Computer Evaluation of Error-correcting Codes for PCM Telemetry

William W. Goddard

Follow this and additional works at: https://openprairie.sdstate.edu/etd

Recommended Citation
https://openprairie.sdstate.edu/etd/3539
COMPUTER EVALUATION OF
ERROR-CORRECTING CODES FOR
PCM TELEMETRY

BY

WILLIAM W. GODDARD

A thesis submitted
in partial fulfillment of the requirements for the
degree Master of Science, Major in
Electrical Engineering, South Dakota
State University

1969

SOUTH DAKOTA STATE UNIVERSITY LIBRARY
COMPUTER EVALUATION OF
ERROR-CORRECTING CODES FOR
PCM TELEMETRY

This thesis is approved as a creditable and independent investigation by a candidate for the degree, Master of Science, and is acceptable as meeting the thesis requirements for this degree, but without implying that the conclusions reached by the candidate are necessarily the conclusions of the major department.

Thesis Advisor

Date

Head, Electrical Engineering
Department

Date
ACKNOWLEDGMENTS

The author wishes to express his appreciation to Dr. A. J. Kurtenbach for his advice throughout this study. Appreciation is also expressed to the author's wife for her assistance during this project.

W.W.G.
### TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>I. INTRODUCTION</strong></td>
<td>1</td>
</tr>
<tr>
<td>A. The PCM System</td>
<td>5</td>
</tr>
<tr>
<td>B. The Simplified PCM System</td>
<td>7</td>
</tr>
<tr>
<td>C. The PCM System with Error Correcting Coding</td>
<td>9</td>
</tr>
<tr>
<td><strong>II. THE PROBLEM</strong></td>
<td>11</td>
</tr>
<tr>
<td>A. The Cost of Coding</td>
<td>11</td>
</tr>
<tr>
<td>B. Criteria for Evaluation</td>
<td>13</td>
</tr>
<tr>
<td>C. Previous Contributions</td>
<td>14</td>
</tr>
<tr>
<td><strong>III. THE COMPUTER IMPLEMENTATION OF THE PROBLEM SOLUTION</strong></td>
<td>19</td>
</tr>
<tr>
<td>A. The Simulation</td>
<td>19</td>
</tr>
<tr>
<td>1. Input Data</td>
<td>19</td>
</tr>
<tr>
<td>2. The Number of Trials</td>
<td>20</td>
</tr>
<tr>
<td>3. The Quantizer Structures</td>
<td>21</td>
</tr>
<tr>
<td>4. The Encoder/Decoder Subroutines</td>
<td>22</td>
</tr>
<tr>
<td>5. Probability of Bit Error in the Noisy Channel</td>
<td>22</td>
</tr>
<tr>
<td>B. The Codes Used</td>
<td>25</td>
</tr>
<tr>
<td>1. Some Important Properties of Group Codes</td>
<td>26</td>
</tr>
<tr>
<td>2. A Decoding Table for Single Bit Errors</td>
<td>27</td>
</tr>
</tbody>
</table>
3. Expanding the Decoding Table for Multiple Bit Errors ................. 28
4. Encoding the (n,k) Sequences ....................................... 32
5. Decoding the Hamming Sequences .................................. 33
6. Decoding the Slepian Sequences ................................... 33
C. The Comparisons Made ............................................... 37

IV. RESULTS .................................................................. 39
A. The (7,4) Hamming Code Performance ............................... 40
B. The Three-Check-Bit Family of Codes ............................... 45
C. The Effect of Including Extra Check Bits ........................... 55
D. The Effect of Correcting Two Different Error Patterns for the (8,5) Codes ........................................ 59

V. CONCLUSIONS .......................................................... 65

LIST OF REFERENCES ....................................................... 67

APPENDIX A .................................................................. 70
APPENDIX B .................................................................. 77
APPENDIX C .................................................................. 79
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Basic PCM system</td>
<td>6</td>
</tr>
<tr>
<td>1.2</td>
<td>Simplified PCM system</td>
<td>8</td>
</tr>
<tr>
<td>1.3</td>
<td>Basic PCM system with error-correcting coding</td>
<td>10</td>
</tr>
<tr>
<td>3.1</td>
<td>Computer flow chart for Slepian decoder subroutine &quot;Deslep&quot;</td>
<td>36</td>
</tr>
<tr>
<td>4.1</td>
<td>$E_2^2$ and $E_c^2$, and $P$ vs relative signal power for (7,4) Hamming code</td>
<td>41</td>
</tr>
<tr>
<td>4.2</td>
<td>$E_2^2$ vs relative signal power</td>
<td>47</td>
</tr>
<tr>
<td>4.3</td>
<td>$E_c^2$ vs relative signal power</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>(Figure 3.6c Wintz and Kurtenbach)</td>
<td></td>
</tr>
<tr>
<td>4.4</td>
<td>Mean squared error, coded transmission</td>
<td>49</td>
</tr>
<tr>
<td>4.5</td>
<td>Channel error for coded and uncoded transmission</td>
<td>51</td>
</tr>
<tr>
<td>4.6</td>
<td>$P$, probability of word error for coded vs uncoded transmission.</td>
<td>53</td>
</tr>
<tr>
<td>4.7</td>
<td>Comparison of (8,5) vs (9,5) code for channel error</td>
<td>56</td>
</tr>
</tbody>
</table>
4.8 Comparison of (8,5) and (9,5) codes, $\epsilon^2$ and $P$ ....... 57

4.9 Comparison of two (8,5) codes--channel error $\epsilon^2$ ....... 60

4.10 Comparison of two (8,5) codes--Mean squared error, $\epsilon^2$, and probability of word error, $P$ .... 61
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Decoding table for single bit errors.</td>
<td>28</td>
</tr>
<tr>
<td>3.2</td>
<td>Partial (5,2) decoding table to correct all single bit, and one double bit error patterns</td>
<td>31</td>
</tr>
<tr>
<td>3.3</td>
<td>Completed (5,2) Slepian decoding table.</td>
<td>32</td>
</tr>
</tbody>
</table>
CHAPTER I

INTRODUCTION

Communication in a broad sense is simply transferring information of some sort from one point to another, usually by means of electrical signals. The sender and the receiver of the information may be either man or machine. If the information to be transferred is in a numerical form, the system is called a Digital Data Communication System. This type of system is usually employed in machine-to-machine communication, but may be applied to transferring other types of information by first converting that information into a numerical (digital) form. In today's typical Digital Data Communications, the numerical data is converted to binary data. This scheme greatly reduces the complexity of the equipment that is needed to transmit and receive, and also makes the receiving problem simpler because there are only two types of signals that are sent. The two signals are made as unlike as possible within the constraints of the system, and thus the receiver is less likely to become confused in deciding which one was sent. The disadvantage of binary signalling is that it takes a group of binary digits (bits) to represent each decimal digit and therefore we may have to send four or more times as many signals in a given time interval to maintain the same data.
rate. This disadvantage is more than offset by the simplicity of the equipment and greater accuracy.

In many machine-to-machine applications of digital communication there is a need for very nearly error free performance. The source of error is noise of some form in the transmitter, channel, or receiver. The classic method of combatting noise is to increase the transmitted power so that signal-to-noise ratio at the receiver is very high and the channel noise is swamped. This method has obvious limitations such as average power, peak power, and governmental regulations. Also the high power is not effective in preventing impulse noise from causing errors in the transmitter, channel, or receiver.

A more sophisticated method of reducing system errors is by use of error-correcting encoding and decoding techniques. These codes map any of \(2^k\) k-bit information sequences (words) into longer n-bit codewords. The extra (n-k) bits are included in each codeword in a systematic manner designed to detect and/or correct bit errors occurring at the receiver. This is illustrated below for the simple case of \(k=2\). The \(2^k=4\) information sequences are \((00), (01), (10), (11)\). It is obvious that a single bit error in any word will result in an erroneous codeword reception. If the four words are systematically encoded by the method of Slepiant, the 4-bit sequences are:
Note that the first two positions of each codeword are the original 2-bit sequences listed above. The third and fourth positions are determined by the following equations.

$$b_3 = b_2$$  \hspace{1cm} (1.1)

$$b_4 = b_1 \oplus b_2$$  \hspace{1cm} (1.2)

where the ring sum $\oplus$ implies ordinary binary addition with no carry, $(1 \oplus 1) = 0$. This is also called "Modulo-two" addition.

It can be seen that each word differs from every other in at least three bit positions. Therefore no single bit error can change one codeword into another. The procedure for using this code is to list the possible sequences that differ from one of the above words by only one position. If any of the listed sequences are received, they are interpreted as illustrated.

<table>
<thead>
<tr>
<th>Original Vocabulary.</th>
<th>0000</th>
<th>0111</th>
<th>1001</th>
<th>1110</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequences which differ by one bit position.</td>
<td>0001</td>
<td>0110</td>
<td>1000</td>
<td>1111</td>
</tr>
<tr>
<td></td>
<td>0010</td>
<td>0101</td>
<td>1011</td>
<td>1100</td>
</tr>
<tr>
<td></td>
<td>0100</td>
<td>0011</td>
<td>1101</td>
<td>1010</td>
</tr>
</tbody>
</table>
Any sequence in column one is interpreted as the column heading 0000 and similarly for the other columns. The lower three sequences in column one illustrate the error patterns which the code will correct; single errors in any of the last three positions. The code will fail for a single error in the first (leftmost) position and for most multiple errors.

The cost of this code is extra complex equipment, and either time or average power/bit. The extra bits also increase the likelihood of single errors, and these costs make systematic codes look less attractive.

It is the purpose of this paper to evaluate the effectiveness of various coding schemes in light of a mean-squared-error criteria. The evaluation is performed by simulating the communication system on a digital computer. The cost of coding is included in the simulation by lowering the average power/bit being transmitted when the longer encoded sequences are sent. This implies that the information is transferred at the same rate, encoded or uncoded.

Chapter I deals with the basic Pulse Code Modulated System, and where the encoding fits in. Chapter II and III discuss the tradeoff involved with encoding and how it is simulated. Results of simulation are included in Chapter IV and the author's conclusion in Chapter V.
A. The PCM System. The Pulse Code Modulated (PCM) system of transmitting a continuous analog waveform, or a set of discrete numbers, is usually implemented in the following three basic steps, which are illustrated in Figure 1.1.

(i) Sample the analog waveform at discrete time points.

(ii) Quantize each of the time samples into any one of a set of discrete quantization values.

(iii) Encode each quantized sample in a natural binary representation.

The binary encoded information is then transmitted bit-by-bit through a noisy communication channel with some modulation scheme such as Phase Shift Keying (PSK) or Frequency Shift Keying (FSK).

The receiver demodulates the received signal and, depending on the type of receiver, may make a decision as to whether each bit is a (1) or (0), or may claim "no decision" if the received bit is too noisy to distinguish. These methods are called binary symmetric channels (BSC), and binary erasure channels (BEC) respectively.

The received sequences of bits are then decoded back into the corresponding quantization values. Different methods are available for reconstructing a continuous waveform from the set of discrete quantization values. Three reconstruction methods are evaluated by
Figure 1.1

Basic PCM system
Wintz and Kurtenbach;\textsuperscript{3} Linear Interpolation, $\frac{\sin \pi x}{x \pi}$, and Zero Order Hold. The linear interpolation is shown to give the smallest mean-squared error.

If the information being sent is in numerical form as mentioned previously for machine-to-machine communication, we would eliminate the time sampler and reconstructor from Figure 1.1. This is done in the next section, but the reason is not to specialize the problem to digital data only, but simply because the sampling and reconstruction is not relevant to the study of the error-correcting codes. The results are equally valid for analog-to-analog, or digital-to-digital transmissions.

B. The Simplified PCM System. An attempt is made here to reduce the system of Figure 1.1 so that it is not necessary to simulate the entire system in order to study the encoding-decoding processes.

First, the input analog wave form and its time sampling unit are replaced by a gaussian random number generator. This simplification is made to conserve memory space in the IBM 360 computer used for the study. The type of modulation, Phase Shift Keying or Frequency Shift Keying, is incorporated in two separate equations of $p$—the probability of bit error in the channel. No reconstruction is done with the received data points because no analog wave was sent. The resulting shortened form of the PCM system is Figure 1.2.
Figure 1.2

Simplified PCM system
C. The PCM System with Error-Correcting Coding. The basic channel is usually encoded by simply numbering the $2^m$ quantization levels in the order $0, 1, \ldots, (2^m-1)$ in $m$-bit natural binary numbers.

Two ways of transferring these natural binary numbers to the user are considered here. The $m$-bit numbers may be directly transmitted bit-by-bit through the noisy channel as in Figure 1.2. The numbers may be modified before transmission by including extra binary bits in a systematic manner designed to detect and correct, at the receiver, some of the bit errors occurring in the noisy channel. It is the purpose of this paper to evaluate the effectiveness of the second method.

The basic PCM system with appropriate switching and error measurement to evaluate systematic codes versus natural codes is illustrated in Figure 1.3. Note that the mean squared error may be made to include quantization error (dotted line) if desired.
Figure 1.3

Basic PCM system with error-correcting coding
CHAPTER II

THE PROBLEM

A. The Cost of Coding. As mentioned before, the advantages of including error-correcting coding are somewhat offset by its cost in terms of complex equipment, time, and/or average power required.

The extra equipment would probably not be a problem at the transmitter; the most difficult part would be in "timing" the information to include the extra check bits in the same time interval. The receiving equipment, however, would have to include some sort of procedure to compare the incoming k-bit sequence with all $2^k$ possible sequences, or at least until a matching one is found. Since this study only simulates the receiver's searching procedure, it gives little information about the difficulty of decoding with real hardware.

Section 3.B, however, examines a convenient sequence that certain Slepian codes follow, which has simplified the decoding subroutine in the computer, and may be applicable to the real hardware.

The time and average power constraints are of more concern here than the equipment's limitations. The available time could become a problem if the uncoded system were inputting data at such a rate as to approach the limit of the speed of the switching circuits in the transmitting and receiving equipment. In that case, the extra
operations necessary for error-correcting coding and decoding may be impossible to do in the same time interval. It is doubtful that this would become a problem since the noisy channel requires that each signal (0 or 1) be sustained for a given amount of time, in order to be recognized at the receiver, and this consideration would probably limit the data rate before switching circuits would.

In any case, this study assumes that the data input information rate is sufficiently less than the system's maximum information rate, as to allow the coding and decoding to be done and the extra bits included in the same time intervals.

This means that our assumed system will operate in real-time, coded or uncoded and that the codewords take an equal time to transmit whether they are k-bit uncoded sequences, or n-bit \((n>k)\) encoded sequences. If the system will operate in the manner we have assumed, the time involved is not a disadvantage of coding, and we turn to the average power as a means of including a cost in our simulation. Here, the average continuous transmitter power is taken to be a constant value (probably a maximum rating for the equipment). If error-correcting coding is used, it must not present any additional demands on the average transmitter power. The trade-off is obvious. We are required to send a codeword of increased length, in a given time
interval, without demanding more transmitted energy, and therefore the transmitted energy for each bit of the new codeword must be less.

Considering the (7,4) code as an illustration, the average energy/codeword might be K. If the uncoded words are 4-bit sequences, the average energy/bit would be K/4. If the words are 7-bit encoded sequences, the average energy/bit is K/7. The use of the error-correcting code has reduced our average energy/bit by 4/7. This reduction in transmitted energy/bit is felt in the noisy channel as lower S/N ratio and a correspondingly higher probability of bit error. Two equations of p—(probability of bit error) are available in the simulation. They represent noncoherent (FSK) or (PSK) signaling. Both equations contain the term \( \beta/m \) which is average word energy to noise ratio (\( \beta \)), divided by the (m), the number of bits per codeword. These equations for p are described in Section 3.A. By simply including the correct value of m in the (p) equations, we insure that \( \beta/m \) (average bit energy/noise) is appropriate to properly weigh the cost of using an error-correcting code.

B. Criteria for Evaluation. Mean-squared-error was chosen as a measure of "goodness" for our system. The lower left portion of Figure 1.3 illustrates the procedure for determining the mean-squared-error from the simulation, and as previously noted the error samples
may be \((x - z)\) or \((y - z)\). The first case would include the quantization error whereas the second case would not. It is pointed out in Clark and Totty\(^4\) that if the quantization is done in an optimum manner (to minimize mean squared quantization error), that the channel and quantization error are additive. Since our simulation uses an optimum quantization technique, we would expect that the effectiveness of the error-code would be reflected in the mean-squared error, regardless of whether we include the quantization error or not.

The mean-squared error is calculated on 1500 codewords being transmitted. A comparison is made by sending the same number of data points in encoded form and again calculating mean-squared error.

This comparison is the basis for deciding whether it is advantageous to use error-correcting coding, and also is a quantitative measure of the advantage or disadvantage.

C. Previous Contributions. Various authors\(^5, 6, 7\) have evaluated the mean squared channel error for numerical data transmitted over a noisy channel. In the work cited above, the data was source encoded in a straightforward manner. No redundancy encoding to combat the adverse effects of the channel was employed. However, two articles\(^4, \ 8\) which report on studies that include channel encoding are available.
Mitryayev uses a "mean risk" function in his paper as a performance criterion for coded channels. Mean risk is similar to our mean-squared error except that it is generalized by using powers of the error other than two.

In most block-coding applications the set of allowable discrete data points is assigned to k-bit natural sequences in order from smallest to largest. Any data point which is received in error will have its own cost of error and this cost will differ throughout the set. Note that when the data points are assigned to their k-bit sequences in natural order, and the k-bit sequences are subsequently encoded into the set of n-bit Slepian codewords, A_0, A_1, A_2,...,A_{n-1}, (n = 2^k), each word A_i (i = 1,2,...,n-1) may have its own probability of word-error \( p_{ij} \). This is given as:

\[
p_{ij} = \sum_{r=1}^{v} p^h(A_j \oplus (S_r \oplus A_i)) n-h(A_j \oplus (S_r \oplus A_i))
\]

where \( v = 2^{(w-k)} \), and \( S_r \) are the words of column one of the Slepian code. The above equation and notation are discussed further in Appendix B if the reader is not familiar with them.

Mitryayev noted that the system performance could be improved if the digital data points could be related to each codeword in such a manner that codewords with higher probability of word error could...
"carry" data points with lower cost of error. His analytical results show a reduction in "mean risk" (mean squared error) by a factor of four at probability of bit error $10^{-3}$ or less.

A study by Clark and Totty\textsuperscript{4} used mean-squared error as a criterion of goodness for error-correcting codes, however, not in a quantitative way as is done in this paper. In their work, Clark and Totty consider their average transmitter power to be the variable quantity, and hold mean-squared error as a constant--just the opposite of what is done here. In varying the average transmitted power, they have sought to make the mean-squared error equal for the coded and uncoded case. They too require that the information be sent in the same interval of time, whether coded or uncoded. The results of their work are interpreted in the following manner: Since they varied the transmitter power to make the values of mean squared error and probability of error coincide for the uncoded and coded cases, they have no measure of the improvement in system performance with the energy per source sample fixed. The quantitative measure of a code's worth is given in terms of the transmitter power that one will gain (or lose) while maintaining the same error performance. Their results are presented in graphical form which clearly illustrates that the probability of bit error for the uncoded channel is the determining factor in deciding whether or not to encode. For the Hamming codes
used, the crossover is in the area of $p = 10^{-3}$, with higher probability of bit error (lower average power) making encoding less desirable. The previously mentioned work of Mitryayev could probably be applied here to improve the mean squared error vs probability of bit error curves by more favorably making the binary word assignments at the quantizer.

A "power gain" of approximately 0.5db is available by decreasing probability of bit error (increasing average power) to the $p = 10^{-6}$ range.

Another interesting point brought out by Clark and Totty is in their comparison of different criteria of goodness. Results are presented in their paper, again graphically, which imply that if we examine the system in the same manner as above except use word error probability instead of mean-squared error, the encoding is still effective at much higher probability of bit error (lower power).

The last result was determined by first computing the probability of word error for the uncoded channel. The same channel is then encoded, and the average power is varied to cause probability of word error to be the same as in the uncoded case. The power required to drive the encoded channel in this case is significantly less and encoding looks much more favorable in light of this criteria.
In this study, the word errors are recorded and an attempt is made to relate the results to those of Clark and Totty.
CHAPTER III

THE COMPUTER IMPLEMENTATION OF THE PROBLEM SOLUTION

A. The Simulation. The block diagram of the PCM system shown in Figure 1.3 is the basis for the simulation program developed for the IBM 360 digital computer.

The approach was to represent each of the elements of Figure 1.3 by a computer subroutine and thus simplify making necessary changes in the type of codes or quantizer schemes being used.

1. Input Data. The input data used was pseudo random Gaussian distributed numbers of zero mean and unit variance for all runs. These numbers were computer generated by an IBM subroutine which was slightly modified to suit this particular application. The Gaussian distribution is a result of the Central Limit Theorem which states that the distribution of the sums of sufficiently large amounts of independent identically distributed random numbers tends to be Gaussian, regardless of the distribution of the numbers being summed. The IBM "Subroutine Gauss" was modified to sum forty-eight random numbers from a uniform distribution rather than summing only twelve. This had the effect of causing the set of data points to assume a more nearly true Gaussian distribution, at the expense of more computer iterations. Appendix C. contains the subroutine "Function Gaussn" along with a distribution of generated data points and shows these
numbers to be very nearly Gaussian distributed. Appendix C. also con- 
tains a uniform random number generator "Function Random" which 
generates a uniform distribution of numbers between 0 and 1. These 
numbers are applied in a noise simulating subroutine in making the 
decision whether or not a bit error should occur. "Function Random" 
is a modification of IBM "Subroutine Randu".10

2. The Number of Trials. The importance of the above random 
distributions is apparent since the validity of the simulation ap- 
proach largely depends on the assumption that the statistical results 
will approach the theoretical values with a sufficiently large num- 
ber of trials. An attempt is made to determine if enough trials 
have been made, by including a sample variance term with each data 
point being received. The sample variance for the jth word Wj is 
simply the variance in squared error computed over all of the words 
which have been received, up to and including Wj. Assume the 
magnitude of error $E_i$ for any received word $W_i$ is defined by 
$E_i = (Z_i - X_i)^2$, where $X_i$ is the data point sent and $Z_i$ is the value 
received.
Sample variance at the jth word $W_j$ is then given as

$$S^2 = \frac{1}{j-1} \sum_{i=1}^{j} (E_i - \bar{E})^2$$

(3.1)

$$\bar{E} = \frac{1}{j} \sum_{i=1}^{j} E_i$$

(3.2)

When the sample variance approaches a constant value and the changes due to word error become small enough we conclude that sufficient trials have been made. A total of 1500 data points were used for all runs in order to insure that the conditions were as nearly the same as possible for making the comparison between coded and uncoded transmission, even though the sample variance sometimes indicated that less would be sufficient.

3. The Quantizer Structures. Uniform quantizer intervals were used for all data runs. The optimum interval which is dependent on the energy/word and the number of bits in the uncoded word, is determined from Wintz and Kurtenbach, 3 Figure 3.4. This interval, while optimum for the uncoded words is not necessarily optimum for the coded words since the probability of word error has changed. This could effect
the comparisons of coded vs uncoded transmission; however, the simula-
tion results of Section 4 show that the optimum coded and uncoded
sequences do correspond closely.

4. The Encoder/Decoder Subroutines. The systematic encoder and
decoder of Figure 1.3 are different subroutines for each (n,k) code
and are all named "codslp" and "deslep" so they may be interchanged
without changing the main program. The basic technique in programming
the simulation is to consider each codeword as an integer decimal
number. All operations are thus written in Fortran IV. computer
language. In cases where it is necessary to determine each individ-
ual bit of a codeword, as in encoding, decoding, and in simulating
individual bit errors, the bits are found by successive division/
multiplication by 2, which in Fortran integer format will change the
value of a number if its rightmost bit is a "1" or will leave the num-
ber unchanged if the rightmost bit is a "0". Similarly, once each
bit of a word is known, any bit may be changed by adding or subtrac-
ing the appropriate power of two in the decimal number.

5. Probability of Bit Error in the Noisy Channel. The noisy
channel is simulated by the "Noise" subroutine.

Various error patterns which occur in actual practice, namely
random, burst, and periodic, have been observed from actual data
transmissions. The burst errors are probably due to impulse noise
or fading which cause many successive bits to be in error. Long code-
words of a Bose-Chaudhuri-Hocquenghem (BCH) type are shown to be
effective in combatting this type of noise; improved performance is
gained by interleaving the codewords, which has the effect of dis-
tributing bursts of bit errors across more than one codeword.\textsuperscript{11} The
random error patterns are the type considered herein. Random error
patterns will result from White Gaussian\textsuperscript{12} noise. Gaussian noise
is the natural result of additive noise from the many noise phenomena
which normally occur in a communications channel, and therefore,
Gaussian noise is used extensively as a noise model for communications
studies.\textsuperscript{1}

An equation for $p$ the probability of channel bit error for the
(BSC), assuming Gaussian noise, is given as\textsuperscript{13}

$$p = \frac{1}{2} \exp \left( - \frac{\xi}{2N_0} \right)$$  \hspace{1cm} (3.3)

where $\frac{\xi}{N_0}$ is the average signal to noise ratio of the received bit
signals. The above equation holds for non-coherent (FSK) reception,
and a similar equation is available for coherent (PSK). Only the
(FSK) system is considered here.
The quantity \( C \) may be written in terms of the following variables:

(i) \( S = \) average transmitter power

(ii) \( T = \) a time interval \([0, T]\)

(iii) \( n = \) the number of words transmitted in time \( T \)

(iv) \( m = \) the number of bits per word.

as 
\[
C = \frac{ST}{nm} \quad \text{(3.4)}
\]

A convenient parameter to use in the computer simulation is defined \( \beta \):

\[
\beta = \frac{ST}{N_0 n} \quad \text{(3.5)}
\]

Substituting 3.5 into 3.3 we have

\[
p = \frac{1}{2} \exp \left( -\frac{\beta}{2m} \right) \quad \text{(3.6)}
\]

As previously mentioned, coded transmission is allowed less transmitter power per bit than in the uncoded case. This decrease in power is included in the simulation by changing \( m \) in equation 3.6, and thus varying \( p \) the probability of bit error. The "Noise" subroutine automatically decides the random bit errors by comparing a random number \( w \) to the probability of bit error \( p \) for each bit of each codeword. The uniformly distributed random numbers lie on
the interval $[0,1]$, and are generated by the "Function Random" mentioned above. If $w > p$, the noise subroutine switches the bit from a zero to a one or vice-versa.

B. The Codes Used. All of the coding schemes which are examined in this paper are included in one class of codes called block codes or group codes. The term "block" is used simply because the information symbols are sent in $n$-bit sequences or $n$-bit "blocks". These sequences are also more commonly called codewords. "Group" is a term applied to these codes because the codewords all satisfy the mathematical definition of a group. Two types of group codes were evaluated, one by R. W. Hamming, and five by D. Slepian. Both codes are systematic $(n,k)$ codes and should give the same results. The difference in the methods is in the decoding scheme. Hamming uses a seemingly more systematic procedure at the decoder. The purpose of the evaluation was to compare the transmission of $k$-bit uncoded sequences with $n$-bit coded sequences. For most cases these $(n,k)$ codes were chosen such that $(n-k) = 3$, that is to say, three check bits were used regardless of the number of information bits $(k)$. Comparisons were made using $(5,2)$, $(6,3)$, $(8,5)$, and $(9,6)$ Slepian codes, and a $(7,4)$ Hamming code. The $(8,5)$ code was further examined by making two separate runs with different multiple error
correction patterns. One run was made using four check bits in a
(9,5) code and the results are compared to an (8,5) code. Graphs
illustrating the relative advantage-disadvantage of all the above
codes are included in Section 4.A.

1. Some Important Properties of Group Codes. In order to ob­
serve some of the properties of group codes, the Slepian (5,2) code
which was shown in Chapter I is used here.

| Codeword 1 | 00000 |
|Codeword 2 | 01110 |
|Codeword 3 | 10101 |
|Codeword 4 | 11011 |

The weight of each codeword is defined as the number of binary "ones"
in the word. Codeword 2 has a weight of three. The Hamming Distance
between two codewords is defined as the number of bit positions in
which the codewords differ. Codewords two and three above have a
Hamming Distance of four. The Minimum Hamming Distance, d, is the
Minimum Hamming distance between any two codewords in the codeword
vocabulary. For the (5,2) code above the Minimum Hamming Distance, d,
is three, and note also that d is equal to the minimum weight of any
codeword, (excluding the zero codeword). This property holds for
all Slepian codes since codeword one always has zero weight.

The error detecting and correcting properties of group codes
are defined in terms of minimum distance (d) as follows:14
(i) If a group code is to detect patterns of \( r \) or fewer bit errors per codeword, it is necessary and sufficient that minimum distance \( d \geq r + 1 \).

(ii) Similarly, for a group code to correct error patterns of \( t \) or fewer bit-errors per codeword, it is necessary and sufficient that the minimum distance \( d \geq 2t + 1 \).

(iii) A group code may be designed to simultaneously detect all patterns of \( r \) or fewer errors, and correct all patterns of \( t \) or fewer errors (\( r \geq t \)) if and only if the minimum distance \( d = r + t + 1 \).

For proofs of the above statements, the reader is referred to Reza.²

Since the communication process we are considering here does not have the ability to re-transmit information if errors are detected at the receiver, (i) and (iii) which deal with detection of errors are of little use. The statement (ii) dealing with correction of bit errors implies that our (5,2) code should correct all patterns of one or fewer errors. The means by which the corrections are made is shown below in the next section.

2. A Decoding Table for Single Bit Errors. In constructing a decoding table for a (5,2) Slepian code, the four codewords of the vocabulary are first written in a row.

\[
\begin{array}{cccc}
00000 & 01110 & 10101 & 11011 \\
\end{array}
\]

A column of 5-bit sequences is then written under each codeword, each of which differs from the column heading in one bit position. This partial decoding table contains only twenty-four of the possible
thirty-two 5-bit sequences. It is capable of correcting single bit errors only.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>01110</td>
<td>10101</td>
<td>11011</td>
</tr>
<tr>
<td>00001</td>
<td>01111</td>
<td>10100</td>
<td>11010</td>
</tr>
<tr>
<td>00010</td>
<td>01100</td>
<td>10111</td>
<td>11001</td>
</tr>
<tr>
<td>00100</td>
<td>01010</td>
<td>10001</td>
<td>11111</td>
</tr>
<tr>
<td>01000</td>
<td>00110</td>
<td>11101</td>
<td>10011</td>
</tr>
<tr>
<td>10000</td>
<td>11110</td>
<td>00101</td>
<td>01011</td>
</tr>
</tbody>
</table>

Table 3.1

(5,2) Decoding table for single bit errors.

If any of the above 5-bit sequences are received, they are interpreted as the codeword of the column heading.

3. Expanding the Decoding Table for Multiple Bit Errors. According to (ii) above, all two bit error patterns cannot be corrected with the (5,2) code. This is not to say that no two-bit errors can be corrected. In fact the (5,2) code can be constructed to correct two of the possible ten double bit error patterns. The decision as to which two of the double error patterns to correct is somewhat arbitrary and is discussed in detail below. First, recall that the leftmost column of Table 3.1 has a "one" in the bit position which will be corrected and that each row of 5-bit sequences below the codewords
corresponds to the same single bit error position. For example the first row below the column headings corresponds to a single bit error in the rightmost position, and every 5-bit sequence in this first row differs from its column heading in the rightmost bit position only. The leftmost column of 5-bit sequences may be thought of as the "generator" of the other columns. By adding word two of column one (00001) to the successive column headings by "Modulo-two" addition, and recalling that for "Modulo-two"
\[
1 \oplus 1 = 0 \\
1 \oplus 0 = 1 \\
0 \oplus 1 = 1 \\
0 \oplus 0 = 0
\]
we get the "Modulo-two" sums, (01111), (10100), (11010) which are the remaining elements of row one. Similarly the lefthand element of every row can be made to generate the elements of that row. It now remains to select some elements for the lefthand column which have weight two and which can similarly generate rows of double bit correctors. The general method for selecting elements for the left-hand column is to arbitrarily select sequences of minimum weight which have not been previously written anywhere in the decoding table. The possible sequences of weight two are:
Of these, only (11000), (10010), (01001) and (00011) have not been used in Table 3.1. Here the decision should be made as to which double error pattern is most costly, and then select the sequence which corrects that pattern. In selecting the double error patterns for all codes used in this paper, the general rule has been to correct the errors corresponding to higher binary numbers. The reason is simply that bit errors in the more significant bits cause larger data errors. The effect of this decision is shown later by comparing two (8,5) coding schemes which have different double error correction patterns. The code which corrects the most significant binary bit positions is shown to be superior in the mean square error sense.

Input data distributions, other than the Gaussian which is considered here, may dictate a different choice of double bit patterns to correct. Selecting (11000) by the above reasoning, and generating the remainder of the row, Table 3.1 becomes:
Table 3.2 contains twenty-eight 5-bit sequences. The remaining four of the possible thirty-two 5-bit sequences are used to correct one more double bit error pattern. Of the two remaining possibilities (10010) and (01001), we select (10010) by the same reasoning as before and finally the (5,2) decoding table is:

<table>
<thead>
<tr>
<th>00000</th>
<th>01110</th>
<th>10101</th>
<th>11011</th>
</tr>
</thead>
<tbody>
<tr>
<td>00001</td>
<td>01111</td>
<td>10100</td>
<td>11010</td>
</tr>
<tr>
<td>00010</td>
<td>01100</td>
<td>10111</td>
<td>11001</td>
</tr>
<tr>
<td>00100</td>
<td>01010</td>
<td>10001</td>
<td>11111</td>
</tr>
<tr>
<td>01000</td>
<td>00110</td>
<td>11101</td>
<td>10011</td>
</tr>
<tr>
<td>10000</td>
<td>11110</td>
<td>00101</td>
<td>01011</td>
</tr>
<tr>
<td>11000</td>
<td>10110</td>
<td>01101</td>
<td>00011</td>
</tr>
</tbody>
</table>

Partial (5,2) decoding table to correct all single bit, and one double bit error patterns.
Table 3.3
Completed (5,2) Slepian decoding table.

4. **Encoding the (n,k) Sequences.** The procedure used for encoding the (n,k) codewords in the subroutines "Codslp" (Slepian) or "Codham" (Hamming) is very similar. For Slepian codes the uncoded binary numbers are mapped into the codewords by use of a matrix\(^2\) like the one below for a (7,4) Slepian code.

\[
\begin{array}{cccc}
5 & 1 & 3 & 4 \\
6 & 1 & 2 & 4 \\
7 & 1 & 2 & 3 \\
\end{array}
\]

This is read as "bit five equals the Modulo-two sum of bits one, three, and four", that is, \(b_5 = b_1 \oplus b_3 \oplus b_4\), and similarly for bits six and seven. The Slepian codes are all written in this form,
with the information bits followed by the check bits. Hamming, on the other hand, chose to intermingle the information and check bits. In the (7,4) Hamming code bits 1, 2, and 4 are check bits rather than 5, 6, and 7 as in the Slepian method. Hamming's method has some advantage in ease of decoding which is shown in the next section.

5. Decoding the Hamming Sequences. The (7,4) Hamming codewords are decoded as follows.

(i) Find each bit \( b_i \) of the received sequence.

(ii) Form the Modulo-two binary sum \( A_1 = b_4 \oplus b_5 \oplus b_6 \oplus b_7 \)

and similarly:

\[
A_2 = b_2 \oplus b_3 \oplus b_6 \oplus b_7 \\
A_3 = b_1 \oplus b_3 \oplus b_5 \oplus b_7
\]

(iii) The binary number \( A_1A_2A_3 \) is the number of the bit position which is in error.

(iv) Change the bit in the position indicated in (iii) and remove the check bits to get the correct binary number.

This procedure is quite simple in comparison to the Slepian technique described below; however, the Slepian codes better lend themselves to correcting multiple bit errors.

6. Decoding the Slepian Sequences. The Slepian codes were decoded by successively comparing the received codeword to each of the column headings in the decoding table. If no codeword matched the
received sequence on the first comparison, the sequence was changed in the first bit position and the comparison made again. The sequence was treated similarly for the remaining bit positions and for the correctable multiple bit error patterns. All multiple bit errors which were beyond the capability of the code to correct were decoded as mistakes. No multiple error detection codes were employed, so the "no decision" capability was not available.

An interesting characteristic of the Slepian codes of three check bits made programming the decoder much simpler. This is discussed briefly below. The matrix which is given for the (6,3) code is

\[
\begin{array}{ccc}
4 & 1 & 2 \\
5 & 1 & 3 \\
6 & 2 & 3 \\
\end{array}
\]

By interchanging the fourth and sixth bits we have

\[
\begin{array}{ccc}
4 & 2 & 3 \\
5 & 1 & 3 \\
6 & 1 & 2 \\
\end{array}
\]

This change has no effect on the capabilities of the code or difficulty of encoding. With the code words constructed in the above manner, the quantization values of 0, 1, 2...7 map into 6-bit code-words which, if interpreted as natural binary numbers, are 0, 11, 21,
30, ... 56. These numbers corresponding to the column headings of the
decoding table can be written $N_0, N_1, N_2, \ldots N_7$. The sequence of num-
bers follows the rule

$$
N_0 = 0 \\
N_1 = N_0 + 11 \\
N_2 = N_1 + 10 \\
N_3 = N_2 + 9 \\
\vdots \\
N_7 = N_6 + 5
$$

A mathematical sequence of this type lends itself very favorably to
computer programming techniques and this was one of the reasons for
using the Slepian code. The decoding procedure is illustrated in the
flow chart Figure 3.1 for single bit errors. The routine as illus-
trated causes the input sequence to be first compared to the Jth
column headings called "KTEST". For the case illustrated, if no
match is found in seven tries, then the Kth bit of NZTEST is changed
and the process repeated. The search is continued until all possible
single bit errors are considered (six in the flow chart shown) and if
no codeword match has occurred, the "decoder fails" message is given.
This "decoder fails" situation cannot occur for a decoding table in
which all of the $(2^N)$ n-bit sequences are employed, but rather, a de-
coding error would occur. In constructing the complete decoder, the
Computer flow chart for Slepian decoder subroutine "Deslep"
above flow chart is simply modified to include extra options in the portion which changes the bits of NZTEST.

C. The Comparisons Made. The major part of the computer time used in this study was for runs which compared the relative quality of coded vs uncoded transmission of digital data. The large number of these runs was due to the fact that five different codes were used, and each one at various power settings. Two other interesting comparisons are also included.

First, the value of extra check-bits in every codeword is examined, by examining the results of a \((9,5)\) encoded transmission of fifteen-hundred data points. Both 9-bit and 8-bit encoded sequences are allowed the same transmitter energy per codeword. Therefore, the 9-bit codewords necessarily have less transmitted energy per bit and the accompanying higher probability of bit error. The trade-off of increased bit error probability for a higher capability code is examined for various initial power settings.

Second, a comparison is made of the results of two different methods of encoding/decoding the \((8,5)\) Slepian sequences. The Slepian \((8,5)\) decoder has the capability of correcting seven of the possible eight single bit error patterns, and none of the multiple bit error patterns. This is seen by examining the dimensions of the \((8,5)\) decoding table. There are \(2^5\) (thirty-two) codewords in the vocabulary
and thus thirty-two columns in the table. There are $2^8$ or (256) possible receivable eight-bit sequences. This implies that there are 256/32, or eight rows in the table. Since one of these rows corresponds to the column headings (codewords), there are only seven rows which correspond to error correction patterns. The same result is verified by examining the (8,5) codewords in light of the general capabilities of codes which are given in section 3.B.1. above. Statement (ii) of that section implies that if the (8,5) code is to correct all single bit error patterns, the minimum Hamming distance must be three or greater. The set of (8,5) codewords, however, have a minimum weight, and thus minimum Hamming distance, of one in the second codeword (00001000). Two (8,5) encoders/decoders were programmed, one of which ignored bit errors in the most significant information bit position; the other ignored bit errors in the least significant bit position. 1500 data points were communicated by each method, and a mean squared error evaluation was made.

Results of all three of the above mentioned comparisons are given in Chapter 4.
CHAPTER IV

RESULTS

The results of the computer simulations which were made to evaluate the various error correcting codes are included in this section. Most of the datum points plotted were obtained by computing the square of the channel error for 1500 data samples and using the average of those computations. The remaining points were obtained using more than 1500 source samples. In some cases, word error probability is also given as a criteria. Three types of comparisons are made in this section.

First, comparisons are made of uncoded vs coded transmission of data points. Two other cases are considered: one compares two different codes to send the same set of data points \((9,5)\) vs \((8,5)\) encoding, the other compares two different variations of the \((8,5)\) code. The graphs presented in this chapter are plotted on a minimum number of points in order to conserve computer time. They are, however, intended to illustrate the trends of the curves rather than the precise paths. In that respect the number of points and the linear interpolation between them is considered quite adequate. In all cases the uncoded system is represented by a dashed line and the coded system by a solid line.
A. The (7,4) Hamming Code Performance. The results of the seven runs made on the (7,4) Hamming code are shown in Figure 4.1. As mentioned in the previous chapter, the (7,4) Hamming code corrects all patterns of single bit errors exactly the same as the (7,4) Slepian code. The difference between these codes is simply the method of coding/decoding, so we may therefore interpret the results given in Figure 4.1 as a (7,4) Hamming or a (7,4) Slepian code. The symbols used in Figure 4.1 and in the following graphs are listed here for the reader's convenience.

\[ \beta \] - The ratio of average codeword energy to noise power spectral density.

\[ \varepsilon_2^* \] - The squared error of quantizer and channel averaged over at least 1500 data points transmitted.

\[ \varepsilon_2^c \] - The squared error of the communications channel averaged over at least 1500 data points.

\[ P \] - The probability of word error for codeword transmission.

This is the empirical value determined from the number of word errors which occur in 1500 data point transmissions.

The mean squared term \( \varepsilon_2^* \) is an estimate of the expected value (E) in terms of the notation of Figure 1.3

\( \varepsilon_2^* \triangleq E (x-z)^2 \)  \hspace{1cm} (4.1)
\[ \epsilon^2, \text{ Mean squared total error and } \epsilon_c^2, \text{ Mean squared channel error} \]

Figure 4.1: Average (energy/codeword)/noise spectral density for (7,4) Hamming code: \( \epsilon^2 \) and \( \epsilon_c^2 \) vs relative signal power, \( P \). Probability of Word Error.
This equation is expanded to explicitly include the \( y \) quantization values.

\[
\varepsilon^2_{\ast} = E \left[ (x-y) + (y-z)^2 \right] \tag{4.2}
\]

\[
= E (x-y)^2 + 2E (x-y)(y-z) + E(y-z)^2 \tag{4.3}
\]

The three terms on the right-hand side of equation 4.3 are written in order in the following notation.

\[
\varepsilon^2_{\ast} = \varepsilon^2_q + \varepsilon^2_m + \varepsilon^2_c \tag{4.4}
\]

Note that the mutual error term \( \varepsilon_m \) is not squared since it may be positive or negative and therefore complicates the explanation of the graphical results. The quantization error \( \varepsilon_q \), and the mutual error \( \varepsilon_m \) were not given in the computer output listing because \( \varepsilon^2_{\ast} \) and \( \varepsilon^2_c \) are probably of more immediate concern to the user.

A total of 1500 trials was used for most data runs except in the regions where the power was high enough to severely limit the number of bit errors. In this region (typically \( \beta = 80 \)), 3000 or 4500 trials were made to obtain a more accurate estimate of the average.

The mean squared error curves of Figure (7,4) show the coded vs uncoded comparison. We see that the uncoded sequences are telemetered
with less error in the low power regions and more error in the high power regions. This is shown later to be typical of most codes investigated. The crossover point is approximately at $\beta$ equals forty-five for the mean squared error terms. The channel error curves are seen to tend to zero as fewer and fewer word errors are made in the high power region. The $E^2$ term, which contains both quantizer and channel errors approaches a constant value related to average quantizer error in the higher power regions. This is apparent because of the fact that the mutual term $E_{m}^2$ also tends to zero along with the channel error term $E_{c}^2$ at higher power. (See equation 4.3 and 4.4).

Probability of word error ($P$) is seen to cross at a lower power setting than did the mean squared error curves. This implies that, in the region of the $P$ crossover ($\beta = 42$), the coded system can make fewer word errors, and yet have higher mean squared errors $E^2$ and $E_{c}^2$. In order to account for this behavior we must examine the types of word errors involved in both the coded and uncoded cases. Since the (7,4) code corrects all single bit errors, all codeword errors must be the result of two or more bit errors per word. On the other hand, the uncoded words are subject to single bit error patterns; in fact the results show that only single bit errors affect the uncoded words in the region being considered. (See Table A-6). That is, for
<table>
<thead>
<tr>
<th>Bit position in error.</th>
<th>Probability of error in that position, given single error has occurred.</th>
<th>Number of quantization level changes due to error.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1/4</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>1/4</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>1/4</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1/4</td>
<td>1</td>
</tr>
</tbody>
</table>

On the average we can conclude that a single word error will cause the channel error to be $G$ quantization levels where:

$$G = \frac{1}{4}(8) + \frac{1}{4}(4) + \frac{1}{4}(2) + \frac{1}{4}(1) \quad (4.5)$$

$$= 3 \frac{3}{4}$$

A similar result could be obtained by analyzing the $(7,4)$ codewords for all twenty-one double bit error patterns, and also all the higher number error patterns. The above reasoning helps to explain the work of Clark and Totty\(^4\) which also shows the $(7,4)$ coded transmission to look more favorable in terms of word error probability than when mean squared error is used as a criteria.

B. The Three-Check-Bit Family of Codes. The graphs presented in this section show the coding comparisons for the entire family of three check bit codes tested. As previously mentioned, the subroutines
were made easier to write for different codes by keeping the number of check bits constant. The codes used in this section are (5,2), (6,3), (7,4), (8,5) and (9,6). The mean squared total error \((x-z)^2\), \(\overline{E_x^2}\) is given in Figure 4.2. The curves shown tend to approach a constant value as the signal power (and correspondingly \(\beta\)) are increased sufficiently. As shown in previous discussion, this value is approaching the average quantization error squared \(E_q^2\). The curves of Figure 4.2 could be used in selecting an optimum word length for any given relative signal strength \(\beta\). At lower transmitter power, or higher noise levels, (corresponding to lower \(\beta\)) the systems with fewer information bits are superior in terms of \(E_x^2\). As \(\beta\) is increased, the systems with higher numbers of information bits have less mean squared error as expected. A set of analytically determined curves given by Wintz and Kurtenbach\(^3\), is used to find the optimum word length for uncoded sequences. These curves, which are reproduced here in Figure 4.3, provide a good cross-reference for the accuracy of our simulation results. Due to the difference in scales, Figure 4.2 and 4.3 are not readily compared by inspection; however, point-by-point comparison shows the simulation values to compare favorably to the theoretical values.

Figure 4.4 also provides a means for selecting an optimum code-word length for any given transmitter signal to noise ratio. The
Figure 4.2 $\varepsilon^2_*$ vs relative signal power
Figure 4.3 $\epsilon^2$ vs relative signal power
(Fig. 3.6c Wintz and Kurtenbach)
Figure 4.4 $\epsilon^2_*$ Mean squared error, coded transmission

\( \beta \): Average (energy/word)/noise spectral density
curves of $\varepsilon^2_*$ for the three-check-bit coded sequences again show the tendency to approach the constant value $\varepsilon^2_q$. The tables of data points included in Appendix "A" are a good cross-reference for the uncoded and coded $\varepsilon^2_*$ curves of Figures 4.2 and 4.4. Comparison shows the optimum uncoded sequence for any given $\beta$. For example, at $\beta = 50$, the optimum uncoded sequence is (4,4) while the optimum coder is the (7,4).

In general, the $\varepsilon^2_*$ coded curves have higher values than the uncoded curves in the regions of low $\beta$. As $\beta$ is increased, the coded $\varepsilon^2_*$ curves approach and cross the uncoded curves, thereby being effective enough to perhaps justify their use. The crossover points are approximately at Beta equals forty-two for the (6,3), (7,4) and (8,5) codes, and sixty-two for the (9,6) code. No crossover takes place for the (5,2) code. The poorer performance of the (9,6) code is probably due to its increased number of bits without an increased error-correcting capability. The (9,6) code can only correct seven single-bit error patterns, the same as the (8,5) code. The advantage of the (9,6) code (smaller quantizer intervals) is apparently more than offset by its disadvantage in transmitted energy per bit.

Channel mean squared error $\varepsilon^2_C$, which is presented in Figure 4.5 for the entire family of codes, shows the crossover points similar to
Figure 4.5  Channel error for coded and uncoded transmission
the $\bar{E}_{c}^2$ curves. The $\bar{E}_{c}^2$ terms tend to zero with increasing power, while the $\bar{E}_{*}^2$ values tend to approach a constant value $\bar{E}_{q}^2$ mean squared quantization error.

Word error probability ($P$) is presented in Figure 4.6, again for the entire three-check-bit family of coders tested. The quantity $P$ is an experimental value which is determined after each run by simply dividing the total number of word errors by the number of words sent. In examining the set of curves, it is noted that the crossover does not occur for three of the five cases. This is further illustration of the point mentioned in the discussion of the $(7,4)$ code: that the $(P)$ curves may exhibit different behavior than the $\bar{E}_{*}^2$ curves, even though the $\bar{E}_{c}^2$ is a result of word errors. This is very clearly shown by the $(9,6)$ and $(6,6)$ $P$ curves. The coded case $(9,6)$ is not able to improve the probability of word error to better the uncoded $(6,6)$ case; however, referring back to Figures 4.4 and 4.5, at $\beta = 80$ the $(9,6)$ code is far superior to the uncoded $(6,6)$ transmission both in terms of $\bar{E}_{*}^2$ and $\bar{E}_{c}^2$. A similar comparison could be made for the $(8,5)$ code. As mentioned previously, in the discussion of the $(7,4)$ coder/decoder, the average probability of word error and the average squared error are not expected to show similar results. This is because the $\bar{E}_{c}^2$ and $\bar{E}_{*}^2$ criterion take into account the magnitude of each word error, while the average probability of word error $P$ considers all word errors to be weighted the same.
Figure 4.6 $P$, probability of word error for coded vs uncoded transmission.
The set of probability-of-word error curves is thought to be a good representation of a code's efficiency. For example, the (7,4) code is able to improve on $P$ over a much greater region of the graph than is the (6,3) code, and the (5,2), (8,5), and (9,6) curves do not cross at all. The (7,4) code is the only "perfect" code; that is, the only code studied here that corrects any/all single bit error patterns. Other codes, (6,3) and (5,2), are capable of some multiple bit correction; however, they achieve this at a higher cost in terms of transmitted energy/bit. Since the single bit error patterns are most prevalent, these codes have an "over capability" which was purchased at too high a cost and thus are less efficient than the (7,4) in combatting total word errors. On the other hand, the (8,5) and (9,6) codes could be said to have an "under capability" in terms of correcting bit errors. These codes are not capable of correcting all the single bit errors that may occur. Yet their use demands that less transmitter power be allotted to each bit sent. They are understandably less efficient than the (7,4) code for word error.

Finally, it is pointed out that the graphs of Figures 4.5 and 4.6 could be useful in properly matching the data source to the type of transmission scheme. For instance, assume that it is necessary to send data points at a given rate, in sequences with six information bits and under signal to noise conditions of $\beta = 80$. 
Referring to Figures 4.5 and 4.6, we attempt to make the decision whether to use 6-bit uncoded words or a (9,6) coder. The decision is obviously a function of the error criteria considered. If the data is to be samples of an analog waveform, for example, the least mean squared channel error $\left( \sum_{c}^{2} E \right)$ might be the criteria which would give the best reproduced waveform. In this case Figure 4.5 calls for a (9,6) encoder. On the other hand, if the data were strictly numerical, as with financial data, the (P) probability of word error would likely be a more appropriate criteria. Now Figure 4.6 shows a big advantage of not using the (9,6) encoder, but rather simply transmitting the 6-bit words.

C. The Effect of Including Extra Check Bits. At this point a deviation is made from the three-check-bit type coder/decoders. An attempt is made to determine the effect of including extra check bit positions in the code. From the discussion in the previous section about coding efficiency, one might expect the extra check bits to give little or no improvement. This is shown to be the case in this section; in fact, the (9,5) encoding was shown to be inferior to (8,5) encoding. Referring to Figures 4.7 and 4.8, the only advantage ever gained by the (9,5) code is in word error probability at $\beta = 80$. The log scale for (P) shows this to be a sizable difference; however, reference to Table A-11 shows the actual average to be four words in
Figure 4.7 Comparison of (8,5) vs (9,5) code for channel error
Figure 4.8 Comparison of (8, 5) and (9, 5) codes, $\epsilon$ and P
The mean squared error curves indicate no advantage for the (9,5) vs (8,5) code and, for the most part, the (5,5) uncoded transmission is better. The (8,5) code used is designed to correct single bit errors in the first seven positions. The (9,5) code corrects all nine single bit error patterns, along with six of the double bit error patterns. Double error patterns to be corrected are selected as those with highest cost of error; that is, double bit patterns lying in the most significant information bits. Ordinarily, one would assume that a change from (8,5) to (9,5) encoding would be advantageous since it is possible to gain all this error-correcting capability at the expense of only one extra bit. The simulation certainly shows otherwise. The extra bit in each (9,5) codeword causes the average transmitted energy per bit to decrease by a factor of 8/9. This is apparently enough to induce sufficiently more double bit error patterns to occur and to offset the advantages of the increased error-correcting capability. No further codes were examined by including extra check bits as above. It is difficult to project from these results what to expect from other similar comparisons. One would suspect, however, that a change in the direction towards a "perfect" code, or more efficient code, (for example, changing a (6,4) to a (7,4) code) would net an advantage.
D. The Effect of Correcting Two Different Error Patterns for the (8,5) Codes. In constructing the (8,5) coder/decoder for the computer simulation, it was noted that many options were available as to which single bit error positions to correct. The (8,5) code has a vocabulary of \(2^5 = 32\) codewords. Since the decoding table must contain all of the possible \(2^8 = 256\) words, we conclude that the columns must be \(256/32 = 8\) words long. Subtracting the column heading (codeword) from this number, there remain seven elements in each column which correspond to correction of single bit error patterns. The remaining bit position must be ignored and a bit error in that position causes a word error.

The first case considered, called \((8,5)_1\), is an error corrector which ignores bit errors in the rightmost (least significant) information position. The other extreme is an (8,5) code, \((8,5)_2\), which ignores bit errors in the leftmost (most significant) information bit position. The results which appear in Figures 4.9 and 4.10 show the decided advantage of the \((8,5)_1\) scheme. The \((8,5)_2\) code is unable to even compare to the uncoded case at any point on the curves. An explanation of the behavior of the two (8,5) coder/decoders follows.

The following listing of binary numbers is the set of 3-bit information sequences. (Three bits are used now for simplicity and the argument will be extended to the 5-bit sequences.)
$\epsilon^2_{(8,5)_2}$

$\epsilon^2_{(5,5)}$

$\epsilon^2_{(8,5)_1}$

Figure 4.9 Comparison of two (8,5) codes--channel error $\epsilon^2_c$
Figure 4.10 Comparison of two (8,5) codes—Mean squared error, $E^2_*$, and probability of word error, $P_*$
The lines on the right indicate the word changes which would occur if we were to ignore bit errors in the rightmost bit position. This value is always one quantization level. Likewise, the lines on the left show that bit errors in the most significant position cause the value of a word to change by four quantization levels. Note that this is independent of the data distribution. Extended to the (8,5) code, bit errors in the most significant information position cause the word to change by sixteen quantization levels. The mean squared channel error terms may be written for the (8,5) \(_1\) and (8,5) \(_2\) codes as:

\[
\varepsilon_c^2 = \frac{\sum_m (y-z)^2 + \sum_s (y-z)^2}{1500} \quad (4.6)
\]

Where \(\sum_m (y-z)^2\) is the summation of all word error terms caused by multiple bit errors, and similarly \(\sum_s (y-z)^2\) for single bit errors. Assuming that the number of multiple bit errors remains the same for both (8,5) codes and similarly for the total number of word errors,
equations for the \((8,5)_1\) and \((8,5)_2\) mean squared channel error are respectively:

\[
\mathcal{E}_{c1}^2 = \frac{\sum_m (y-z)^2}{1500} + \frac{N_1(r^2)}{1500} \\
\mathcal{E}_{c2}^2 = \frac{\sum_m (y-z)^2}{1500} + \frac{N_1(16r)^2}{1500}
\]

(4.7)

(4.8)

with \(N_1\) being the number of word errors due to single bit errors and \(r\) the quantizing interval. Subtracting, we have the difference between the two mean squared error curves.

\[
(\mathcal{E}_{c2}^2 - \mathcal{E}_{c1}^2) = \mathcal{E}_{c3}^2 = \frac{(16^2-1)N_1r^2}{1500}
\]

(4.9)

Since the quantizing interval \((r)\) is \((1.72 \times 10^{-1})\) for \((8,5)\) codes, one would expect that the \((8,5)_2\) code would have a mean squared error which is higher than in the \((8,5)_1\) case by the amount:

\[
\mathcal{E}_{c}^2 = \frac{N_1(16^2-1)(1.72 \times 10^{-1})^2}{1500}
\]

(4.10)

\[
= N_1(5.029 \times 10^{-3})
\]
This equation compares favorably with the actual simulation results. The same reasoning could be applied to the remaining (8,5) codes which would correct other single bit patterns. It would be expected that the mean squared error performance would be ordered, with the best performance coming from the codes which correct the most significant information bit, regardless of the data distribution.
CHAPTER V

CONCLUSIONS

Error-correcting binary coding methods were evaluated using a computer model of a pulse code modulation (PCM) communications system. The basic block diagram of the system was given in Chapter I, and the computer program was written to simulate the system. Mean-squared error was chosen as the performance criteria. Constraints on the system were constant average power, and constant input data rate. Comparisons made of mean-squared error for coded vs uncoded channels showed similar results throughout five codes tested. In general, the uncoded sequences are a better system than the coded ones at low signal energy-to-noise ratios. As transmitter power is raised (or noise lowered), the longer coded sequences improve and out perform the uncoded system. In the case of the (7,4) code with (average energy/bit)/noise power (or $\beta$) = 60, the coded system gave forty-four percent less mean squared error $\varepsilon^2$. Another criterion of performance which was available from the computer simulation--(P) probability of word error--was compared to the mean-squared error criterion. The encoding looked more favorable in view of probability of word error. The same result was shown by Clark and Totty. This is largely due to the type of word errors which occur
in coded and uncoded transmission. In the regions of comparison, word errors in the uncoded transmission are mostly single-bit patterns; whereas, the word errors in coded transmission are generally more costly multiple bit errors.

The effect of including an extra check bit in the encoded transmission (9,5) vs (8,5) was shown to be of no value. The (9,5) code has less transmitted energy per bit than the (8,5) code and as a result makes more bit errors. The increased error-correction capability which results from the extra check bit is insufficient to offset the extra bit errors.

Two methods of constructing the (8,5) coder/decoder were evaluated. It was determined that the best (8,5) decoder corrects bit errors in most significant information bit positions. The same result is generalized to apply to any Slepian code that has an option of which bit error patterns to correct. The set of mean-squared error curves $E^2$ for the uncoded system compare favorably to the theoretically computed values as given by Wintz and Kurtenbach$^3$. This is a good verification of the simulation results.
LIST OF REFERENCES


3. P.A. Wintz and A.J. Kurtenbach, Analysis and Minimization of Message Error in PCM Telemetry. Lafayette, Indiana: Purdue University, TR-EE67-19, 1967, p. 90, Figure 3.6 (C) p. 47.


Table A-1

Slepian (2,2) Uncoded Transmission

<table>
<thead>
<tr>
<th>( \beta )</th>
<th># Bit errs. (predicted)</th>
<th># Bit errs. (actual)</th>
<th>( \epsilon^2 )</th>
<th>( \epsilon_c^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>123</td>
<td>120</td>
<td>7.74x10^{-2}</td>
<td>2.85x10^{-1}</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>14</td>
<td>9.17x10^{-3}</td>
<td>1.39x10^{-1}</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td>2</td>
<td>1.33x10^{-3}</td>
<td>1.24x10^{-1}</td>
</tr>
</tbody>
</table>

Table A-2

Slepian (5,2) Coded Transmission

<table>
<thead>
<tr>
<th>( \beta )</th>
<th># Bit errs. (predicted)</th>
<th># Bit errs. (actual)</th>
<th>( \epsilon^2 )</th>
<th>( \epsilon_c^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1397</td>
<td>1422</td>
<td>1.95x10^{-1}</td>
<td>5.74x10^{-1}</td>
</tr>
<tr>
<td>20</td>
<td>507</td>
<td>530</td>
<td>3.13x10^{-2}</td>
<td>2.18x10^{-1}</td>
</tr>
<tr>
<td>30</td>
<td>186</td>
<td>195</td>
<td>2.0 x10^{-3}</td>
<td>1.24x10^{-1}</td>
</tr>
</tbody>
</table>
### Table A-3

**Slepian (3,3) Uncoded Transmission**

<table>
<thead>
<tr>
<th>$\beta$</th>
<th># Bit errs. (predicted)</th>
<th># Bit errs. (actual)</th>
<th>P-probability of Word Error $\left(\frac{(x-z)^2}{(y-z)^2}\right)$</th>
<th>$\varepsilon_*^2$</th>
<th>$\varepsilon_c^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>424</td>
<td>424</td>
<td>2.56x10^{-1}</td>
<td>3.86x10^{-1}</td>
<td>1.93x10^{-1}</td>
</tr>
<tr>
<td>20</td>
<td>80</td>
<td>76</td>
<td>4.92x10^{-2}</td>
<td>1.27x10^{-1}</td>
<td>5.87x10^{-2}</td>
</tr>
<tr>
<td>30</td>
<td>15</td>
<td>18</td>
<td>1.20x10^{-2}</td>
<td>7.06x10^{-2}</td>
<td>2.157x10^{-2}</td>
</tr>
<tr>
<td>40</td>
<td>2</td>
<td>5</td>
<td>3.33x10^{-3}</td>
<td>4.53x10^{-2}</td>
<td>6.51x10^{-3}</td>
</tr>
<tr>
<td>45</td>
<td>1</td>
<td>2</td>
<td>1.33x10^{-3}</td>
<td>4.35x10^{-2}</td>
<td>4.48x10^{-3}</td>
</tr>
<tr>
<td>50</td>
<td>0</td>
<td>1</td>
<td>6.60x10^{-4}</td>
<td>3.96x10^{-2}</td>
<td>8.97x10^{-4}</td>
</tr>
<tr>
<td>60</td>
<td>0</td>
<td>0</td>
<td></td>
<td>3.89x10^{-2}</td>
<td>0</td>
</tr>
</tbody>
</table>

### Table A-4

**Slepian (6,3) Coded Transmission**

<table>
<thead>
<tr>
<th>$\beta$</th>
<th># Bit errs. (predicted)</th>
<th># Bit errs. (actual)</th>
<th>P-probability of Word Error $\left(\frac{(x-z)^2}{(y-z)^2}\right)$</th>
<th>$\varepsilon_*^2$</th>
<th>$\varepsilon_c^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1955</td>
<td>2016</td>
<td>3.83x10^{-1}</td>
<td>7.06x10^{-1}</td>
<td>4.56x10^{-1}</td>
</tr>
<tr>
<td>20</td>
<td>849</td>
<td>876</td>
<td>1.00x10^{-1}</td>
<td>2.63x10^{-1}</td>
<td>2.00x10^{-1}</td>
</tr>
<tr>
<td>30</td>
<td>369</td>
<td>378</td>
<td>2.26x10^{-2}</td>
<td>9.16x10^{-2}</td>
<td>5.09x10^{-2}</td>
</tr>
<tr>
<td>40</td>
<td>160</td>
<td>157</td>
<td>5.32x10^{-3}</td>
<td>5.18x10^{-2}</td>
<td>1.14x10^{-2}</td>
</tr>
<tr>
<td>45</td>
<td>105</td>
<td>111</td>
<td>1.33x10^{-3}</td>
<td>4.05x10^{-2}</td>
<td>1.12x10^{-3}</td>
</tr>
<tr>
<td>50</td>
<td>69</td>
<td>67</td>
<td>0</td>
<td>3.89x10^{-2}</td>
<td>0</td>
</tr>
<tr>
<td>60</td>
<td>30</td>
<td>37</td>
<td>0</td>
<td>3.89x10^{-2}</td>
<td>0</td>
</tr>
<tr>
<td>Table A-5</td>
<td>Hamming (4,4) Uncoded Transmission</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>β</strong></td>
<td><strong># Bit errs. (predicted)</strong></td>
<td><strong># Bit errs. (actual)</strong></td>
<td><strong>P-probability of Word Error.</strong></td>
<td>$\epsilon^2_*$</td>
<td>$\epsilon^2_c$</td>
</tr>
<tr>
<td>20</td>
<td>246</td>
<td>252</td>
<td>1.59x10^{-2}</td>
<td>1.96x10^{-1}</td>
<td>1.80x10^{-1}</td>
</tr>
<tr>
<td>30</td>
<td>70</td>
<td>79</td>
<td>5.26x10^{-2}</td>
<td>8.78x10^{-2}</td>
<td>3.55x10^{-2}</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>28</td>
<td>1.86x10^{-2}</td>
<td>4.69x10^{-2}</td>
<td>2.98x10^{-2}</td>
</tr>
<tr>
<td>44</td>
<td>12</td>
<td>18</td>
<td>1.20x10^{-2}</td>
<td>3.94x10^{-2}</td>
<td>2.49x10^{-2}</td>
</tr>
<tr>
<td>50</td>
<td>5</td>
<td>12</td>
<td>8.00x10^{-3}</td>
<td>3.17x10^{-2}</td>
<td>1.74x10^{-2}</td>
</tr>
<tr>
<td>60</td>
<td>1</td>
<td>4</td>
<td>2.66x10^{-3}</td>
<td>2.33x10^{-2}</td>
<td>1.01x10^{-2}</td>
</tr>
<tr>
<td>80</td>
<td>0</td>
<td>1</td>
<td>6.67x10^{-4}</td>
<td>1.33x10^{-2}</td>
<td>2.90x10^{-4}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table A-6</th>
<th>Hamming (7,4) Coded Transmission</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>β</strong></td>
<td><strong># Bit errs. (predicted)</strong></td>
</tr>
<tr>
<td>20</td>
<td>1258</td>
</tr>
<tr>
<td>30</td>
<td>615</td>
</tr>
<tr>
<td>40</td>
<td>301</td>
</tr>
<tr>
<td>44</td>
<td>226</td>
</tr>
<tr>
<td>50</td>
<td>147</td>
</tr>
<tr>
<td>60</td>
<td>72</td>
</tr>
<tr>
<td>80</td>
<td>17</td>
</tr>
<tr>
<td>( \beta )</td>
<td># Bit errs. (predicted)</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>20</td>
<td>507</td>
</tr>
<tr>
<td>40</td>
<td>68</td>
</tr>
<tr>
<td>60</td>
<td>9</td>
</tr>
<tr>
<td>80</td>
<td>1</td>
</tr>
</tbody>
</table>

Table A-8

Slepian (8,5) Coded Transmission

<table>
<thead>
<tr>
<th>( \beta )</th>
<th># Bit errs. (predicted)</th>
<th># Bit errs. (actual)</th>
<th>P-probability of Word Error.</th>
<th>( \varepsilon_*^2 )</th>
<th>( \varepsilon_c^2 )</th>
<th>( (x-z)^2 )</th>
<th>( (y-z)^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>1719</td>
<td>1763</td>
<td>( 3.87 \times 10^{-1} )</td>
<td>4.31 \times 10^{-1}</td>
<td>3.12 \times 10^{-1}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>492</td>
<td>502</td>
<td>( 6.60 \times 10^{-2} )</td>
<td>8.80 \times 10^{-2}</td>
<td>6.76 \times 10^{-2}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>141</td>
<td>145</td>
<td>( 1.00 \times 10^{-2} )</td>
<td>1.69 \times 10^{-2}</td>
<td>1.05 \times 10^{-2}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>40</td>
<td>42</td>
<td>( 3.32 \times 10^{-3} )</td>
<td>5.30 \times 10^{-3}</td>
<td>1.50 \times 10^{-4}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table A-9
Slepian (6,6) Uncoded Transmission

<table>
<thead>
<tr>
<th>( \beta )</th>
<th># Bit errs. (predicted)</th>
<th># Bit errs. (actual)</th>
<th>P-probability of Word Error</th>
<th>( \varepsilon_*^2 )</th>
<th>( \varepsilon_c^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>849</td>
<td>876</td>
<td>4.57x10^{-1}</td>
<td>3.19x10^{-1}</td>
<td>1.68x10^{-1}</td>
</tr>
<tr>
<td>40</td>
<td>160</td>
<td>157</td>
<td>9.85x10^{-2}</td>
<td>8.61x10^{-2}</td>
<td>5.60x10^{-2}</td>
</tr>
<tr>
<td>60</td>
<td>30</td>
<td>37</td>
<td>2.46x10^{-2}</td>
<td>2.35x10^{-2}</td>
<td>1.55x10^{-2}</td>
</tr>
<tr>
<td>70</td>
<td>13</td>
<td>16</td>
<td>1.06x10^{-2}</td>
<td>1.50x10^{-2}</td>
<td>9.82x10^{-3}</td>
</tr>
<tr>
<td>80</td>
<td>5</td>
<td>9</td>
<td>6.00x10^{-3}</td>
<td>8.66x10^{-3}</td>
<td>5.35x10^{-3}</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>2</td>
<td>1.32x10^{-3}</td>
<td>2.31x10^{-3}</td>
<td>4.09x10^{-4}</td>
</tr>
</tbody>
</table>

### Table A-10
Slepian (9,6) Coded Transmission

<table>
<thead>
<tr>
<th>( \beta )</th>
<th># Bit errs. (predicted)</th>
<th># Bit errs. (actual)</th>
<th>P-probability of Word Error</th>
<th>( \varepsilon_*^2 )</th>
<th>( \varepsilon_c^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>2222</td>
<td>2272</td>
<td>5.48x10^{-1}</td>
<td>4.86x10^{-1}</td>
<td>3.06x10^{-1}</td>
</tr>
<tr>
<td>40</td>
<td>731</td>
<td>758</td>
<td>1.72x10^{-1}</td>
<td>1.24x10^{-1}</td>
<td>9.22x10^{-2}</td>
</tr>
<tr>
<td>60</td>
<td>240</td>
<td>241</td>
<td>4.2x10^{-2}</td>
<td>3.04x10^{-2}</td>
<td>2.25x10^{-2}</td>
</tr>
<tr>
<td>70</td>
<td>138</td>
<td>140</td>
<td>2.53x10^{-2}</td>
<td>1.22x10^{-2}</td>
<td>7.16x10^{-3}</td>
</tr>
<tr>
<td>80</td>
<td>79</td>
<td>84</td>
<td>1.40x10^{-2}</td>
<td>2.67x10^{-3}</td>
<td>2.32x10^{-4}</td>
</tr>
<tr>
<td>100</td>
<td>26</td>
<td>31</td>
<td>4.00x10^{-3}</td>
<td>2.04x10^{-3}</td>
<td>1.08x10^{-4}</td>
</tr>
</tbody>
</table>
### Table A-11

Comparison of (8,5) and (9,5) coders

<table>
<thead>
<tr>
<th>$\beta$</th>
<th># Bit errs. (predicted)</th>
<th># Bit errs. (actual)</th>
<th>P-probability of Word Error.</th>
<th>$\epsilon^2_\text{a}$</th>
<th>$\epsilon^2_c$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>$\left(\frac{x-z}{(x-z)}\right)^2$</td>
<td>$\left(\frac{y-z}{(y-z)}\right)^2$</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>507</td>
<td>530</td>
<td>$3.10 \times 10^{-1}$</td>
<td>$2.89 \times 10^{-1}$</td>
<td>$1.73 \times 10^{-1}$</td>
</tr>
<tr>
<td>40</td>
<td>68</td>
<td>64</td>
<td>$4.92 \times 10^{-2}$</td>
<td>$8.30 \times 10^{-2}$</td>
<td>$5.92 \times 10^{-2}$</td>
</tr>
<tr>
<td>60</td>
<td>9</td>
<td>15</td>
<td>$1.00 \times 10^{-2}$</td>
<td>$2.70 \times 10^{-2}$</td>
<td>$2.07 \times 10^{-2}$</td>
</tr>
<tr>
<td>80</td>
<td>1</td>
<td>$\frac{1}{4}$</td>
<td>$8.32 \times 10^{-3}$</td>
<td>$7.90 \times 10^{-3}$</td>
<td>$4.13 \times 10^{-3}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(5-Bit Uncoded Sequences)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>1719</td>
<td>1763</td>
<td>$3.87 \times 10^{-1}$</td>
<td>$4.31 \times 10^{-1}$</td>
<td>$3.12 \times 10^{-1}$</td>
</tr>
<tr>
<td>40</td>
<td>492</td>
<td>502</td>
<td>$6.60 \times 10^{-2}$</td>
<td>$8.80 \times 10^{-2}$</td>
<td>$6.76 \times 10^{-2}$</td>
</tr>
<tr>
<td>60</td>
<td>141</td>
<td>145</td>
<td>$1.00 \times 10^{-2}$</td>
<td>$1.69 \times 10^{-2}$</td>
<td>$1.05 \times 10^{-2}$</td>
</tr>
<tr>
<td>80</td>
<td>40</td>
<td>42</td>
<td>$3.32 \times 10^{-3}$</td>
<td>$5.30 \times 10^{-3}$</td>
<td>$1.50 \times 10^{-4}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(8-Bit Coded Sequences)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>2222</td>
<td>2272</td>
<td>$4.16 \times 10^{-2}$</td>
<td>$6.30 \times 10^{-1}$</td>
<td>$4.68 \times 10^{-1}$</td>
</tr>
<tr>
<td>40</td>
<td>731</td>
<td>758</td>
<td>$8.53 \times 10^{-2}$</td>
<td>$1.96 \times 10^{-1}$</td>
<td>$1.77 \times 10^{-1}$</td>
</tr>
<tr>
<td>60</td>
<td>240</td>
<td>241</td>
<td>$1.13 \times 10^{-2}$</td>
<td>$4.50 \times 10^{-2}$</td>
<td>$3.80 \times 10^{-2}$</td>
</tr>
<tr>
<td>80</td>
<td>79</td>
<td>83</td>
<td>$6.67 \times 10^{-4}$</td>
<td>$5.89 \times 10^{-3}$</td>
<td>$1.73 \times 10^{-3}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(9-Bit Coded Sequences)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table A-12

*Slepian (8,5)\(_1\) Coder (ignores bit errs. in rightmost position)*

<table>
<thead>
<tr>
<th>(\beta)</th>
<th># Bit errs. (predicted)</th>
<th># Bit errs. (actual)</th>
<th>P-probability of Word Error.</th>
<th>(\epsilon^2_*)</th>
<th>(\epsilon^2_c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>1719</td>
<td>1763</td>
<td>3.87x10(^{-1})</td>
<td>4.31x10(^{-1})</td>
<td>3.12x10(^{-1})</td>
</tr>
<tr>
<td>40</td>
<td>492</td>
<td>502</td>
<td>6.60x10(^{-2})</td>
<td>8.80x10(^{-2})</td>
<td>6.76x10(^{-2})</td>
</tr>
<tr>
<td>60</td>
<td>141</td>
<td>145</td>
<td>1.00x10(^{-2})</td>
<td>1.69x10(^{-2})</td>
<td>1.05x10(^{-2})</td>
</tr>
<tr>
<td>80</td>
<td>1</td>
<td>1(\frac{1}{2})</td>
<td>3.32x10(^{-3})</td>
<td>5.30x10(^{-3})</td>
<td>1.50x10(^{-4})</td>
</tr>
</tbody>
</table>

### Table A-13

*Slepian (8,5)\(_2\) Coder (ignores bit errs. in leftmost position)*

<table>
<thead>
<tr>
<th>(\beta)</th>
<th># Bit errs. (predicted)</th>
<th># Bit errs. (actual)</th>
<th>P-probability of Word Error.</th>
<th>(\epsilon^2_*)</th>
<th>(\epsilon^2_c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>1719</td>
<td>1763</td>
<td>3.80x10(^{-1})</td>
<td>4.87x10(^{-1})</td>
<td>3.50x10(^{-1})</td>
</tr>
<tr>
<td>40</td>
<td>492</td>
<td>502</td>
<td>6.80x10(^{-2})</td>
<td>1.86x10(^{-1})</td>
<td>1.64x10(^{-1})</td>
</tr>
<tr>
<td>60</td>
<td>141</td>
<td>145</td>
<td>1.46x10(^{-2})</td>
<td>9.20x10(^{-2})</td>
<td>8.59x10(^{-2})</td>
</tr>
<tr>
<td>80</td>
<td>40</td>
<td>45.3</td>
<td>4.00x10(^{-3})</td>
<td>3.38x10(^{-2})</td>
<td>2.98x10(^{-2})</td>
</tr>
</tbody>
</table>
APPENDIX B

WORD ERROR PROBABILITY

Equation 2.1 for word error probability given in Chapter 2. expresses the probability of codeword $A_i$ being transformed into codeword $A_j$, in terms of $p$—probability of channel bit error. The equation is repeated here.

$$P_{ij} = \sum_{r=1}^{V} p^h[A_j \oplus (S_r \oplus A_1)] q^{n-h[A_j \oplus (S_r \oplus A_1)]}$$

(B.1)

The $h \times$ notation is read as the Hamming Weight of the binary number $x$. The term $(n)$ is the number of bits/codeword, and $q = (1-p)$. Recall that Hamming Weight is simply the number of "1s" in a binary number. The rules for "Modulo-two" addition are repeated below.

$$1 \oplus 1 = 0$$
$$1 \oplus 0 = 1$$
$$0 \oplus 1 = 1$$
$$0 \oplus 0 = 0$$

Mitryayev simplifies the equation for $P$, by noting that $P_{ij} = P_{ji}$, and by using some of the properties of group codes. He also
includes a sample calculation of word error probability \( P \) for the (4,2) Slepian code.
APPENDIX C

RANDOM NUMBER GENERATORS USED IN THE SIMULATION

Two random number generators were used in the computer simulation program. "FUNCTION RANDOM" generates uniformly distributed numbers on the interval (0,1).

```
FUNCTION RANDOM(IX)
C
C   SUCCESSIVE MULTIPLICATION TO CAUSE
C   OVERFLOW AND THUS RANDOM NUMBER
C   IY=IX*65539
C
C   FOLLOWING STATEMENTS SHIFT
C   INTO POSITIVE NUMBERS
C
   IF(IY) 5,6,6
      5 IY=IY+2147483647+1
      6 YFL=IY
C
C   NORMALIZE RANDOM NUMBER TO INTERVAL (0,1)
C   BY DIVISION WITH COMPUTERS LARGEST NUMBER.
C
   RANDOM=YFL*.46566613E-9
   IX=IY
   RETURN
END
```

A test program was run which called 1000 numbers from "FUNCTION RANDOM" and listed the total amount of numbers falling in successive equal subdivisions of the interval (0,1). The output of the test program, given below, shows 505 numbers fell in the interval (0,.5), with the remaining 495 in the interval (.5,1.0). These two
subintervals are again divided into four intervals 

\((0.0, 0.25), (0.25, 0.50), (0.50, 0.75), (0.75, 1.0)\), and similarly for the remaining smaller subdivisions.

<table>
<thead>
<tr>
<th>505</th>
<th>495</th>
</tr>
</thead>
<tbody>
<tr>
<td>245</td>
<td>260</td>
</tr>
<tr>
<td>239</td>
<td>256</td>
</tr>
<tr>
<td>111</td>
<td>134</td>
</tr>
<tr>
<td>141</td>
<td>119</td>
</tr>
<tr>
<td>133</td>
<td>106</td>
</tr>
<tr>
<td>133</td>
<td>123</td>
</tr>
<tr>
<td>50</td>
<td>61</td>
</tr>
<tr>
<td>68</td>
<td>66</td>
</tr>
<tr>
<td>78</td>
<td>63</td>
</tr>
<tr>
<td>66</td>
<td>53</td>
</tr>
<tr>
<td>71</td>
<td>62</td>
</tr>
<tr>
<td>51</td>
<td>55</td>
</tr>
<tr>
<td>62</td>
<td>71</td>
</tr>
<tr>
<td>57</td>
<td>66</td>
</tr>
</tbody>
</table>

"FUNCTION GAUSSN" forms the gaussian distribution by summing random numbers from a uniform distribution. As mentioned in Chapter III, a set of sums of random numbers tends to be gaussian by the Central Limit Theorem.

```
FUNCTION GAUSSN(IX)
A=0.0
C THE LOOP THROUGH STATEMENT 50
C SUMS 48 NUMBERS FROM A UNIFORM DISTRIBUTION (0,1).
C
C THE (0,1) DISTRIBUTION IS GENERATED THE SAME AS IN
C FUNCTION RANDOM.
C
DO 50 J=1,48
IY=IX*65539
IF(IY) 5,6,6
5 IY=IY+2147483647+1
6 YFL=IY
AX=YFL*.4656613E-9
IX=IY
50 A=A+AX
C
C NORMALIZE THE RANDOM NUMBER TO ZERO MEAN.
C
GAUSSN=(A-24.0)/2.
RETURN
END
```
The same test program used for "FUNCTION RANDOM" was applied to "FUNCTION GAUSSN". The distribution is given below in subdivisions as follows. The first row subdivides the 1000 points into numbers greater than two and less than two respectively. The second line is the number of occurrences in the intervals (0,1), (1,2), (2,3), (3,3), and similarly these intervals are bi-sected for the rest of the table given below. The distribution is in close agreement with published tables of the gaussian distribution.

<table>
<thead>
<tr>
<th></th>
<th>942</th>
<th>58</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>704</td>
<td>238</td>
</tr>
<tr>
<td></td>
<td>403</td>
<td>301</td>
</tr>
<tr>
<td>200</td>
<td>203</td>
<td>159</td>
</tr>
</tbody>
</table>

NO. OF TIMES ON THE SUBDIVISION LINE = 0