110, 100, 0, 100, 0, 0, 0, 0, 100, 0, 11100, 0, 0, 101, 0, 0, 0, 110, 101, 100, 0, 0, 11101, 0, 110	56
$\tilde{\Delta}_0(4)$	0,5,1,5,7,7,14,11,2,0,0,12,10,14,0,0	
$\psi_3[\tilde{\Delta}_0(4)]$	000001010001010101110111111010110010000000000110010101110000000000	64

Then using the binary entropy function in (5-2)

$$H_\beta(\bar{p}_0) = 0.728 \text{ bits/sample} \tag{5-44}$$

By contrast

$$\frac{1}{T}\mathscr{L}(\psi_\beta^5(\tilde{D})) = \frac{182}{256} = 0.711 \text{ bits/sample} \tag{5-45}$$

Optimum decision. It is worth noting that if we had actually used the optimum Basic Compressor decision rule (counting bits) code operator $\psi_2[\cdot]$ instead of $\psi_1[\cdot]$ would have been chosen for block two, $\tilde{\Delta}_0(2)$, resulting in a reduction of coding bits by 3. Note however that the advantage obtained by use of the optimum rule is (by observation) typically much less than this.

Coding of $\tilde{Z}$. Observe that the coding of $\tilde{D}$ in this example completes the coding example for non-binary sources initiated in Fig. 4-5. From (4-26) we have

$$\mathscr{L}(\psi_\theta[\tilde{\theta}]) = 235 \tag{5-46}$$

and using (5-42) we get

$$\mathscr{L}(\psi_9[\tilde{Z}]) = 182 + 235 = 417 \text{ bits} \tag{5-47}$$

for an average of 1.63 bits/sample.

BINARY OPERATORS FOR VERY LOW ENTROPY

Using the ideal memoryless model with constant p_0 as a practical guide, note that the average performance of $\psi_\beta^4[\cdot]$ or $\psi_\beta^5[\cdot]$ in Fig. 5-7 remains within about 0.1 bits/sample of the entropy $H_\beta(p_0)$ for $0 \le p_0 \le 1$. At $p_0 = 0$ and

$p_0 = 1$ the performance "bottoms out" at 0.095 bits/sample for the example shown. As entropy decreases from a maximum of 1.0, this (approximately) 0.1 bits/sample difference represents an increasing fraction of $H_\beta(p_0)$. In this section, we seek to improve the efficiency at these low entropy values.

A closer look at the distribution of the non-binary $\tilde{\Delta}_\zeta$ samples as $p_\zeta \to 1$ (see 5-5) reveals the same situation characterized by Fig. 4-1: a spike at zero, dominating the distribution. This clearly suggests that the approach taken in Chapter IV should be directly applicable to the coding of the $\tilde{\Delta}_\zeta$ and hence binary sequence $\tilde{D}$. This is precisely the approach we will take. It is suggested that the reader review the operator structure developed in Chapter IV before proceeding.

<u>Introduction to Operators $\psi_\beta^9[\cdot]$ and $\psi_{\beta'}^9[\cdot]$</u>

A direct substitution of operator $\psi_9[\cdot]$ from Fig. 4-2 into the block diagrams for $\psi_\beta^j[\cdot]$ and $\psi_{\beta'}^j[\cdot]$ in Figs. 5-5 and 5-6 respectively, provides the two new operators shown in Figs. 5-8 and 5-9. A more elaborate expansion of $\psi_9[\cdot]$ is provided in Fig. 5-8 for discussion purposes. At the same time, we have taken the liberty to expand the notation in an obvious way.

Recall that any operator indicated by the notation $\psi_a^b[\cdot]$ is not completely specified without reference to a "parameter string" which identifies its internal parameters such as input block lengths, decision rules, internal code operators, etc. The $\psi_a^b[\cdot]$ really identify a code operator structure.

With this in mind we can make some observations about $\psi_\beta^9[\cdot]$ and $\psi_{\beta'}^9[\cdot]$.

<u>Operator form</u>. From Figs. 5-8 and 5-9 we have

$$\psi_\beta^9[\tilde{D}] = \zeta * \psi_\beta^j[\tilde{D}_\zeta] * \psi_\theta[\tilde{\Theta}_\zeta] \tag{5-48}$$

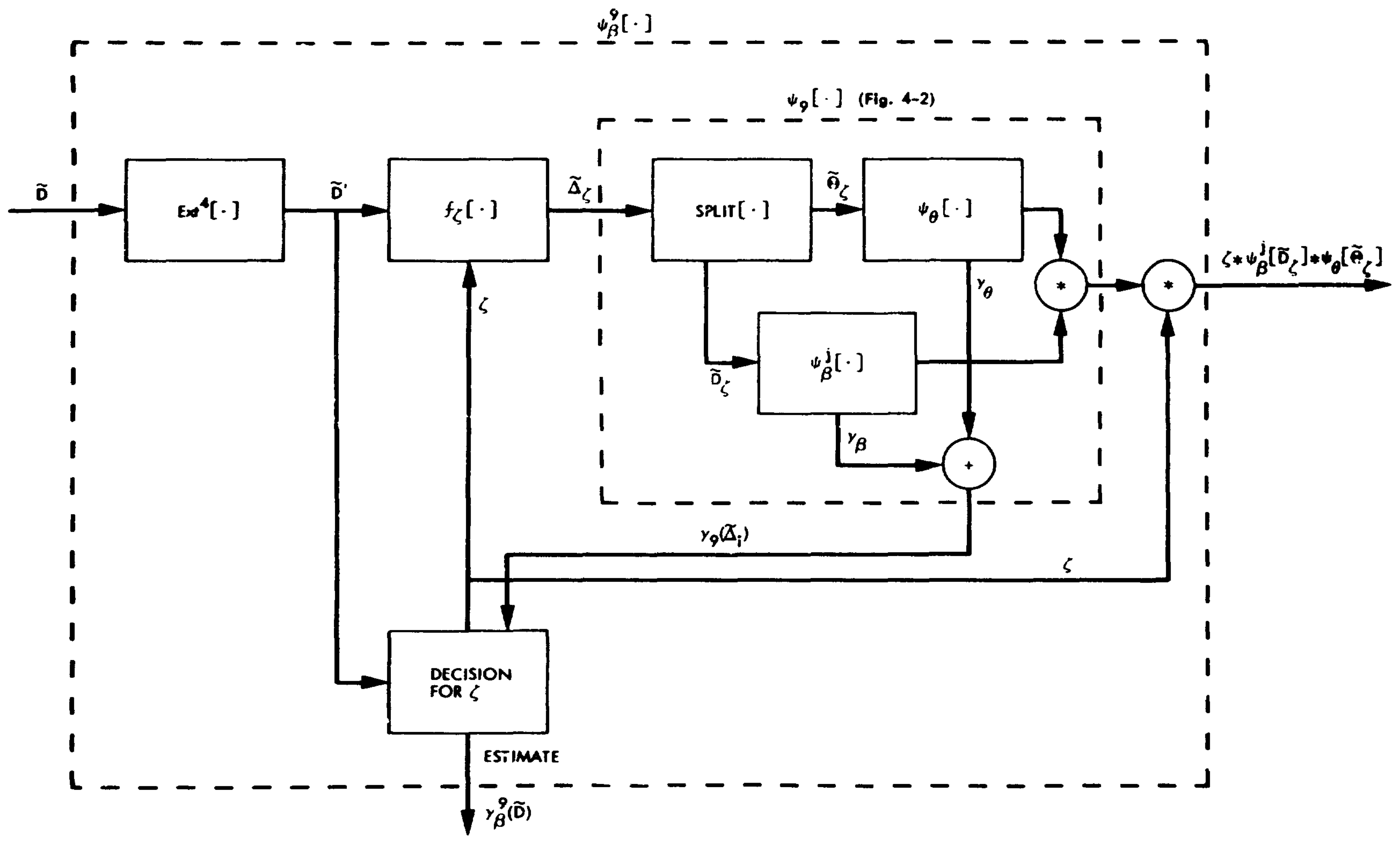


Fig. 5-8. Block Diagram $\psi_\beta^9[\cdot]$

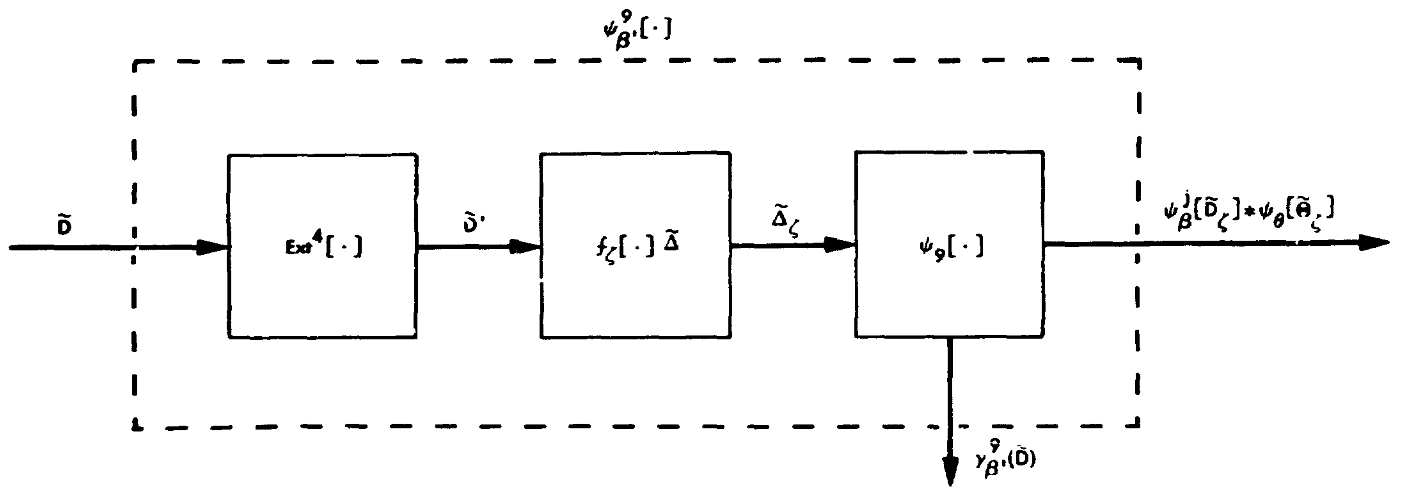

$\psi^{9}_{\beta'}[\cdot]$
$\tilde{D}$
$Ext^{4}[\cdot]$
$\tilde{\delta}'$
$f_{\zeta}[\cdot]\ \tilde{\Delta}$
$\tilde{\Delta}_{\zeta}$
$\psi_{9}[\cdot]$
$\psi^{j}_{\beta}[\tilde{D}_{\zeta}]*\psi_{\theta}[\tilde{\Theta}_{\zeta}]$
$\gamma^{9}_{\beta'}(\tilde{D})$

Fig. 5-9. Block Diagram Operator $\psi^{9}_{\beta'}[\cdot]$

and

$$\psi^{9}_{\beta'}[\tilde{D}] = \psi^{j}_{\beta}[\tilde{D}_{\zeta}] * \psi_{\theta}[\tilde{\Theta}_{\zeta}] \qquad (5\text{-}49)$$

where we note that $\psi^{9}_{\beta}[\cdot]$ and $\psi^{9}_{\beta'}[\cdot]$ differ only in the fact that ζ is chosen during operation by $\psi^{9}_{\beta}[\cdot]$ whereas it is preselected for $\psi^{9}_{\beta'}[\cdot]$.

Block Lengths. With $T = \mathscr{L}(\tilde{D})$ we have, by (5-3) and (5-4) $\mathscr{L}(\tilde{\Delta}_{\zeta}) = \tau = T/4 = \mathscr{L}(\tilde{D}_{\zeta})$.

Requirements for $\psi_{\theta}[\cdot]$. Since the samples of $\tilde{\Delta}_{\zeta}$ and hence $\tilde{\Theta}_{\zeta}$ are 4-bit numbers there is no need for split-sample modes by operator $\psi_{\theta}[\cdot]$. $\psi_{\theta}[\cdot]$ can be replaced by a variable length version of the Basic Compressor, $\psi_{6}[\cdot]$ in (2-70).

Operator $\psi^{j}_{\beta}[\cdot]$. Now consider the block labeled $\psi^{j}_{\beta}[\cdot]$ within $\psi_{9}[\cdot]$ in Fig. 5-8. Just as in the original definition of $\psi_{9}[\cdot]$, $\psi^{j}_{\beta}[\cdot]$ can in general be any binary operator (structure) for coding the T/4 sample sequence $\tilde{D}_{\zeta}$, including $\psi^{9}_{\beta}[\cdot]$ or $\psi^{9}_{\beta'}[\cdot]$ which we are currently discussing. Thus the block diagrams in Figs. 5-8 and 5-9 actually describe an infinite class of possible operators (e.g., the parameter string for any $\psi_{9}[\cdot]$ could specify $\psi^{9}_{\beta}[\cdot]$ as its internal binary operator $\psi^{j}_{\beta}[\cdot]$, which leads to another $\psi_{9}[\cdot]$ and so on). The usual theoretical concerns for such infinite classes is not our main concern in this paper since our interest here is to provide code operators for practical use. However we will return to the possibility of expanding $\psi^{j}_{\beta}[\cdot]$ in this manner later.

Expected Performance for $\psi_\beta^9[\cdot]$ and $\psi_{\beta'}^9[\cdot]$

From (5-48) and Fig. 5-8, the per sample performance of $\psi_\beta^9[\cdot]$ can be written as

$$\frac{1}{T}\mathscr{L}(\psi_\beta^9[\tilde{D}]) = \frac{1}{T} + \min_\zeta \frac{1}{4}\left\{\frac{\mathscr{L}(\psi_\theta\ [\tilde{\theta}_\zeta])}{\mathscr{L}(\tilde{D}_\zeta)} + \frac{\mathscr{L}(\psi_\beta^j[\tilde{D}_\zeta])}{\mathscr{L}(\tilde{D}_\zeta)}\right\} \tag{5-50}$$

where $\mathscr{L}(\tilde{D}_\zeta) = T/4$. Then under the assumption that all the T samples of $\tilde{D}$ are the result of a binary memoryless source with a priori unknown but constant p_0 (over $\tilde{D}$), the expected performance can be bounded by

$$\frac{1}{T}E\left\{\mathscr{L}(\psi_\beta^9[\tilde{D}])\right\} \leq \Lambda_\beta^9\left(p_0,\ T\right) = \frac{1}{T} + \frac{1}{4}\left\{\Lambda_\theta\left(a,\ \frac{T}{4}\right) + \Lambda_\beta^j\left(b,\ \frac{T}{4}\right)\right\} \tag{5-51}$$

where

$$\Lambda_\theta\left(a,\ \frac{T}{4}\right) \geq \frac{4}{T}E\left\{\mathscr{L}(\psi_\theta[\tilde{\theta}_\zeta])\right\} \tag{5-52}$$

$$\Lambda_\beta^j\left(b,\ \frac{T}{4}\right) \geq \frac{4}{T}E\left\{\mathscr{L}(\psi_\beta^j[\tilde{D}_\zeta])\right\} \tag{5-53}$$

and

$$a = \max\ (p_0,\ 1 - p_0) \tag{5-54}$$

and

$$b = a^4 \tag{5-55}$$

Similarly, we note that without the need to identify ζ

$$\frac{1}{T}E\left\{\mathscr{L}\left(\psi_{\beta'}^9[\tilde{D}]\right)\right\} \leq \Lambda_{\beta'}^9\left(p_0,\ \frac{T}{4}\right) = \Lambda_\beta^9\left(p_0,\ \frac{T}{4}\right) - \frac{1}{T} \tag{5-56}$$

The term Λ_θ in (5-52) bounds the expected bits (per $\tilde{\Delta}_\zeta$ sample) to code the variable length sequence $\tilde{\Theta}_\zeta$. The details are provided in Appendix B assuming that $\psi_\theta[\cdot] = \psi_6[\cdot]$ and that $\tilde{\Theta}_\zeta$ is treated as a single Basic Compressor block.

The expression Λ_β^j in (5-53) bounds the expected performance of a binary operator $\psi_\beta^j[\cdot]$ on T/4 sample sequence $\tilde{D}_\zeta$ with a priori unknown but constant probability of a zero given by b in (5-55). Thus a complete determination of expected performance for $\psi_\beta^9[\cdot]$ requires that internal binary operator $\psi_\beta^j[\cdot]$ be specified. In general, $\psi_\beta^j[\cdot]$ could again be any binary operator, including the one we are currently investigating, $\psi_\beta^9[\cdot]$. For example, if we let the internal binary operator of $\psi_\beta^9[\cdot]$ be $\psi_\beta^9[\cdot]$, the bound in (5-51) could be expanded to

$$\Lambda_\beta^9(p_0, T) = \frac{1}{T} + \frac{1}{4}\left\{\Lambda_\theta\left(a, \frac{T}{4}\right) + \Lambda_\beta^9\left(b, \frac{T}{4}\right)\right\}$$

where

$$\Lambda_\beta^9\left(b, \frac{T}{4}\right) = \frac{1}{T/4} + \frac{1}{4}\left\{\Lambda_\theta\left(a', \frac{T}{16}\right) + \Lambda_\beta^j\left(b', \frac{T}{16}\right)\right\} \tag{5-57}$$

and by (5-54) and (5-55)

$$a' = \max(b,\ 1 - b) \tag{5-58}$$

and

$$b' = a'^4 \tag{5-59}$$

Clearly the expected performance of a large class of binary operators can be derived by appropriate substitution of parameters.

Example. An interesting example is provided by using the basic binary operators $\psi_\beta^4[\cdot]$ or $\psi_\beta^5[\cdot]$ in the internal structure of $\psi_\beta^9[\cdot]$. The expected performance of these operators was investigated earlier and displayed graphically in Fig. 5-7. Using $\psi_\beta^4[\cdot]$ as the internal binary operator of $\psi_\beta^9[\cdot]$ yields the results shown in Fig. 5-10 assuming an input block length of T = 256. The graph was obtained by replacing the last term in (5-51) by $\Lambda_\beta^4(b, 64)$. The corresponding results for $\psi_\beta^4[\cdot]$ directly operating on $\tilde{D}$ are repeated for comparison purposes.

Introduction to Operators $\psi_\beta^{10}[\cdot]$ and $\psi_{\beta'}^{10}[\cdot]$

By Fig. 5-10 $\psi_5[\cdot]$ operating on $\tilde{\Delta}_\zeta$ performs better than $\psi_y[\cdot]$ for a range of intermediate values of p_0. Following our usual procedure for such situations we can simply choose between these operators. This amounts to replacing $\psi_j[\cdot]$ internal to $\psi_\beta^j[\cdot]$ (Fig. 5-5) and $\psi_{\beta'}^j[\cdot]$ (Fig. 5-6) by $\psi_{10}[\cdot]$. The resulting block diagrams of binary operators $\psi_\beta^{10}[\cdot]$ and $\psi_{\beta'}^{10}[\cdot]$ are shown in Figs. 5-11 and 5-12.

Operator form. From the figures we have

$$\psi_\beta^{10}[\tilde{D}] = \zeta*\lambda_1*\psi_{\lambda_1}[\tilde{\Delta}_\zeta] \tag{5-60}$$

and

$$\psi_{\beta'}^{10}[\tilde{D}] = \lambda_1*\psi_{\lambda_1}[\tilde{\Delta}_\zeta] \tag{5-61}$$

where λ_1 is either 9 or 5.† Again the only difference between $\psi_\beta^{10}[\cdot]$ and $\psi_{\beta'}^{10}[\cdot]$ is that ζ is chosen during operation by $\psi_\beta^{10}[\cdot]$ whereas it must be a priori selected for $\psi_{\beta'}^{10}[\cdot]$.

†$\psi_5[\cdot]$ or $\psi_4[\cdot]$ replaces the $\psi_8[\cdot]$ assumed in Fig. 4-3 here since the alphabet size of $\tilde{\Delta}_\zeta$ is only 16 so that no split-sample modes are needed. Recall that these operators are really special cases of $\psi_8[\cdot]$.

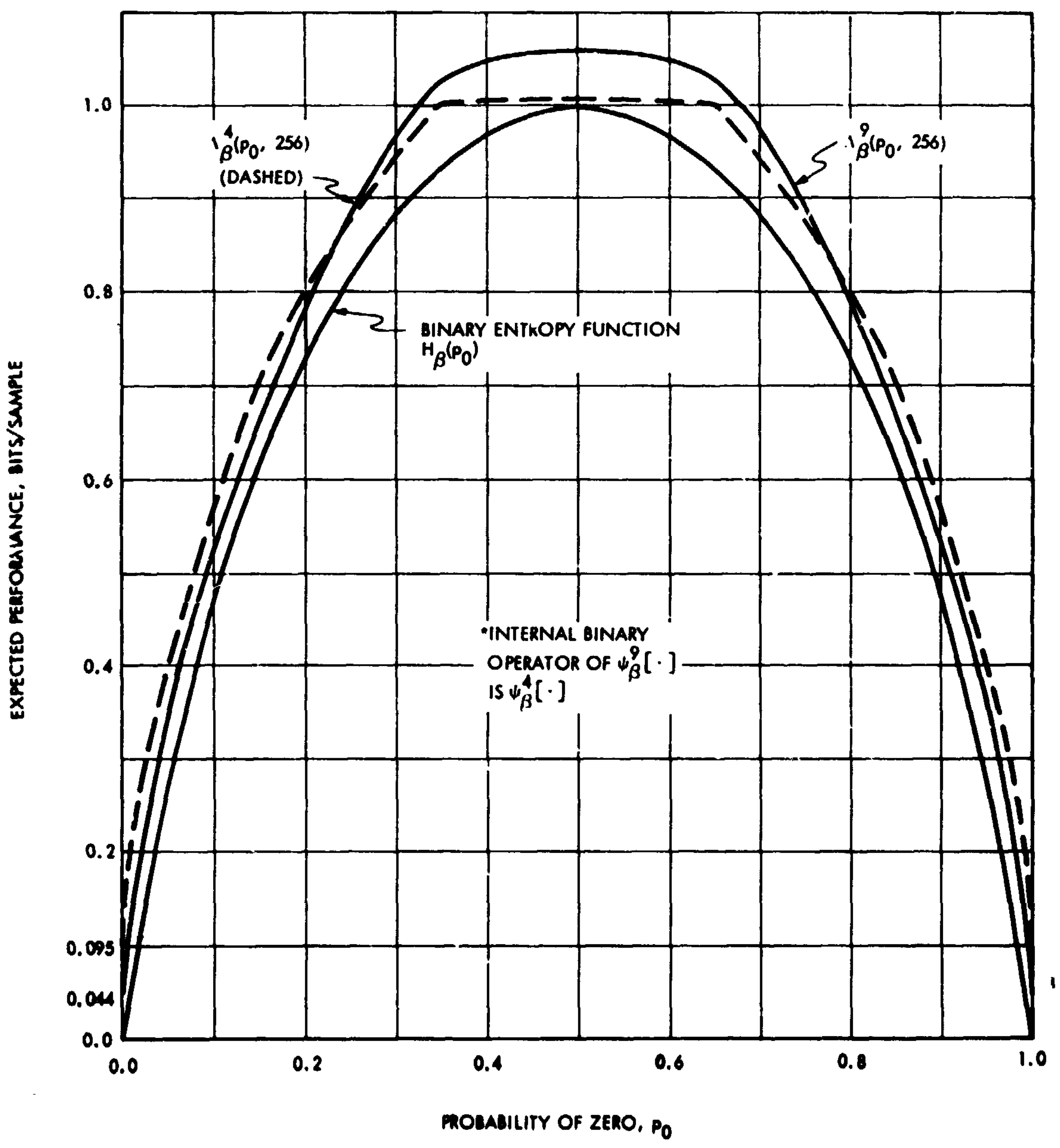
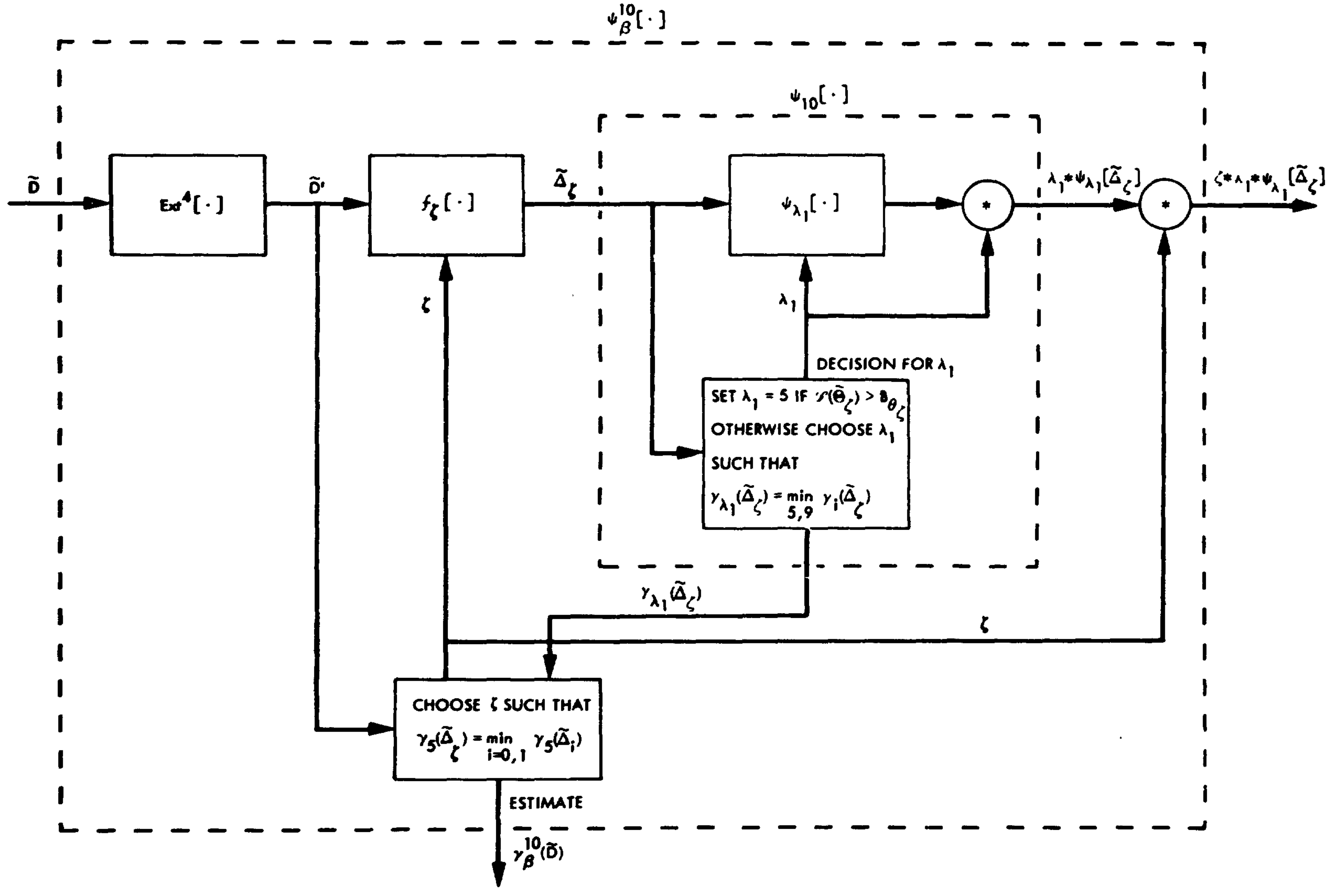


Fig. 5-10. Bounds to Expected Performance of $\psi_\beta^9[\cdot]$ on Ideal Memoryless Source, Unknown but Constant p_0



$\psi_\beta^{10}[\cdot]$
$\psi_{10}[\cdot]$
$\tilde{D}$
$Ext^4[\cdot]$
$\tilde{D}'$
$f_\zeta[\cdot]$
$\tilde{\Delta}_\zeta$
$\psi_{\lambda_1}[\cdot]$
$*$
$\lambda_1 * \psi_{\lambda_1}[\tilde{\Delta}_\zeta]$
$*$
$\zeta * \lambda_1 * \psi_{\lambda_1}[\tilde{\Delta}_\zeta]$
ζ
λ_1
DECISION FOR λ_1
SET $\lambda_1 = 5$ IF $\mathcal{T}(\tilde{\Theta}_\zeta) > B_{\theta_\zeta}$
OTHERWISE CHOOSE λ_1
SUCH THAT
$\gamma_{\lambda_1}(\tilde{\Delta}_\zeta) = \min_{5,9} \gamma_i(\tilde{\Delta}_\zeta)$
$\gamma_{\lambda_1}(\tilde{\Delta}_\zeta)$
ζ
CHOOSE ζ SUCH THAT
$\gamma_5(\tilde{\Delta}_\zeta) = \min_{i=0,1} \gamma_5(\tilde{\Delta}_i)$
ESTIMATE
$\gamma_\beta^{10}(\tilde{D})$

Fig. 5-11. Basic Block Diagram Operator $\psi_\beta^{10}[\cdot]$

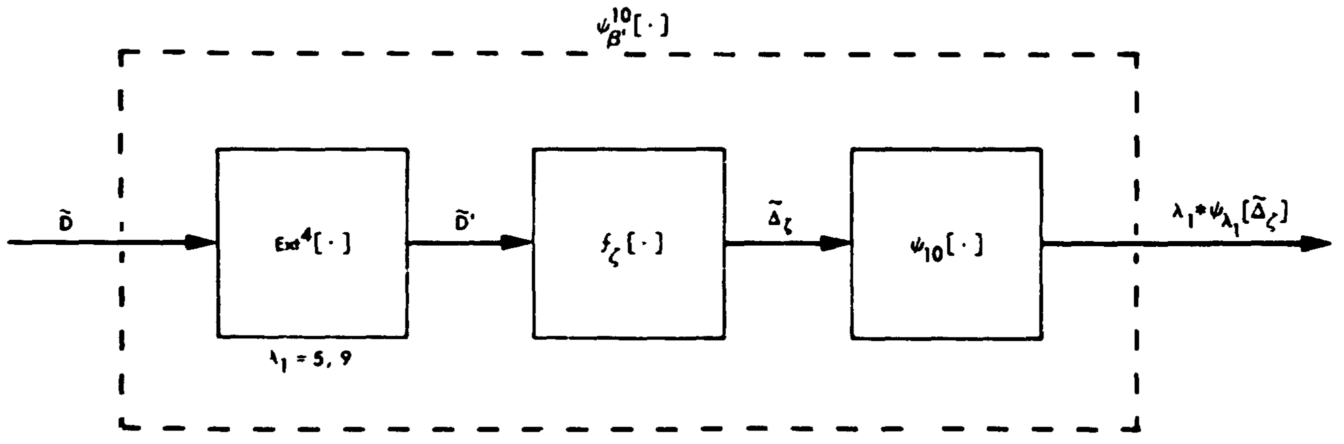


Fig. 5-12. Basic Block Diagram Operator $\psi^{10}_{\beta'}[\cdot]$

Decision rules for ζ. A selection of decision rules for choosing ζ in $\psi_\beta^{10}[\cdot]$ is described in (5-19) - (5-22). The simplified rule of (5-20) is preferred in this case, and can be further simplified.

A direct application of (5-20) would require that ζ be chosen such that

$$\gamma_{10}(\tilde{\Delta}_\zeta) = \min_{i=0,1} \gamma_{10}(\tilde{\Delta}_i) \tag{5-62}$$

But by considerations of expected performance of $\psi_\beta^5[\cdot]$ and $\psi_\beta^9[\cdot]$ just investigated and practical observation, whenever the internal decision rule of $\psi_{10}[\cdot]$ would choose operator $\psi_9[\cdot]$ instead of $\psi_5[\cdot]$ we would also find that

$$\gamma_9(\tilde{\Delta}_\zeta) < \gamma_5(\tilde{\Delta}_\zeta) < \gamma_{10}(\tilde{\Delta}_{\bar{\zeta}}) \tag{5-63}$$

where $\bar{\zeta} = \text{COMP}[\zeta]$. Thus there is no need to consider $\psi_9[\cdot]$ in the determination of ζ. The decision rule in (5-62) reduces to choosing ζ such that

$$\gamma_5(\tilde{\Delta}_\zeta) = \min_{i=0,1} \gamma_5(\tilde{\Delta}_i) \tag{5-64}$$

Internal decision rules. As in Chapter IV the decision rules for selecting either $\psi_5[\cdot]$ or $\psi_9[\cdot]$ may be simplified in some applications to reduce the required buffer size for $\tilde{\Theta}_\zeta$ samples. Following (4-24) and (4-25) the rule for choosing λ_1 is:

Choose $\lambda_1 = 5$ if $\mathscr{S}(\tilde{\Theta}_\zeta) \geq B_{\theta_\zeta}$

Otherwise choose λ_1 such that

$$\gamma_{\lambda_1}(\tilde{\Delta}_\zeta) = \min_{i=5,9} \gamma_i(\tilde{\Delta}_\zeta) \tag{5-65}$$

where $\tilde{\Theta}_\zeta$ buffer size B_{θ_ζ} can be experimentally chosen so that the loss in performance is acceptable. Basically, this rule forces a decision to use $\psi_\beta^5[\cdot]$ at intermediate values of p_0. By Fig. 5-10 $\psi_5[\cdot]$ is either the best choice or a good second best for a wide range of p_0.

Choosing B_{θ_ζ} quite small means the $\psi^9[\cdot]$ is used only to improve performance for very high or low values of p_0. Under these conditions an additional simplification results because the internal $\psi_\beta^9[\cdot]$ will never be chosen unless $p_\zeta^4 > 1/2$. Thus $\psi_\beta^9[\cdot]$ can be replaced by $\psi_{\beta'}^9[\cdot]$. The particular options chosen for implementation will depend on the particular application which may not have such ideal stationary statistics.

Expected Performance. A bound to the expected performance of operators $\psi_\beta^{10}[\cdot]$ and $\psi_{\beta'}^{10}[\cdot]$, assuming $\tilde{D}$ arises from an ideal binary memoryless source with unknown but constant p_0, is easily obtained from previous results. We have

$$\frac{1}{T} E\left\{\psi_\beta^{10}[\tilde{D}]\right\} \le \Lambda_\beta^{10}(p_0, T) \tag{5-66}$$

where

$$\Lambda_\beta^{10}(p_0, T) = \frac{2}{T} + \min\left\{\Lambda_{\beta'}^4(p_\zeta, T), \Lambda_{\beta'}^9(p_\zeta, T)\right\} \tag{5-67}$$

and as usual

$$\zeta = \begin{cases} 0 & \text{if } p_0 \ge 1/2 \\ 1 & \text{otherwise} \end{cases} \tag{5-68}$$

Again we have assumed that $\psi_{\beta'}^5[\cdot]$ incorporates a single Basic Compressor block making it equivalent to $\psi_{\beta'}^4[\cdot]$. A graph of $\Lambda_\beta^{10}(p_0, 256)$ would essentially be the lower envelope of the curves in Fig. 5-10 provided $\psi_9[\cdot]$ is terminated after one SPLIT$[\cdot]$ operation and the $\tilde{\Theta}_\zeta$ buffer is not restricted.

Similarly, without the need to identify ζ $\Lambda_{\beta'}^{10}(p_0, T)$ is given by

$$\Lambda_{\beta'}^{10}(p_0, T) = \Lambda_\beta^{10}(p_0, T) - \frac{1}{T} \tag{5-69}$$

Multi-Level $\psi_{\beta}^{10}[\cdot]$ and $\psi_{\beta'}^{10}[\cdot]$

As noted earlier any operator structure which includes $\psi_9[\cdot]$ in its definition really defines an infinite class of operators. This is because the SPLIT[·] operation results in a new internal binary source which must be coded by some binary operator $\psi_{\beta}^{j}[\cdot]$. Since $\psi_{\beta}^{j}[\cdot]$ may again contain the SPLIT[·] operation the procedure can be repeated over and over again. But the number of binary samples decreases by four at each branch into this "tree of code operators." Thus the incremental impact on computation required for decision making at each step diminishes rapidly.

We provide an illustration of the procedures by expanding $\psi_{\beta}^{10}[\cdot]$ several times yielding an operator which would code a T = 1024 bit all zero or all ones sequence with only 11 bits. A block diagram is provided in Fig. 5-13 where we have taken the liberty to expand notation in an obvious way. Additionally, we have also elected to accept some slight loss in performance for this example by choosing internal binary operators $\psi_{\beta'}^{10}[\cdot]$ and $\psi_{\beta'}^{5}[\cdot]$ in the expansion instead of $\psi_{\beta}^{10}[\cdot]$ and $\psi_{\beta}^{5}[\cdot]$. By earlier discussions (see 5-65) the $\tilde{\Theta}_{\zeta}$ and $\tilde{\Theta}_{\zeta,0}$ buffers in Fig. 5-13 can be substantially reduced.

Expansion of $\psi_{\beta}^{10}[\cdot]$. The creation of multilevel operators such as $\psi_{\beta}^{10}[\cdot]$ in Fig. 5-13 is accomplished by continually replacing the binary operator $\psi_{\beta}^{j}[\cdot]$ which follows a SPLIT[·] by another which also includes a SPLIT[·]. For the example in Fig. 5-13, we have

$$\psi_{\beta}^{10}[\tilde{D}] = \zeta * \lambda_1 * \psi_{\lambda_1}[\tilde{\Delta}_{\zeta}] \tag{5-70}$$

where λ_1 equals 5 or 9 just as in Fig. 5-11. If λ_1 = 9 we would have

$$\psi_{\beta}^{10}[\tilde{D}] = \zeta * \lambda_1 * \psi_{\beta'}^{10}[\tilde{D}_{\zeta}] * \psi_{\theta}[\tilde{\Theta}_{\zeta}] \tag{5-71}$$

since we have used $\psi_{\beta'}^{10}[\cdot]$ as the internal binary operator of $\psi_9[\cdot]$.

Expanding $\psi_{\beta'}^{10}[\cdot]$ we have

$$\psi_{\beta'}^{10}[\tilde{D}_\zeta] = \lambda_2 * \psi_{\lambda_2}[\tilde{D}_\zeta] \tag{5-72}$$

where again λ_2 equals 5 or 9. If $\lambda_2 = 9$

$$\psi_{\beta'}^{10}[\tilde{D}_\zeta] = \lambda_2 * \psi_{\beta'}^{5}[\tilde{D}_{\zeta,0}] * \psi_\theta[\tilde{\theta}_{\zeta,0}] \tag{5-73}$$

since we have terminated expansion by using $\psi_{\beta'}^{5}[\cdot]$ as the binary operator for $\tilde{D}_{\zeta,0}$ (i.e., $\psi_{\beta'}^{5}[\cdot]$ contains no SPLIT[·] operation).[†]

If the code decisions are actually $\lambda_1 = \lambda_2 = 5$ (very low or high p_0) the final expanded form of $\psi_\beta^{10}[\tilde{D}]$ is given as

$$\psi_\beta^{10}[\tilde{D}] = \zeta * \lambda_1 * \lambda_2 * \psi_{\beta'}^{5}[\tilde{D}_{\zeta,0}] * \psi_\theta[\tilde{\theta}_{\zeta,0}] * \psi_\theta[\tilde{\theta}_\zeta] \tag{5-74}$$

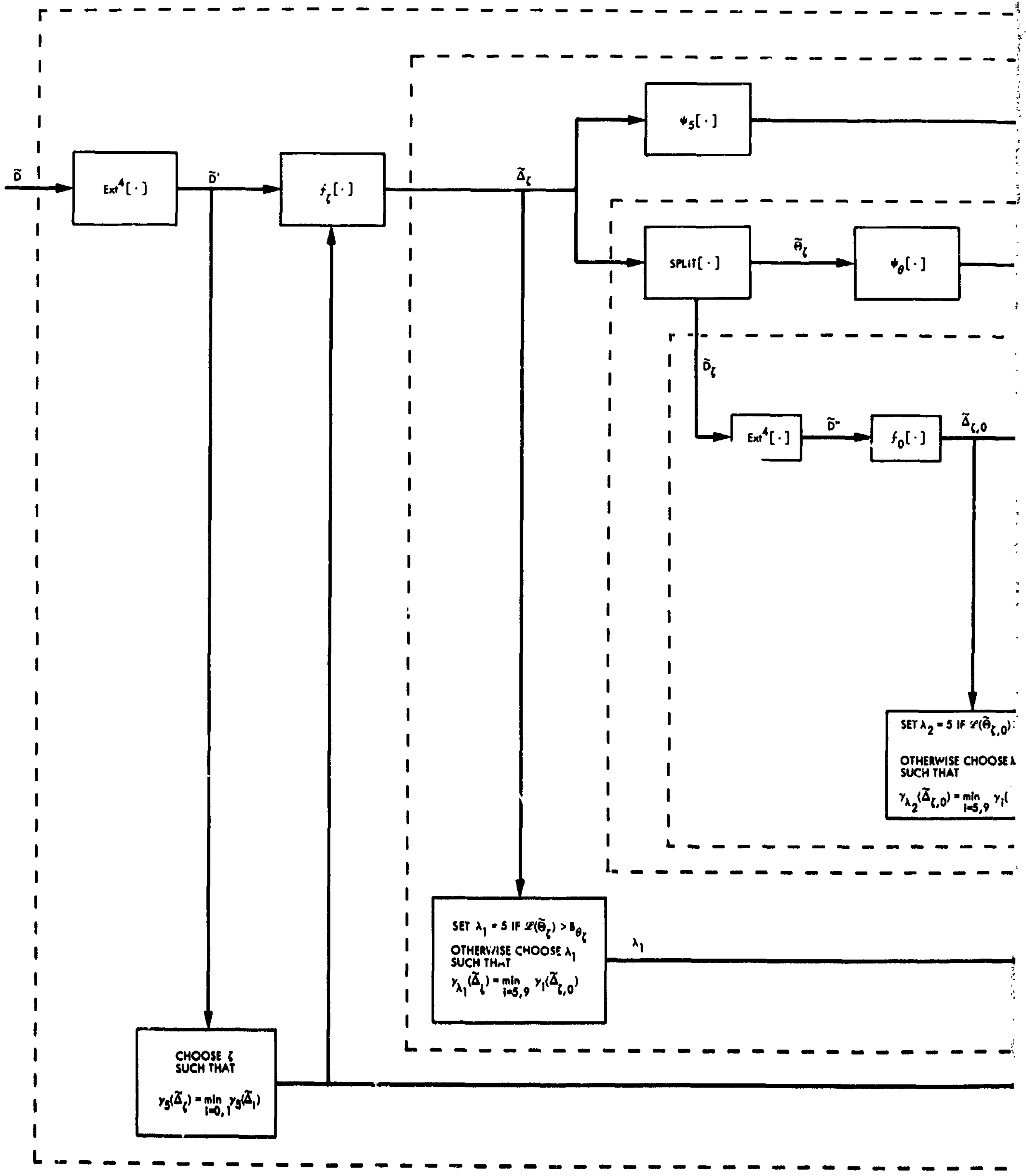
<u>All zero input sequence</u>. If we take the length of $\tilde{D}$ as T = 1024 and assume it is either all zeroes or all ones, $\psi_\beta^{10}[\tilde{D}]$ will take the form in (5-74) with both $\psi_\theta[\tilde{\theta}_{\zeta,0}]$ and $\psi_\theta[\tilde{\theta}_\zeta]$ contributing zero bits. Since $\tilde{\Delta}_{\zeta,0,0}$ reduces to a 16 sample all zeroes sequence, $\psi_{\beta'}^{5}[\tilde{D}_{\zeta,0}]$ will require only 8 bits. Adding in the 3 bits for ζ, λ_1 and λ_2 we have

$$\mathscr{L}(\psi_\beta^{10}[\tilde{D} = \text{all zeroes or all ones}]) = 11 \text{ bits} \tag{5-75}$$

or approximately 0.01 bits/sample.

[†]Note that using $\psi_\beta^{10}[\cdot]$ instead would change (5-73) to the form

$$\psi_\beta^{10}[\tilde{D}_\zeta] = \lambda_2 * \zeta_1 * \psi_{\beta'}^{5}[\tilde{D}_{\zeta,\zeta_1}] * \psi_\theta[\tilde{\theta}_{\zeta,\zeta_1}] \tag{5-76}$$



$\tilde{D}$
$\mathrm{Ext}^4[\cdot]$
$\tilde{D}'$
$f_\zeta[\cdot]$
$\tilde{\Delta}_\zeta$
$\psi_5[\cdot]$
SPLIT$[\cdot]$
$\tilde{\theta}_\zeta$
$\psi_\theta[\cdot]$
$\tilde{D}_\zeta$
$\mathrm{Ext}^4[\cdot]$
$\tilde{D}''$
$f_0[\cdot]$
$\tilde{\Delta}_{\zeta,0}$
SET $\lambda_2 = 5$ IF $\mathcal{L}(\tilde{\theta}_{\zeta,0})$
OTHERWISE CHOOSE λ
SUCH THAT
$\gamma_{\lambda_2}(\tilde{\Delta}_{\zeta,0}) = \min_{i=5,9} \gamma_i($
SET $\lambda_1 = 5$ IF $\mathcal{L}(\tilde{\theta}_\zeta) > B_{\theta_\zeta}$
OTHERWISE CHOOSE λ_1
SUCH THAT
$\gamma_{\lambda_1}(\tilde{\Delta}_\zeta) = \min_{i=5,9} \gamma_i(\tilde{\Delta}_{\zeta,0})$
λ_1
CHOOSE ζ
SUCH THAT
$\gamma_5(\tilde{\Delta}_\zeta) = \min_{i=0,1} \gamma_5(\tilde{\Delta}_i)$

FOLDOUT FRAME 1

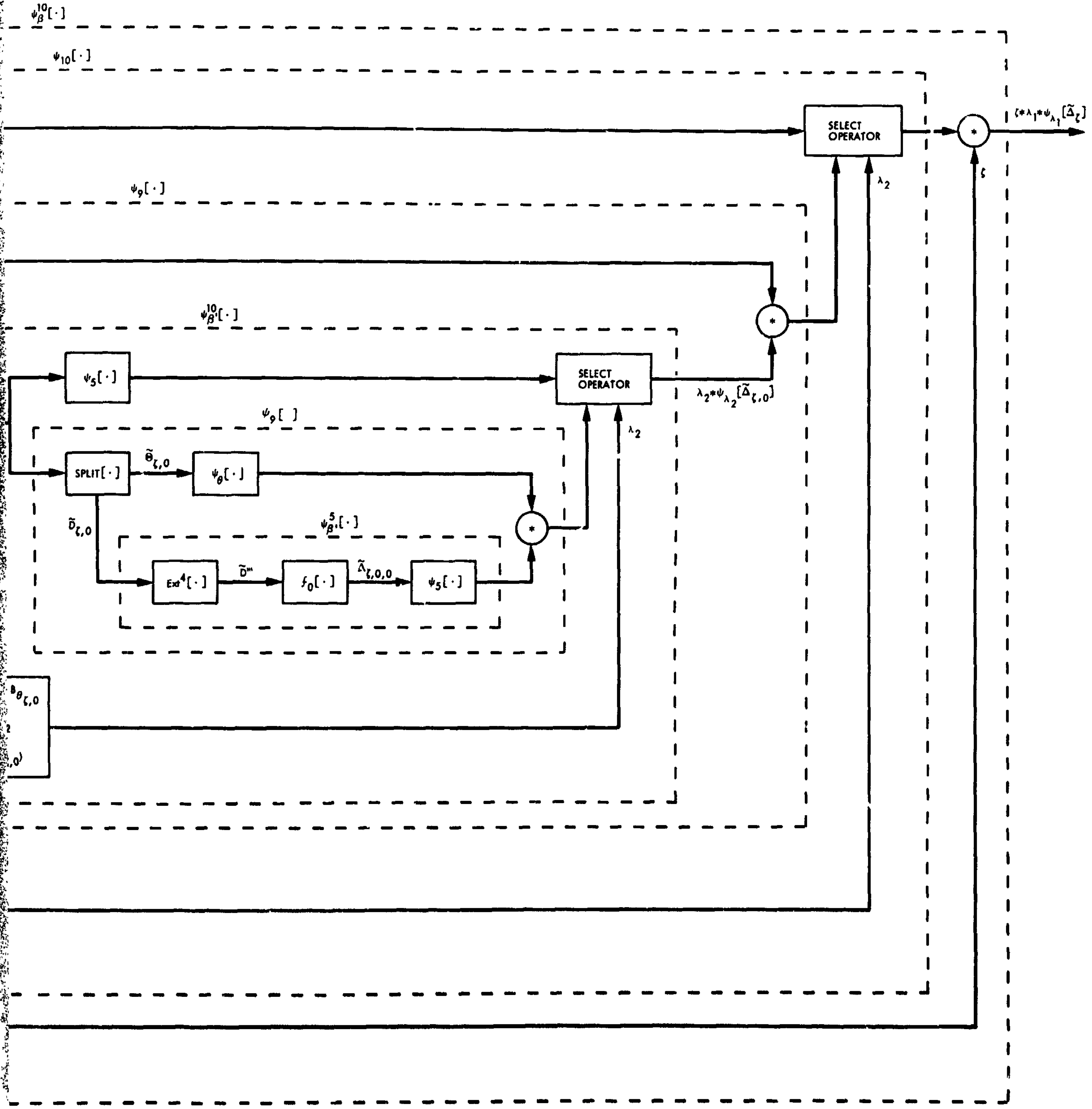


Fig. 5-13. Block Diagram Multi-Level $\psi_\beta^{10}[\cdot]$

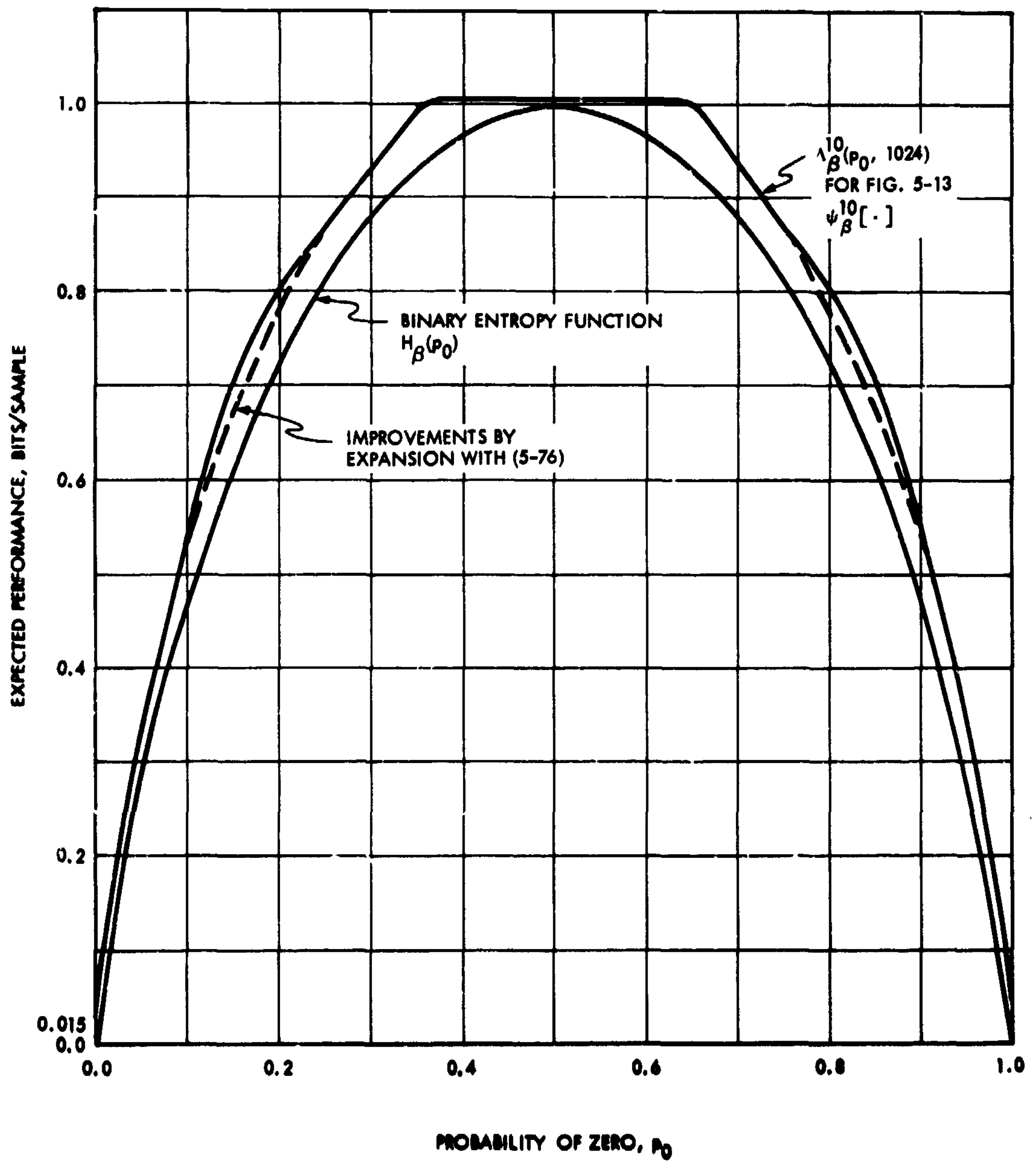


Fig. 5-14. Bounds to Expected Performance of Multi-Level $\psi_\beta^{10}[\cdot]$ on Ideal Memoryless Source, Unknown but Constant p_0

PRECEDING PAGE BLANK NOT FILMED

Additional Adaptivity

In many real problems p_0 may not only vary but the manner in which it changes may also vary. Whereas the algorithms described in this chapter will typically perform well under a measured $H_\beta(\bar{p}_0)$ in these situations, in many cases, it may be desirable to provide additional adaptivity. Such desirable modifications and extensions of the preceding developments will be the subject of future reports.

APPENDIX A
STATISTICAL PERFORMANCE BOUNDS FOR $\psi_\beta^4[\cdot]$ AND $\psi_\beta^5[\cdot]$

Here we develop bounds on the expected performance of binary operators $\psi_\beta^4[\cdot]$ and $\psi_\beta^5[\cdot]$ on T sample binary sequences $\tilde{D}$ (5-1) under the assumption that the samples of $\tilde{D}$ are the result of an ideal binary memoryless source with a priori unknown but constant probability of a zero, p_0, where $0 \le p_0 \le 1$. Specifically, we develop the result

$$\frac{1}{T} E\{\mathscr{L}(\psi_\beta^j[\tilde{D}])\} \le \Lambda_\beta^j(p_0, T, \eta) \text{ bits/sample} \tag{A-1}$$

where η refers to the number of Basic Compressor blocks used by $\psi_j[\cdot]$ to code $\tilde{\Delta}_\zeta$ in Fig. 5-5. Λ_β^j is given by

$$\Lambda_\beta^j(p_0, T, \eta) = \frac{(1+2\eta)}{T} + \frac{1}{T}\gamma_{ID}(\bar{F}^\zeta) \tag{A-2}$$

where

$$\bar{F}^\zeta = \frac{T}{4}\Big[p_\zeta^4 + 14p_\zeta^3(1-p_\zeta) + 51p_\zeta^2(1-p_\zeta)^2 + 54p_\zeta(1-p_\zeta)^3 + 16(1-p_\zeta)^4\Big] \tag{A-3}$$

is a fundamental sequence length for a $J = T/4$ Basic Compressor block and

$$\zeta = \begin{cases} 0 & \text{if } p_0 \ge \frac{1}{2} \\ 1 & \text{otherwise} \end{cases} \tag{A-4}$$

γ_{ID} **is evaluated using (2-27).**

DERIVATION OF Λ_β^j

First observe that operator $\psi_4[\cdot]$ is a special case of operator $\psi_5[\cdot]$ for which only one Basic Compressor block is used ($\eta = 1$). Thus, we can henceforth assume $j = 5$.

As in (2-67) and (2-68), $\tilde{\Delta}_\zeta$ can be partitioned into η Basic Compressor blocks so that

$$\tilde{\Delta}_\zeta = \tilde{\Delta}_\zeta(1) * \tilde{\Delta}_\zeta(2) * \ldots \tilde{\Delta}_\zeta(\eta) \tag{A-5}$$

and where the $\tilde{\Delta}_\zeta(\ell)$ consist of J_i samples satisfying

$$\frac{T}{4} = \sum_{\ell=1}^{\eta} J_\ell \tag{A-6}$$

Now tracing through the appropriate equations, by (5-21)

$$\mathscr{L}(\psi_\beta^5[\tilde{D}]) \leq 1 + \gamma_5[\tilde{\Delta}_\zeta] \tag{A-7}$$

Denoting the fundamental sequence length for $\tilde{\Delta}_\zeta(\ell)$ by F_ℓ^ζ and using (2-29) and (2-74) we have

$$\frac{1}{T} E\left\{\mathscr{L}(\psi_\beta^5[\tilde{D}])\right\} \leq \frac{(1 + 2\eta)}{T} + \frac{1}{T} \sum_{\ell=1}^{\eta} E\left\{\gamma_{ID}(F_\ell^\zeta)\right\} \tag{A-8}$$

where the first term is simply the overhead in identifying ζ and Basic Compressor code options for the η blocks.

But by (2-32) we have

$$\frac{1}{T} \sum_{\ell=1}^{\eta} E\left\{\gamma_{ID}(F^\zeta)\right\} \leq \frac{1}{T} \sum_{\ell=1}^{\eta} \gamma_{ID}(\bar{F}_\ell^\zeta) \tag{A-9}$$

Using the binary memoryless assumption and recalling the mapping of $\tilde{D}$ into $\tilde{\Delta}_\zeta$, $\bar{F}_\ell^\zeta$ is given as

$$\bar{F}_\ell^\zeta = E\left\{F_\ell^\zeta\right\} = J_\ell\Big[p_\zeta^4 + 14p_\zeta^3(1 - p_\zeta) + 51p_\zeta^2(1 - p_\zeta)^2 + 54p_\zeta(1 - p_\zeta)^3 + 16(1 - p_\zeta)^4\Big] \tag{A-10}$$

where quite obviously

$$\bar{F}_\ell^0 \le \bar{F}_\ell^1 \quad \text{for} \quad p_0 \ge \frac{1}{2} \tag{A-11}$$

identifying the choice of ζ as in (A-4).

Substituting F_ℓ^ζ in the $\gamma_k(\cdot)$ of (2-23) - (2-26) we see that each function is a multiple of block size (ignoring truncation). Thus choosing the minimum in (2-28) is independent of block size. Operator decisions, ID, are the same for each of the η blocks. Then the right hand side of (A-9) can more simply be replaced by a single T/4 sample Basic Compressor block with expected fundamental sequence length given by ΣF_ℓ^ζ in (A-3) with ζ determined by (A-4).

Comment

The reduction of the righthand side of (A-9) to a single Basic Compressor block should come as no surprise. Because of the ideal memoryless model, data statistics do not change over $\tilde{D}$. There is no advantage to adapting because the extra code blocks just cost in overhead (i.e., $2\eta/T$ in A-8). However, if data statistics were actually changing over $\tilde{D}$, as in many practical situations, the extra adaptivity might more than make up for the additional overhead.

APPENDIX B

STATISTICAL PERFORMANCE BOUND ON $\psi_\theta[\tilde{\Theta}_\zeta] = \psi_6[\tilde{\Theta}_\zeta]$

Here we provide a bound to the expected bits/sample required to code $\tilde{\Theta}_\zeta$ where $\tilde{\Theta}_\zeta$ is the result of the SPLIT[·] operator on $\tilde{\Delta}_\zeta$ for operator $\psi_\beta^9[\cdot]$ or $\psi_{\beta'}^9[\cdot]$ in Figs. 5-8 and 5-9 respectively, and T-sample input sequence $\tilde{D}$ is assumed to be the result of a binary memoryless source with a priori unknown but constant probability of a zero, p_0 (over $\tilde{D}$). Operator $\psi_\theta[\cdot] = \psi_\zeta[\cdot]$ from (2-70) will be assumed to consist of only one Basic Compressor block, equal in length to $\mathscr{S}(\tilde{\Theta}_\zeta)$.

The desired result is of the form (see B-10)

$$\frac{1}{T/4} E\left\{\mathscr{S}\left(\psi_\theta[\tilde{\Theta}\ ]\right)\right\} \leq \Lambda_\theta\left(p_\zeta, \frac{T}{4}\right) \tag{B-1}$$

where[†]

$$\zeta = \begin{cases} 0 & \text{if } p_0 \geq \frac{1}{2} \\ 1 & \text{otherwise} \end{cases} \tag{B-2}$$

$$p_\zeta = a = \max(p_0, 1 - p_0) \tag{B-3}$$

and

$$p_{\zeta,0} = b = p_\zeta^4 \tag{B-4}$$

[†]We recognize p_ζ as the probability of a zero in any bit position of $\tilde{\Delta}_\zeta$. That is, after the decision ζ. Similarly, $p_{\zeta,0}$ is the probability of an all zero 4-tuple symbol in $\tilde{\Delta}_\zeta$. The additional notation a, b is provided to allow easier usage in the main text.

DERIVATION OF $\Lambda_\theta(p_\zeta,\ T/4)$

By (2-28) and (2-39)

$$E\left\{\mathscr{L}\left(\psi_\theta[\tilde{\Theta}_\zeta]\right)\right\} \le 2 + \gamma_{ID}(\bar{F}_\theta,\ \bar{J}_\theta) \tag{B-5}$$

where

$$\bar{F}_\theta = E\left\{\mathscr{L}\left(\psi_1[\tilde{\Theta}_\zeta]\right)\right\} \tag{B-6}$$

and as in (4-9)

$$\bar{J}_\theta = E\left\{\mathscr{L}\left(\tilde{\Theta}_\zeta\right)\right\} = \frac{T}{4}\,(1 - p_{\zeta,0}) \tag{B-7}$$

$\underline{\bar{F}_\theta}$

The expression in (B-3) is easily evaluated by noting that a sample of $\tilde{\Delta}_\zeta$ of value i will contribute i bits to the fundamental sequence for $\tilde{\Theta}_\zeta$. Then by summing the expected contributions for all T/4 samples we have

$$\bar{F}_\theta = \frac{T}{4}\Big[10p_\zeta^3(1 - p_\zeta) + 45p_\zeta^2(1 - p_\zeta)^2 + 50p_\zeta(1 - p_\zeta)^3 + 15(1 - p_\zeta)^4\Big]$$

$\bar{F}_\theta$ can be written into a more convenient form by factoring out $p_{\zeta,0}$ yielding

$$\bar{F}_\theta = \bar{J}_\theta\,\Omega_\zeta \tag{B-8}$$

where

$$\Omega_\zeta = \left(\frac{1 - p_\zeta}{1 - p_{\zeta,0}}\right)\Big\{10p_\zeta^3 + 45p_\zeta^2(1 - p_\zeta) + 50p_\zeta(1 - p_\zeta)^2 + 15(1 - p_\zeta)^3\Big\} \tag{B-9}$$

Bound Expression

Substituting (B-3) and (B-6) into (B-2) using (2-23) - (2-27)[†] we get after simplification

$$\frac{T}{4}\Lambda_\theta(p_\zeta, \frac{T}{4}) = 2 + \bar{J}_\theta \min\left\{\frac{7}{3}\Omega_\zeta - 2, \Omega_\zeta, \frac{\Omega_\zeta}{3} + 2, 4\right\} \qquad \text{(B-10)}$$

and we observe that this bound may be replaced by zero if p_ζ is precisely 1 (there would be no $\tilde{\Theta}_\zeta$ samples).

[†]Note that $\gamma_3 = 4$ for this problem.

REFERENCES

1. L. D. Davisson, "Universal Noiseless Coding," IEEE Trans. Information Theory, IT-19, June 1973, pp. 783-795.

2. R. F. Rice, "The Rice Machine," Jet Propulsion Laboratory, California Institute of Technology, JPL Internal Doc. 900-408, Sept. 1, 1970.

3. R. F. Rice and J. R. Plaunt, "Adaptive Variable Length Coding for Efficient Compression of Spacecraft Television Data," IEEE Trans. Commun. Technol., Vol. COM-19, part I, Dec. 1971, pp. 889-897.

4. R. F. Rice, "RM2: transform operations," Technical Memorandum 33-680. Jet Propulsion Laboratory, Pasadena, CA. March 1, 1974.

5. ____________, "An Advanced Imaging Communication System for Planetary Exploration," Vol. 66 SPIE Seminar Proceedings, Aug. 21-22, 1975, pp. 70-89.

6. ____________, "RM2: rms Error Comparisons," Technical Memorandum 33-804. Jet Propulsion Laboratory, Pasadena, CA., Sept. 15, 1976.

7. ____________, "Potential End-to-End Imaging Information Rate Advantages of Various Alternative Communication Systems," JPL Publication 78-52. Jet Propulsion Laboratory, Pasadena, CA., June 15, 1978.

8. ____________, "A Concept for Dynamic Control of RPV Information System Parameters," Proceedings of 1978 Military Electronics Exposition, Anaheim, CA., Nov. 1978.

9. H. Abramson, Information Theory and Coding. New York: McGraw-Hill, 1963.

10. D. Spencer, C. May, "Data Compression for Earth Resource Satellites", Proceedings of the 1972 ITC Conference, October 1972.

11. E. E. Hilbert, "Cluster Compression Algorithm, A Joint Clustering/Data Compression Concept," JPL Publication 77-43. Jet Propulsion Laboratory, Pasadena, CA., Dec. 1, 1977.