NASA CR-166,402

NASA CONTRACTOR REPORT 166402

NASA-CR-166402
19820025186

Channel Coding in the Space Station Data System Network

Tim Healy

Grant NCC 2-128
May 1982

NASA

ENTER:
18 1 1 RN/NASA-CR-166402
DISPLAY 1R/2/1
82N33062*# ISSUE 23 PAGE 3342 CATEGORY 62 RPT#: NASA-CR-166402 NAS 1.26:166402 CNT#: NCC2-128 82/05/00 54 PAGES UNCLASSIFIED DOCUMENT

AUTH: A/HEALY, T.
CORP: Santa Clara Univ., Calif. CSS: (Dept. of Electrical Engineering.)
AVAIL.NTIS SAP: HC A04/MF A01
MAJS: /*CHANNELS (DATA TRANSMISSION)/*CODING/*DATA TRANSMISSION/*SPACE STATIONS /*SPACECRAFT COMMUNICATION
MINS: / DATA LINKS/ DATA SYSTEMS/ ERROR CORRECTING CODES/ PRIVACY/ SECURITY/ SYNCHRONISM
ABA: Author
ABS: A detailed discussion of the use of channel coding for error correction, privacy/secrecy, channel separation, and synchronization is presented. Channel coding, in one form or another, is an established and common element in data systems. No analysis and design of a major new system would fail to consider ways in which channel coding could make the system more effective. The presence of channel coding on TDRS, Shuttle, the

NASA CONTRACTOR REPORT 166402

Channel Coding in the Space Station Data System Network

Tim Healy
Dept. of Electrical Engineering
and Computer Science
University of Santa Clara
Santa Clara, California

Prepared for
Ames Research Center
under Grant NCC 2-128

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035

CHANNEL CODING IN THE SPACE STATION DATA SYSTEM NETWORK

Tim Healy
Department of Electrical Engineering and Computer Science
University of Santa Clara
Santa Clara, California 95053

for the

Data Management Working Group
Space Station Technology Steering Committee
National Aeronautics and Space Administration

NASA AMES RESEARCH CENTER GRANT NCC2-128

May 4, 1982

CHANNEL CODING IN THE SPACE STATION DATA SYSTEM NETWORK

Table of Contents

Section

0. SUMMARY AND RECOMMENDATIONS

This section presents a short summary of the full paper, including recommendations about the need for additional studies which should be carried out.

Channel coding, in one form or another, is an established and common element in data systems today. No analysis and design of a major new system would fail to consider ways in which channel coding could make the system more effective. (To support this conclusion, we cite the presence of channel coding on TDRS, Shuttle, the Advanced Communication Technology Satellite Program system, the JSC-proposed Space Operations Center, and the proposed 30/20 Ghz Satellite Communication System.) The eventual designers of the Space Station data system will have to consider the use of channel coding. Coding will probably be used in a number of forms in the system which eventually evolves. Hence, channel coding should be a part of the preliminary studies of space station data systems.

PURPOSES OF CHANNEL CODING

While channel coding is most often associated with error detection and correction, there are actually at least five reasons for considering its use.

1. Error detection and correction
2. Privacy or secrecy
3. Channel separation(CDMA)
4. Synchronization
5. Fault tolerance

Of the above it seems very likely that coding will be used in connection with the Space Station for the first two reasons. It is also very possible that coding could be developed to effect channel separation for the Space Station. The final two reasons are more speculative, and may not be ready for application in the Space Station time frame.

TECHNOLOGY DRIVERS

There are at least three technological reasons why the use of coding appears to be increasingly attractive.

1. There is a growing history of the successful use of coding in a wide variety of applications in both earth

and space applications.

2. The cost and size of the sophisticated electronics necessary for coders and decoders is rapidly decreasing.

3. There have been significant improvements in the efficiency of algorithms used in coding and decoding.

ERROR DETECTION AND CORRECTION

Of the five purposes listed above for coding this document is primarily concerned with error detection and correction. There are two basic approaches to error control.

1. Feedback

2. Forward Error Correction(FEC)

In the feedback approach, sometimes called ARQ (automatic repeat request) and sometimes "ACK/NACK", the transmitter sends a block of data along with some additional "check bits" which help the receiver detect errors. The receiver checks the data to see if there is evidence of error. If there is not, it sends an acknowledgement(ACK) to the transmitter which then sends its next message. If there is evidence of error, the receiver sends back to the transmitter a non-acknowledgement(NACK), and the transmitter resends the message. This type of error correction is very commonly used today in data systems over relatively short distances, up to perhaps a few thousands of miles, where the transmission time delays are small. For many space communication applications the time delays become prohibitive.

In forward error correction systems the transmitter sends more check bits than in the error detection case, enough to permit the receiver to correct a certain number of errors without the need for retransmission. The bit overhead cost is higher, but the problem of time delay is obviated.

There are a number of types or forms of FEC codes. These are summarized below.

1. Block Codes

   Parity Check Codes

   Cyclic Codes

      Bose-Chaudhuri-Hocquenghem(BCH) Codes

      Reed-Solomon(RS) Codes

2. Convolutional Codes

   Sequential Decoding

   Viterbi Decoding

   Threshold Decoding

Block codes are commonly used for variable length messages. Parity check codes find application in computer memory systems. The BCH codes are probably the most powerful codes available for correction of single random errors. On the other hand, RS codes are most effective against bursts of errors.

Convolutional codes have been very successfully applied in many space systems, where noise tends to be Gaussian rather than bursty. Sequential decoding techniques are used extensively in deep space apllications. Closer to earth, convolutional codes are used in the shuttle data system, and are a part of most of the near-earth system proposals now under consideration or development.

PRIVACY OR SECRECY

Privacy or secrecy can be effected either by separately encrypting the data stream prior to channel coding, and treating the encrypted data as any other data stream, or the encryption can be incorporated into the channel coding by giving the necessary decoding codebooks or unscrambling algorithms only to chosen receivers.

CHANNEL SEPARATION

Channel separation can be effected by assigning different codewords to different users so that a receiver is able to determine which user must have sent the message in question. In this way coding can substitute Code Division Multiple Access (CDMA) in place of FDMA or TDMA.

RECOMMENDATIONS

In summary, coding is an established technology which has been shown to improve the performance of data communication systems. Its role in the Space Station data system will have to be studied, and we believe that in the end it will have an important role in the system which is developed. Therefore, we make the following recommendations:

1. A survey of existing applications of coding, and of

the development of new hardware and software for coding should be carried out.

2. As the data needs of the space station are further defined, system studies should include consideration of the ways in which coding can improve system performance.

3. A study should be made of the possibility of using coding to obtain channel separation in the multiple user environment.

4. At least cursory attention should be paid to the question of whether coding might be useful for synchronization, or for fault tolerance.

5. Coding should be incorporated in simulation studies of the Space Station data system.

1. INTRODUCTION

The National Aeronautics and Space Administration is presently considering the development and deployment of a "space station" which would provide a platform in space for unmanned or manned scientific experiments and other activities. Such a station will require an extensive data system to provide for the exchange of data between the space station and users on earth and in space, as well as data transfer within these system components. This document presents a preliminary analysis of the role which coding can play within such a system.

Coding is a systematic method for altering the data in the system for one of a number of purposes which we will discuss in detail in Section 3. Briefly, these purposes are:

1. Error Correction
2. Privacy/secrecy
3. Channel Separation
4. Synchronization
5. Fault Tolerance

There are at least three technological reasons for seriously considering the use of coding in the space station.

1. There is a history of successful use of coding over the past one to two decades in applications similar to some of those which will be found in the space station.
2. The cost effectiveness of integrated circuits which can be used to implement coders and decoders is increasing.
3. Significant improvements in decoding algorithms continue to appear.

Of the five purposes given above for coding, we will consider only the first four in any detail in this paper. The issue of fault tolerance is raised elsewhere. The only reason for discussing the point briefly here is to point out that in theory coding used for other purposes could also improve fault tolerance. There are at least three basic approaches to fault tolerance.

1. Make every effort to improvement system components or parts so that the probability of failure is as low as possible.

2. Provide for redundancy of parts so that a failure can be masked, and hence ignored by the system as a whole.

3. Encode data signals in such a way that component failures are reflected in changes in signals so that fault decoders can correct for these failures.

The last of these approaches could in principle be combined with coding for other purposes. If a system element fails, one of three actions is possible.

1. Correction of the inaccurate data by means of error correcting codes.

2. Diagnosis and replacement of faulty component.

3. Reorganization of the data system network(probably with some degree of graceful degradation.)

We shall not discuss fault tolerance in any more detail in this paper.

We begin this paper with a discussion, in Section 2, of the assumptions we will make, as well as some definitions. Section 3 then explains briefly how coding can accomplish each of the purposes listed above. Some of the basic theory of coding is reviewed in Section 4. In Section 5 we turn to a discussion of a number of practical coding schemes. Some applications which have been made of coding are given in Section 6, and some preliminary recommendations concerning the space station are then given in Section 7.

In concluding this section we raise the rhetorical question of whether coding should be seriously considered for the space station. The answer is yes if it appears that coding can be successfully implemented in the space station timetable, and if the purposes of the coding cannot be met in any more effective manner. It is our opinion that both of these conditions will be met in some of the possible applications within the system. Hence the question will not be <u>if</u> coding will be applied, but rather <u>where</u> it will be applied.

2. System Assumptions

The purpose of this section is to present some assumptions which will be used in this paper, and some definitions which are intended to simplify the discussion. We define a number of terms at this point.

2.1 System Definitions

The following definitions will be used in the discussion in this section.

Data System

The data system is the set of all hardware, and the associated software, protocols, and controls, used to exchange data between the space station and other system components, and also to store, process, and move data within these components.

Components

Components of the data system are any major susbsystems which it is convenient to isolate and define. Examples are the space station, a deep-space scientific probe, the Shuttle, TDRSS, an earth-bound sensing and telemetry station, a space station satellite operating near the space station for scientific or logistic reasons.

Component Elements

Component elements are subsystems of components which it is convenient to isolate or define. Examples are a memory bank in the space station, an encoder in a deep-space probe, a digital filter in an earth station, etc.

To illustrate some of these definitions, Figure 2.1 shows an example of a space station data system with some major components. Also shown are the ionosphere and atmosphere which separate earth and space components of the system. Dashed lines indicate radio paths or channels between components. Solid lines suggest hardwire channels.

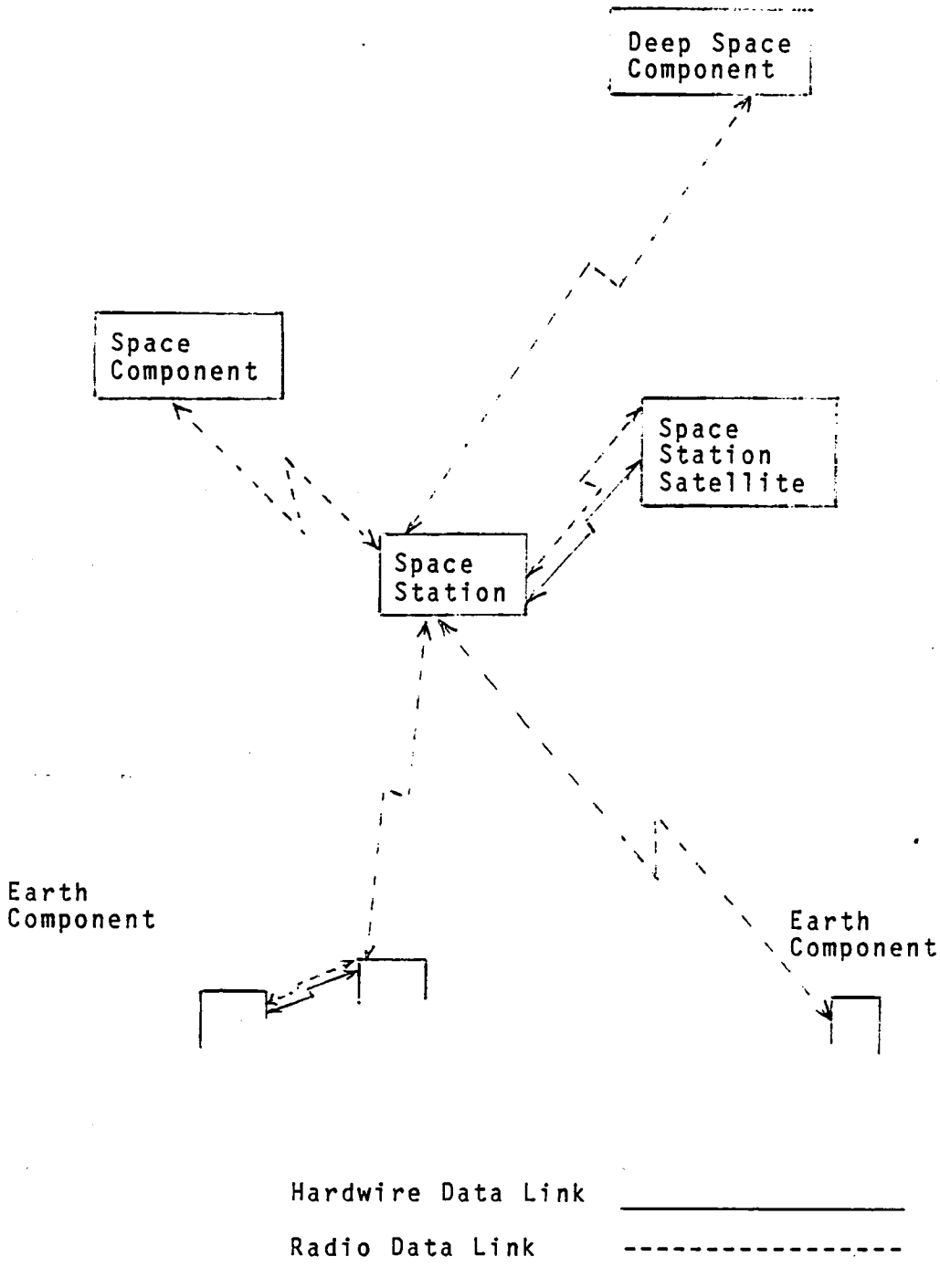


Figure 2.1 Space Station Data System

2.2 Perfect and Imperfect Channels

As a general rule it is likely, and we shall assume, that most data exchange between system components will be over imperfect channels, subject to noise, interference, and signal distortion. On the other hand, data movement within a component, that is, between component elements, will probably be over channels which can be assumed perfect, that is, in which the above sources of degradation can be neglected. The primary implication of this assumption is that coding will not be necessary for error correction purposes within a component, though it may still be used for channel separation or fault tolerance. However, for the exchange of data between components, coding may also be used for error correction, and possibly for synchronization, as well as for the other purposes mentioned.

2.3 System Variations in Time

It is assumed here that the system is time-variant over short, medium, and long periods. Over short periods, on the order of seconds or minutes, the system varies due to changes in the environment, such as propagation conditions, fading, etc. Over medium periods, on the order of minutes to days, the system changes due to element failures, with graceful or catastrophic degradation, and with the addition or deletion of system components. For example, when the Shuttle approaches the space station, it becomes a component of the data system. Likewise, if a ground station fails, or is removed from operation, it ceases to be a component, or at least an active component. Over long periods of time, on the order of weeks to years, the system changes because some elements or components become obsolete, and others are added due to changes in mission requirements, and to the evolution of technology.

The primary implication of these time variations in the system is that the system must be designed to adapt to these variations over the short term and the long. While system adaptibility is not a primary topic of this paper, it should be noted that any coding scheme which is implemented should be sufficiently robust to survive such variations over the life of the system.

3. REASONS FOR THE USE OF CODING

In Section 1 we indicated five purposes or reasons for coding data in the space station data system. In this section we briefly review what coding means, and then show how coding can achieve each of these five purposes.

3.1 The Meaning of Coding

Let us assume that the data which is of interest to us and is to be moved within the system can be represented as a sequence X of binary signals represented by the symbols 0 and 1. For example, such a sequence of length 12 is:

X = (0 1 0 0 1 1 1 0 1 0 1 0)

Coding is simply a systematic method of mapping this sequence into another sequence of the same or different length. For example, one such mapping might yield the new sequence:

Y = (1 0 1 1 0 0 0 1 0 1 0 1)

In this case the mapping or encoding rule simply changes 0 to 1 and 1 to 0. A slightly more complex coding algorithm yields:

Y = (0 1 1 0 0 1 0 0 1 0 0 0 0 1 1 0)

In this case we consider three message symbols at a time, keeping the first two unchanged, reversing the third, and adding a 0. Neither of these simple codes are very interesting. They simply illustrate the concept of coding as a mapping. In Section 5 we shall consider some practical coding schemes.

Before we see how coding can help us, let us consider an alternate way of viewing binary data which may help clarify the explanations which follow. Suppose that the data of interest is represented by a matrix of binary values of dimension r by m, as shown in Figure 3.1. This can be thought of as equivalent to a sequence of length mr since the first r terms in the sequence could make up the bottom row, the next r the second row, and so on up to the mth row. Such a matrix could be generated in many other ways. One, suggested by the time and frequency axes in Figure 3.1, is to generate a set of 0 to m frequencies in the bandwidth W during each of r time slots, letting the shaded

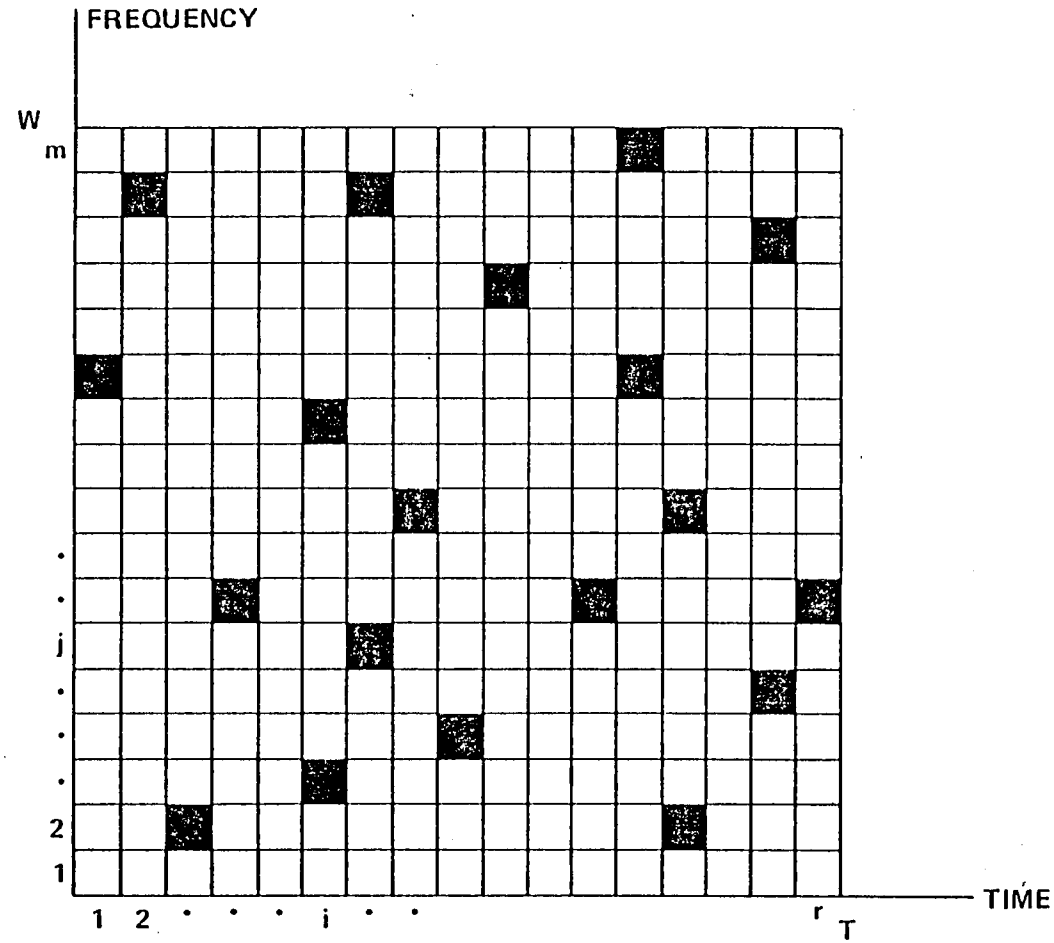


Figure 3.1 Matrix Coding by Frequency Hopping

square imply that that frequency was sent during that time slot. The resulting pattern of 0's and 1's then represents the data to be transferred form one place to another.

Again, the reason for introducing the matrix of Figure 3.1 is to emphasize that the sequence can be thought of as a pattern. This should make it easier to see how the purposes of coding described below are achieved.

Before leaving the matrix it is interesting to note the large number of distinct patterns which can be represented by a relatively small matrix. For example, the matrix as shown in Figure 3.1 has 289 squares. The number of unique patterns it can represent is:

$$n = 2^{289} = 10^{87}$$

Even a much more modest 4 by 5 matrix, equivalent to a sequence of length 20, can represent over one million distinct patterns.

We shall next see how sequence (or matrix) patterns can accomplish the desired purposes indicated earlier.

3.2 Error Correction

Imperfect channels have a tendency to cause errors in individual bits or symbols due to noise, interference, or distortion. One of the purposes of coding is to detect or correct these errors by treating a sequence of bits as a group with some known pattern. Then it may be possible to detect a single error by observing the received sequence, and changing any single bits which do not fit a probable pattern. As a very simple example, assume that the sender(encoder) may only send a pattern which is a solid horizontal row in the matrix of Figure 3.1. Since there are 17 such rows in this case, the sender may select any one of 17 messages to send. Now suppose that the receiver(decoder) receives a matrix in which all of the 17 squares in the second row are filled except one, which is in Row 3. A prudent decoder would no doubt conclude that the encoder had sent the message corresponding to Row 2. Of course the decoder might be wrong. The sender may have sent Row 3, and the channel made 16 errors and got one signal correct. However, that is extremely improbable, so the decoder will probably be correct in choosing Message 2.

There are a number of difficulties with this pattern coding scheme. First, note that the encoder was only able to send 17 different messages, even though as we saw earlier the number of unique sequences in the matrix is:

$$n = 2^{289}$$

The advantage of having only 17 messages, and choosing this form for them is that they look very different from each other. It is very unlikely that enough errors can be made to make one of these rows look like another. But if we try to increase the set of patterns which the encoder is free to use, the difference or distinguishability between patterns decreases. And in fact if we used all possible patterns, then no errors could be detected since a single error would make the pattern look like another acceptable pattern. Hence, the secret of successful decoding is to make the patterns appear as unalike as possible, that is to limit the acceptable number of patterns. Because some patterns can never be sent, the number of different messages which can be sent decreases, and hence the average data rate decreases.

A second difficulty arises if we attempt to implement a coding and decoding scheme in this manner. It will be necessary for both the encoder and the decoder to store all of the allowed sequences (called "codewords") in a "codebook". The decoder, on receipt of a sequence or matrix, compares all of the codewords in the codebook with the received sequence, and chooses that codeword, and hence message, which is closest to the received sequence. But if the number of codewords is at all large, so that the data rate can be high, the task of storing the codebook, and then making all of these comparisons, is likely to be beyond the capacities of any practical system.

This last difficulty is so great that this approach is not used in practical systems. Another set of approaches, which can be implemented, has evolved over the years. These are discussed in Section 5. However, the codebook concept does lie at the heart of much of the theory which is used to determine theoretical bounds for coding, which we briefly review in Section 4.

3.3 Privacy/Secrecy

The same basic concept of structuring a sequence or giving a pattern to a matrix can also be used to effect coding for secrecy or for privacy. The terms secrecy and privacy refer to the same basic end result. However, privacy usually implies that a more serious effort is made to keep the message secure.

There are essentially two ways in which the sequence can be used to effect secure communication. First, if a codebook approach, as discussed above, is used, the appropriate codebooks can be distributed only to those persons who are intended to be a part of the secure communication. Other users or eavesdroppers would not know which patterns applied to which messages.

The second method of effecting secure coding makes public the

basic code patterns generated by each sender. However, before each pattern is sent, the matrix is scrambled or reorganized in some systematic manner known only to the encoder and desired decoder. The scrambling rule or algorithm can be thought of as an address, or as a secret envelope, which permits the message to be sent only to the proper source. This latter approach is essentially what is used in most communication systems having a need for secrecy or privacy. The codebook approach is usually not very attractive of the need to deliver the codebook to both parties, and because the security of the system goes down with each use.

It is important to note that secrecy coding or addressing has the effect of using up available patterns in the matrix, and hence in effect, of reducing the data transmission rate for a given user.

3.4 Channel Separation

Channel separation is necessary for multiple-access applications where more than one user wishes to transmit or receive signals. Instead of being given a specific frequency (frequency division multiple access), or a specific time slot (time division multiple access), a sender is given some mix of frequency and time slots. This is called code division multiple access. There are a number of ways to do this. A user may be assigned a portion of the matrix. In Figure 3.1, perhaps User 1 is given the 4 by 4 submatrix in the lower lefthand corner. No one else is allowed to radiate into this slot.

Another approach is to give each user a unique codebook, and proceed with the codeword matching process describe above.

A third approach is to give each user a unique scrambling algorithm which then acts as an addressing mechanism. These latter two approaches are of course identical essentially to the schemes discussed under secrecy/privacy.

3.5 Synchronization

It is also possible in principle to use this matrix coding approach to effect block synchronization, though the actual implementation of this approach to synchronization might well not be attractive compared to other methods. The concept is simply to permit only patterns which do not look like other patterns which have been shifted by i columns, where i lies between 1 and r-1. (See Figure 3.1) Then the decoder will not accept a match of codewords until the bits are in the appropriate locations. Block synchronization is then obtained.

Once again we note that this process has the affect of reducing the number of possible signals which can be sent, and hence the

data rate.

3.6 Fault Tolerance

Finally, it is possible to use coding to obtain tolerance to some kinds of faults. Suppose that a certain kind of failure had the affect of changing a bit or symbol in a sequence. If a pattern were observed with this error, the encoder could do one or both of two things. First, it could change the errored bit and hence mask the failure. Second, it could send some kind of a system corrective maintenance message to the system control.

Before we leave this section, it is of value to summarize the results, indicating what we have and have not done.

1. We have shown how coding can, at least in principle, accomplish the five purposes of coding.

2. All of the tasks discussed have the affect of using up available structure in the set of allowed patterns, and hence of reducing the data rate, all else remaining equal.

3. We have not given practical implementations. This will come in Section 5.

4. CODING THEORY LIMITS

We saw in Section 3 how the coding or structuring of sequences could be used to correct a certain number of the errors caused by imperfect channels. Shannon (1948) obtained an upper limit or bound on the rate at which messages could be sent over a given channel, such that the probability of error goes to zero. This bound is obtained by allowing the sequence length n (the number of squares in the pattern matrix) to go to infinity. The approach which Shannon used in this argument cannot be implemented in practice, but the bound is nonetheless of great interest. We begin with the simple case of a single-user channel with Gaussian additive noise, and then turn to the multi-user case.

4.1 Single-User Channels

Consider a channel, as shown in Figure 4.1, with input X and output Y.

Figure 4.1 Single-User Communication Channel

Shannon defined the "mutual information", I(X,Y), of this channel, and also established the relation:

$$I(X,Y) = H(Y) - H(Y|X) \tag{4.1}$$

where H(Y), the "entropy" of the random variable Y is:

$$H(Y) = -\sum p(yi)\log p(yi) \tag{4.2}$$

where p(yi) is the probability that the random variable Y takes on the value yi, for all possible values of i.

Before proceeding, let us review the meanings of entropy and mutual information. Entropy is a measure of the uncertainty associated with the random variable Y. If there is only one possible value for the variable, then the probability of that value is unity, and the entropy is zero. That is, there is no uncertainty. On the other hand, if there are a number of possible values of Y, and they are equally likely to occur, the entropy is quite large. It is necessary that a variable have non-zero entropy if it is to convey information. If the outcome of the communication is certain before the communication takes place, then no information is provided by the process. The larger the entropy the greater is the average information which is provided by the communication process.

The mutual information between X and Y is a measure of the degree to which X and Y are related, or correlated. If the noise and disturbance on the channel is so bad that Y bears no relation to X, that is, is caused only by the randomness introduced by the channel, then the mutual information between X and Y is zero. On the other hand, if the channel is perfect, so that there is a deterministic (not necessarily equality) relation between X and Y, then the mutual information is as large as is possible for the particular distribution which Y has.

With these two ideas in mind, let us look again at Equation 4.1. This equation says that the mutual information between X and Y is the uncertainty associated with Y minus the uncertainty of Y given that you know X. Suppose first that the channel is perfect so that X and Y are perfectly correlated. Then if you know X, there is no uncertainty about Y, and the second term in 4.1 is zero. Hence the mutual information between X and Y is as large as it can be. But, if the channel is very bad, so that X and Y are uncorrelated, the uncertainty on Y given X is just the uncertainly on Y since knowing X doesn't help. Hence H(Y|X) is just H(Y), and I(X,Y) is zero.

Shannon next defined the "channel capacity" as:

$$C = \max_{p(xi)} I(X,Y) \qquad (4.3)$$

where max means the maximum value of I(X,Y) over the input random variable distribution p(xi). The channel is assumed to be fixed in nature. It is either perfect or has some statistical measure of imperfection, such as an additive noise, or a probability of error. However, the probability distribution of the input random variable X is itself variable,

and the maximization indicated by Equation 4.3 is taken over this variable to find the largest possible value of mutual information. The result is called the channel capacity.

Next Shannon used an extraordinary argument to prove that the input X could be coded in such a way as to reduce the probability of error to zero if the information rate R of the input X is less that the channel capacity C, and further that if the information rate exceeds the capacity, the probability of error cannot go to zero. The next logical step in the developement is to calculate the channel capacity of some particular channels of interest. In the following we shall review the results for two cases.

4.2 Additive Gaussian White Noise Channel

In the first special case which we consider, the channel simply adds Gaussian white noise to the input, as shown in Figure 4.2.

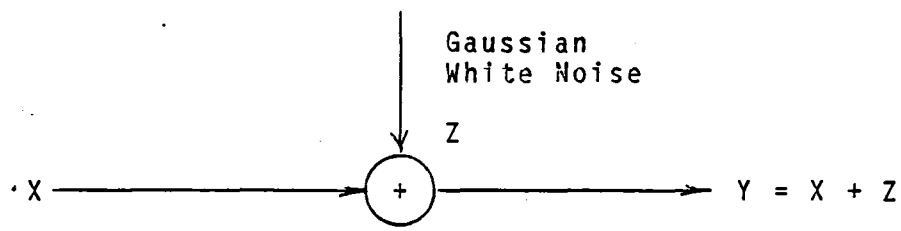


Figure 4.2 The Additive Gaussian White Noise Channel

When the maximization indicated in Equation 4.3 is carried out for this case, the capacity, for the bandlimited channel with bandwidth W, signal power P, and noise power N, turns out to be:

$$C = W\log_2(1 + P/N) \tag{4.4}$$

The input distribution on X required to achieve this capacity is found to be the continuous Gaussian. This fact makes the results of limited value since it is seldom convenient to constrain the input to follow a Gaussian law. A somewhat more interesting case is that in which the input is binary, that is, it can take one of only two values. We consider this next.

4.3 The Binary Symmetric Channnel

The next case is illustrated in Figure 4.3. The input X can

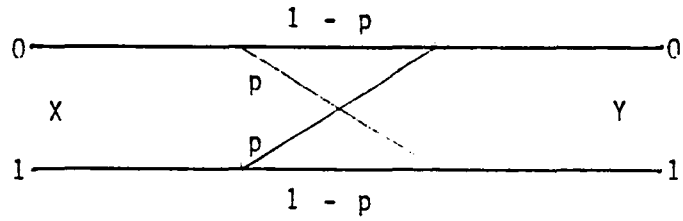


Figure 4.3 The Binary Symmetric Channel

take values of 0 or 1. In going through the channel, either of these values changes to the other with probability p. It does not change with probability 1 - p. Thus, p is the probability of error. The channel capacity for this case turns out to be (See, for example, McEliece[1977].):

$$C = 1 - h(p) \quad \text{bits per binary symbol} \tag{4.5}$$

where h(p), the uncertainty (or entropy) introduced by the channel, is:

$$h(p) = -p\log p - (1-p)\log(1-p) \tag{4.6}$$

If the units of h(p) and C are to be bits per symbol, the logarithm is taken to the base 2.

If p is 1/2, which means that half of the bits are in error on the average, h(p) is one, so the channel capacity is zero. If p is zero, h(p) is zero and the capacity is one. If the channel bandwidth W is assumed to be equal to one divided by the bit duration, a rather common assumption, then the channel capacity is:

$$C = W[1 - h(p)] \tag{4.7}$$

4.4 The Multiple-Access Channel

The final example we shall consider is the multiple access channel, illustrated in Figure 4.4, in which n users attempt to send messages to a single receiver by transmitting binary signals. The multiple access problem has been studied extensively over the past decade. Some of the primary references are: Ahlswede(1971), Chang and Wolf(1981), Cohen, Heller and Viterbi(1971), El Gamal and Cover(1980), Farrell (1981), Goodman, Henry and Prabhu(1980), Gyorfi and Kerekes (1981), Matsumoto and Cooper(1981), Timor(1981), Van der Meulen (1977), Wolf(1981), and Yue(1980).

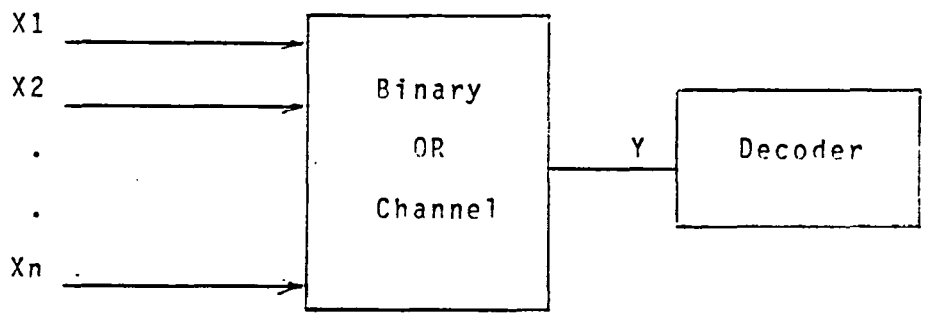


Figure 4.4 The Multiple Access OR Channel

We assume here that the system is block synchronized, that is that the codewords transmitted by all of the n users arrive at the same time (corresponding symbols are lined up), or that the receiver can reconstruct this order. We also assume that the system is bit or symbol synchronized, that is, that all symbols start at the same time.

The channel is a so-called OR channel. It's output Y is zero only if all of the inputs are zero. If one or more inputs are one, the output is one. Hence, the assumed channel is not sensitive to the number of inputs which are one. Such a channel is characterized by mutual interference from the users since the receiver cannot tell which sources transmitted a one if more than one is sent. This mutual interference reduces the effectiveness of any given user. We shall see a mathematical expression of this shortly.

Finally, we assume that channel noise is zero, or more accurately that the interference dominates the channel noise, so that the latter can be neglected. (For an interesting discussion of the interference effects in spread spectrum multiple access, see Scales(1980).)

In this multiple access case the channel is characterized by a so-called "capacity region" which gives bounds on the rate at which the various users can tansmit information.

For the case where each of the n users does not transmit a binary signal with probability q, it can be shown (Healy(1982)) that the capacity region is:

$$\begin{aligned} R1 &\leq q^{n-1}\, h(q) \\ &\vdots \\ R1+R2 &\leq q^{n-2}\, h(q^2) \\ &\vdots \\ \sum_{i=1}^{j} Ri &\leq q^{n-j}\, h(q^j) \\ &\vdots \\ \sum_{i=1}^{n} Ri &\leq h(q^n) \end{aligned} \qquad (4.8)$$

Let us consider the last line in Equation (4.8), the sum of the rates of all n users. It can be shown that this rate is H(Y). It's maximum value is one, which it takes when P(Y=0) and P(Y=1) are both one half. This has a very interesting interpretation in terms of the binary matrix in Figure 3.1. It means that the probability that one one square is black (one or more users is transmitting at this time and frequency) is one half. Hence, on the average half of the squares will be black. If the average were made less than half, the interference would be lower, but the users would have to transmit less often, and hence their information transmission rate would go down. If more than half of the squares are black on the average, the attempted transmission rate is higher, but so is the interference.

For the optimum case, where P(Y) = 1/2 for both 0 and 1, H(Y) is unity. Hence, the sum of the rates of all users is one bit per symbol. This is the same as the single-user case with no noise. This result tells us that the coding is able to mask the interference completely. Hence, the net spectral efficiency of this ideal code division multiple access system is 100%. The system uses the spectrum as efficiently as ideal frequency division and time division multiple access systems.

These theoretical results are most encouraging. They provide us with an upper bound on the transmission data rate for the single user case. Furthermore, they indicate that ideally coding can recover all of the lost spectral efficiency due to interference effects in the multiple access case. However, the proofs used to obtain these results use code lengths which go to infinity, implying an implementation with infinite delay and infinite memory, and hence infinite cost.

Encouraged by coding gains promised by these theoretical results, and yet sobered by the impossibility of implementation, we turn to some practical approaches to implementation.

5. PRACTICAL ERROR CONTROL SCHEMES

We saw in the last section that it is possible in theory to reduce the probability of error in transmitting a message over a channel to zero by coding and decoding the message in an appropriate manner. However, the coding technique requires that the length of the message blocks be infinite, implying infinite time delay and cost. In this section we consider some practical approaches to coding which exhibit coding gains which are more modest than the ideal, but which can be implemented at finite cost. Real channels introduce errors in messages due to noise, interference, and distortion. It is convenient to identify three essentially different approaches to controlling these errors.

Channel Improvement

A very basic approach is to improve the characteristics of the channel. This is usually synonymous with increasing the signal to noise ratio.

Error Detection

In this approach errors are detected through the use of codes capable of indicating that an error has occurred, but not how it can be corrected.

Forward Error Correction

In this case the message is encoded in such a way that errors can be corrected as well as detected.

In the remainder of this section we shall discuss each of these approaches, with an emphasis on the last of the three. For the reader who wishes more detail than is available in this paper, the literature is extensive. Some of the more readable surveys are: McEliece(1977), Lin and Costello(1981), Wiggert(1978), and Berlekamp(1980).

5.1 Channel Improvement

There are three approaches to channel improvement. These are:

1. Choose an alternative channel with less disturbance.
2. Increase the signal power in the existing channel.
3. Decrease the noise in the existing channel.

Sometimes it is possible to discard a poor channel and select one with less noise or interference. A common example is in the telephone system. If the vagaries of switching happen to provide one with a particularly bad channel, it is possible to hang up and redial. As another example, some radio systems have channel diversity which permits selection of an alternate path or channel if one becomes unacceptable, perhaps because of fading or excessive noise. On the other hand, a unique deep-space channel would not permit such an approach. The costs of this channel-switching approach are in the need for alternate channels, for systems to detect low channel quality, to search for alternates, and to switch to the superior channel, and also in the time delay associated with this process.

If it is not possible to discard a poor channel, it may be possible to enhance the quality of the existing channel by increasing the signal to noise ratio. Sometimes the signal power can be increased. This will of course have some cost associated with it, which will have to be compared with the cost of alternative approaches. In extraterrestrial space communication components power is usually quite expensive or limited because of the limited number and size of solar cells, or other power sources.

Sometimes an existing channel can be improved by decreasing the disturbances which it introduces. Noise may be reduced by decreasing the temperature of receiver components, by using devices with lower noise figures, by better filtering, or by controlling outside sources of noise. Interference may be reduced by use of special purpose hardware such as equalizers, or by external control of the sources of interference, particularly if these sources are a friendly part of the overall system under design. Distortion can be reduced by improved filtering.

5.2 Error Detection

The simplest error detection scheme uses a single parity check bit. This is illustrated in Figure 5.1.

Message	Code	Code with Parity
A	00	001
B	01	011
C	10	101
D	11	110

Figure 5.1 Parity Check Example

A communication terminal sends one of four messages, called A, B, C, and D, coded as shown in the center column. In the righthand column a third bit, called a parity bit, has been added. It is formed by taking the sum of the code bits, modulo-2. Equivalently, it is set to zero if the number of ones in the codeword is even, and to one if the number of ones is odd. At the receiver the code bits and the parity bit are added. If no error has been made, this sum is zero. If exactly one of the three bits is in error, the sum is one, and hence the error is detected. Unfortunately, if two errors are made, this fact will not be detected.

The above code is called a "block code" because a block of data of fixed length is tranferred for each message. This block might be quite short, as in the above example, representing perhaps a single character in an alphabet. Or the block could contain thousands of bits, representing perhaps many sentences or paragraphs of text, or columns of numbers, etc. The ability of a block code to detect errors can be enhanced by increasing the number of parity bits, each of which would then check some part of the message. Adding parity bits has of course the effect of increasing the total number of bits which must be sent, implying a decrease in the message data rate, or an increase in the bandwidth, or some combination of the two.

While the concept of adding parity bits to message bits as a check on the accuracy of the message bits is a useful way to look at block coding, it is instructive to consider an alternative viewpoint. If we look again at the original code in the center column of Figure 5.1, we see that we have used all four of the possible combinations of 0 and 1. However, in the case of the codewords with parity, there are three bit positions with eight possible combinations. The parity coding process has effectively chosen four of the eight, with a very interesting result. Each of the chosen four is different from any of the others in <u>two</u> positions. Hence, if a single error is made, the resulting word does not look like any of the chosen codewords, and the receiver knows that an error has been made. The effect of the parity coding has been to "push the codewords away from each other." It is this separation which allows us to detect errors, and, as we shall see later, can also allow us to correct errors under some circumstances. The more parity bits used, the greater can be the separation or distance between codewords, and hence the greater the ability to detect and correct erors.

Once an error has been detected, it can be noted and ignored, or an attempt can be made to correct it. It may be necessary to ignore errors, for example, in the case of a very deep space probe, involving long time delays, where data is not stored long enough to be retransmitted. Often, however, it is possible to correct the message. One approach which is widely used is called ARQ (automatic request for retransmission). Actually this term is ofter used synonymously with error detection. In

ARQ the receiver uses parity bits to check each block of data for errors. If no errors are found, it sends a message to the transmitter acknowledging the data, and essentially requesting the next block (ACK). If an error is found, however, a non-acknowledgement message (NACK) is sent, and the sender retransmits the block.

One of the major problems with ARQ is the time delay associated with the two-way acknowledgement process. If sender and receiver are on earth, this delay may be on the order of microseconds or milliseconds, and may be quite acceptable. But, if space communication is involved, the delay might be prohibitive. For example, the round-trip time between sender and receiver via synchronous satellite is on the order of one half second. This delay is often much longer than a block, and can have a disasterous effect on the average rate at which data can be sent.

Error detection schemes are used very extensively in data communication systems between senders and receivers that are relatively close, such as in many earth-based situations. It is extremely likely that parts of the space station data system would make use of such techniques. About the only thing that might change this is if the much more complex error correcting schemes become very inexpensive due to decreases in the cost of microelectronics, or if overall system design considerations dictate a uniform error control scheme for all parts of the system. In that event the need for error correction in some cases might eliminate the option of using some error detection.

5.3 Forward Error Correction - Block Codes

The basic concept in error correction, or "forward error correction" (FEC) schemes is to transmit codewords of such structure that the receiver will be able to deduce the correct message even though the channel has caused one or more bit errors. These codes can take one of two general forms:

Block Codes

Convolution Codes

Block codes, as we saw above, take a block or unit of data of some fixed length, and associate with it a code sequence or codeword. At the receiver, the entire message is decoded as a unit. Convolution codes, on the other hand, treat the data as a continuous stream, with no beginning or end, interleaving data bits and check bits. In this section we consider a number of types of block codes.

Parity Check Codes

We begin with what is perhaps the simplest of the block codes, called parity check codes. As noted above, data blocks can be of most any length, depending on a number of design considerations. We shall designate the length of the data block, without the check bits, as k. Such data blocks are encoded by being mapped into codewords of length n where n > k. We say that a code with n total bits per codeword, of which k are message bits, is an (n,k) code. It is also said to have a "rate" of k/n, or it is called a "rate k/n" code. If we designate data bits by D, and check bits by C, we can represent a general codeword mapping of a (5,3) code by:

$$D_1D_2D_3 \qquad C_1C_2C_3C_4C_5 \tag{5.1}$$

As another example, the code shown in Figure 5.1 is a rate 2/3 code.

If the data bits appear explicitly in the codeword (if, for example, $D_1 = C_1$, $D_2 = C_2$, and $D_3 = C_3$, in the above), then the code is said to be "systematic". The code in Figure 5.1 is systematic. As we saw earlier, it is not necessary to think of the block as containing k explicit data bits and n - k check bits. A more abstract and powerful view of Equation 5.1 is that three data bits have been mapped into five code bits with no necessarily obvious relation between the two blocks. Perhaps one mapping is (011 11001). The important thing is that there be as great a distance as possible between the codewords so that as many errors as possible can be detected and corrected.

To get a better feeling for the process of selecting dissimilar codewords, let us consider the number of possible words, and the number of actual codewords, available to a (n,k) code. If a binary code sequence has length n, the number of possible unique sequences is 2^n. And, similarly, a sequence of k bits can take any of 2^k forms. For example, in a (7,4) code, we select 16 codewords from a total of 128 possible words. A rather famous example of a block parity code is the Hamming (7,4) code shown in Figure 5.2. From the parity viewpoint the first four bits can be thought of as data, and the last three as parity checks, representing respectively the modulo-2 sum of data bits 2, 3, and 4 for C_5, 1, 3, and 4 for C_6, and 1, 2, and 4, for C_7.

0	0	0	0	0	0	0
0	0	0	1	1	1	1
0	0	1	0	1	1	0
0	0	1	1	0	0	1
0	1	0	0	1	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	1
1	0	0	1	1	0	0
1	0	1	0	1	0	1
1	0	1	1	0	1	0
1	1	0	0	1	1	0
1	1	0	1	0	0	1
1	1	1	0	0	0	0
1	1	1	1	1	1	1

Figure 5.2 Hamming (7,4) Code

Note that each codeword differs from every other in at least three places. Suppose that the first word was transmitted but an error was made in bit number three. This errored codeword looks something like the third word, for example, but it still looks more like the first word than any other. So the receiver would decide that the first word was sent, thus correcting the single error made by the channel.

We refer to the number of places in which two codewords differ as the "Hamming distance". For example, the Hamming distance between the first two words in Figure 5.2 is four. In this example the Hamming distance varies betwen a minimum of three and a maximum of seven. A block code with a minimum Hamming distance of $d + 1 = 2c + 1$ can detect d or less errors, or it can correct c errors. Since the minimum Hamming distance of the (7,4) code above is 3, it can correct one error, or detect two.

Cyclic Codes

Cyclic codes are a special type of block codes with the property that all cyclic or end-around shifts of a codeword are also codewords. As an example, consider the codeword:

0 0 1 0 1

In a cyclic or end-around shift, all of the bits shift one position to the right except the bit on the right end, which shifts around to the position on the far left. Hence, the following are also codewords:

1 0 0 1 0

0 1 0 0 1

1 0 1 0 0

0 1 0 1 0

An important feature of cyclic codes is that encoding can be implemented quite simply, using linear shift registers. An example of a shift register encoder for a (7,4) cyclic code is shown in Figure 5.3.

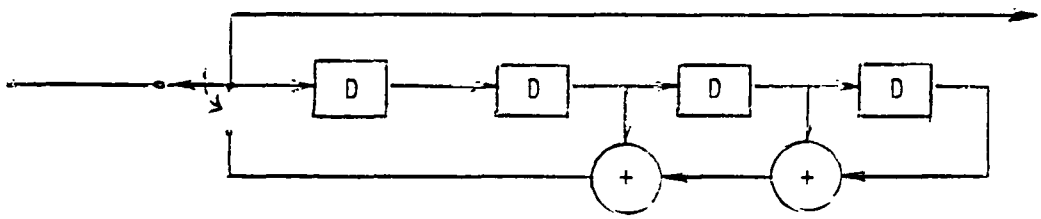


Figure 5.3 A (7,4) Cyclic Code Generator

With the switch on the left in the position as shown, a four-bit message enters the shift register, and also appears at the output. (Hence, the code is systematic.) Suppose, for example, that the message, and hence the contents of the shift register after four clock periods, is:

0 0 1 1

reading from right to left. The switch then moves to the other position, disabling the input. The final three bits in the codeword are generated by what has become, with the change in switch position, a feedback shift register. The inputs to the first stage of the register are the check bits. With the message above in the register when the switch changes, successive states of the register are:

0 0 0 1

1 0 0 0

0 1 0 0

where the bits on the left are the check bits, sent to the output. Thus the full channel codeword is:

0 0 1 1 0 1 0

From the cyclic property the full codeword set is:

0 0 0 0 0 0 0	1 1 1 0 0 1 0
0 0 1 1 0 1 0	0 1 1 1 0 0 1
0 0 0 1 1 0 1	1 0 1 1 1 0 0
1 0 0 0 1 1 0	0 1 0 1 1 1 0
0 1 0 0 0 1 1	0 0 1 0 1 1 1
1 0 1 0 0 0 1	1 0 0 1 0 1 1
1 1 0 1 0 0 0	1 1 0 0 1 0 1
0 1 1 0 1 0 0	1 1 1 1 1 1 1

A number of good cyclic codes have been discovered by coding theorists. These include Bose-Chaudhuri-Hocquenghem (BCH) codes, Reed-Solomon (RS) codes, geometry codes, and Fire codes. It also turns out that Hamming codes can be expressed in cyclic form. We shall briefly discuss the first two of the above.

Bose-Chaudhuri-Hocquenghem Codes

BCH codes are a special class of cyclic codes which result from some particular restrictions on the way in which the codes can be generated. The details of these restrictions will not be discussed here. The effect of these restrictions is to give BCH codes their particular advantages which we shall mention shortly.

First, we must discuss the ways in which errors can occur in data sequences. Errors in data strings may occur independently, that is with no relation between an error in one data symbol and the presence of an error in any other symbol. Such errors occur when the time length of the disturbance which causes the error is short compared with the symbol length. Alternatively, errors may occur in groups or bursts. This happens when the disturbances are much longer that the duration of the symbol.

The primary advantage of BCH codes is that they are considered to be the most powerful codes available today for correcting random or independent errors.

Reed-Solomon Codes

Another class of cyclic block codes, which are also a class of

BCH codes, are Reed-Solomon codes. These are non-binary BCH codes. That is, they operate on data which can have more than two possible levels.

The primary advantage of R-S codes is their superior ability to correct bursts of errors. Nonetheless, implementation problems have slowed the use of these codes as it has been difficult to build hardware to manipulate the non-binary symbols. Today, however, new approaches to implementation, including the use of charge-coupled devices, and integrated circuits, are resulting in increased use of these very attractive codes.

5.4 Forward Error Correction - Convolutional Coding

As we saw above, block coding work with fixed-length blocks of data which are mapped into codewords with the property that these blocks are well separated. Convolutional codes, however, work with data continuously.

A simple way to begin a discussion of convolutional codes is to consider the operation of an example of a convolutional encoder, such as that shown in Figure 5.4.

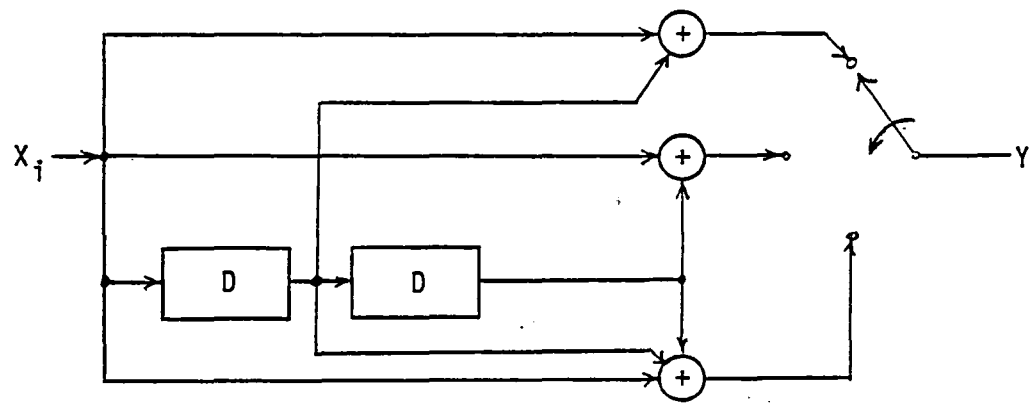


Figure 5.4 A Rate 1/3 Convolutional Encoder

The adders shown are modulo-2. Each of the two blocks marked D delay the message bit Xi for one message bit duration. During one message bit duration the switch on the right cycles through its three positions, producing three code bits, each of which has a duration equal to 1/3 that of the message bit. With the output switch in Position 1 the ith message bit enters the encoder. The topmost summer adds to it the previous message bit (that is, the output of the first delay block). The output of the switch in Position 1 is called Y1i. Hence:

$$Y_{1i} = X_i + X_{i-1} \qquad (5.2)$$

The switch moves to Position 2, and the middle summer yields the sum of X_i and the input delayed by <u>two</u> delay times, that is, X_{i-2}.

$$Y_{2i} = X_i + X_{i-2} \qquad (5.3)$$

The final output value is:

$$Y_{3i} = X_i + X_{i-1} + X_{i-2} \tag{5.4}$$

The switch then recycles to Position 1 as a new message bit X_{i+1} enters the encoder, and the process repeats. The result is a sequence of three output bits for every input bit. Hence, this is a rate 1/3 encoder.

By examining Equations 5.2, 5.3, and 5.4, we see that we could create a sequence which would tell us, for each switch position, which inputs were used to create a given output. We call these "generator sequences." For the three switch positions of the encoder in Figure 5.4 these sequences are:

$$\begin{aligned} g_1 &= (1\ 1\ 0) \\ g_2 &= (1\ 0\ 1) \\ g_3 &= (1\ 1\ 1) \end{aligned} \tag{5.5}$$

For example, the sequence g_2 tells us that Y_2 is formed by adding the present input to the input delayed two time periods, but not the input delayed once. The sequence out of one of the summers (switch positions), say the first one, consists of every third code bit. If we call this sequence Y_1, we can write:

$$Y_1 = X * g_1 \tag{5.6}$$

where " * " stands for discrete convolution. That is, the Y_1 output sequence is the convolution of the input sequence and the generator sequence g_1. It is this fact that gives us the name "convolutional code." Note that this result allows us to consider g_1 as the (discrete) impulse response of a (finite impulse response) digital filter with input X. As an example, if the encoder were initialized to zero, and the input was:

$$X = (1\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ .\ .\ .)$$

the output Y_1 would be:

$$Y_1 = (\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ .\ .\ .\ ) * (\ 1\ 0\ 1\ )$$

$$= (\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ .\ .\ .\ ) \qquad (5.7)$$

In the final composite output sequence Y, the values of the subsequences Y_1, Y_2, and Y_3 are interwoven.

Note that in the above example each of the three output values is affected by three inputs, the present input and each of the two immediate past inputs. Thus any input message bit affects nine code bits. The code is said to have a "constraint length" of nine. In general the number of delay or memory devices is called m, the number of outputs (switch positions) is called v, and the constraint length is:

$$N = (m+1)v \qquad (5.8)$$

Thus the encoder shown in Figure 5.4 is a rate 1/3 convolutional encoder with two delay units and a constraint length of nine.

The study of convolutional encoding and decoding is greatly facilitated by the introduction of two types of diagrams which describe the operation of the encoder. The first is a "state diagram" which indicates the various states which the encoder can be in, and the transitions to new states whichare possible. The second, called a "trellis diagram", is essentially a state diagram spread out over time. It indicates the various combinations of states which the encoder can have.

The state diagram of the encoder in Figure 5.4 is given in Figure 5.5. The "state" of the encoder, at any instant in time, is defined as the outputs of the two delay elements (X_{i-1}, X_{i-2}). Since there are two binary values for these outputs, there are four possible states (00, 01, 10, 11). These are shown in the four boxes in Figure 5.5.

The eight curved lines indicate changes from one state to the next when a new input X_i enters the encoder. Associated with each transition line is a pair of numbers separated by a slash mark, such as, for example:

1/010

This means that the input was a 1 and the output sequence was 010. In general this pair of numbers can be written as:

$$X_i/Y_1Y_2Y_3$$

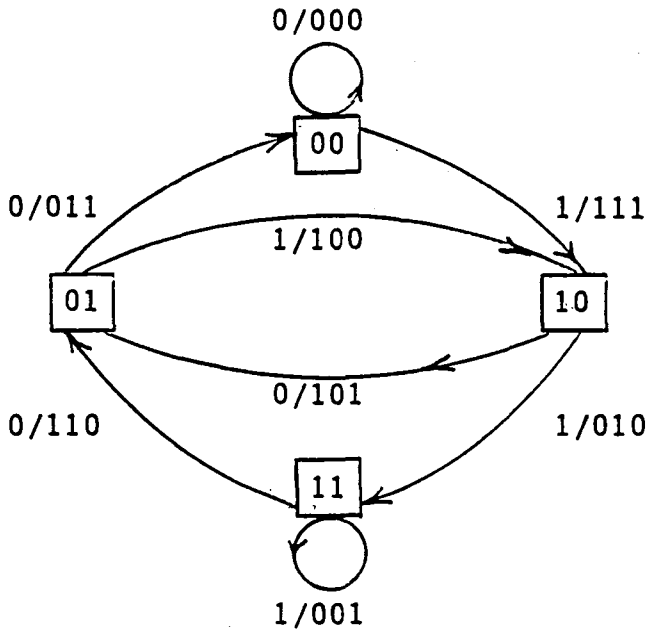


Figure 5.5 Encoder State Diagram

Note that there are only eight transitions shown, since some transitions are not possible. For example, the encoder cannot go from state 01 to state 11 since the 0 in 01 is necessarily shifted to the righthand position when the new input comes in, and hence the new state can only be 10 or 00.

Suppose that a message sequence enters the encoder. We show next a trellis diagram which indicates all of the possible paths which input sequences could follow. Figure 5.6 shows a trellis diagram for the encoder of Figure 5.4 where the case where its input strings have length 4.

We assume that the encoder starts and ends in the 00 state. That is, after four input values, the input is assumed to be 0 for two more clocking periods so that the encoder must return to the 00 state.

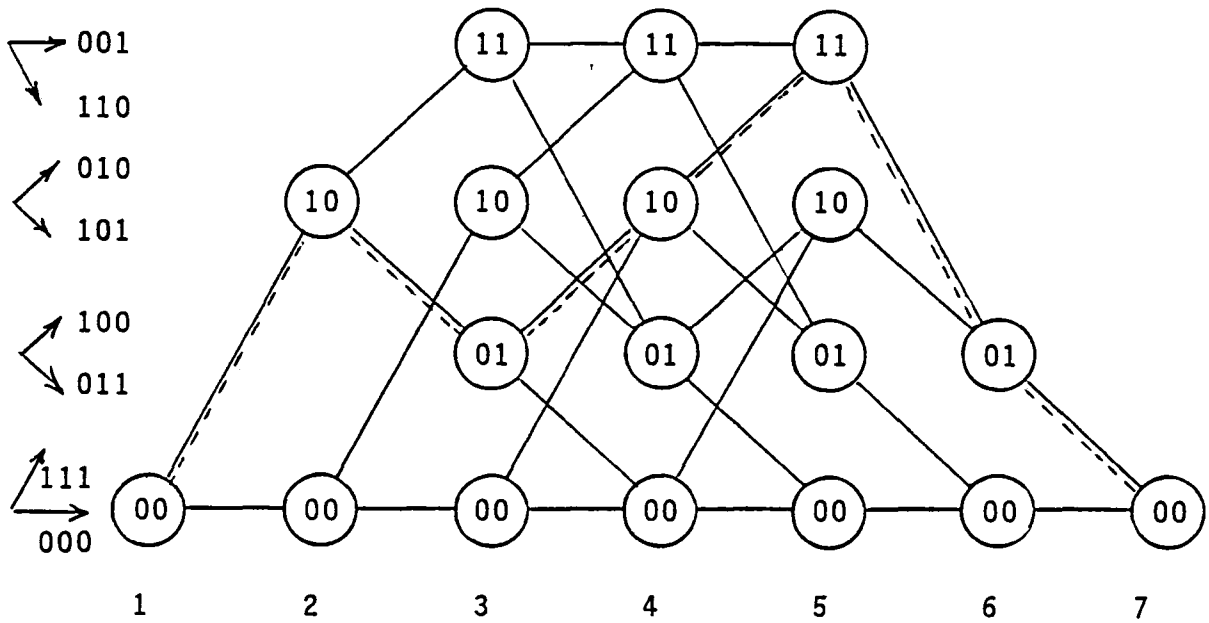


Figure 5.6 Encoder Trellis Diagram

Each of the boxes in the trellis diagram represents the state of the encoder, as before. At the time of each message bit clock pulse, represented by an integer on the time axis, the encoder moves to its new state. If the input is a 1, the transition is indicated by the upper of the two paths leaving the state. If the input is a 0, the lower of the two paths is followed. Thus all the possible sequences of states are represented in the trellis diagram by all the possible paths through the trellis. As an example the path corresponding to:

$$X = 1\ 0\ 1\ 1$$

is represented by the dashed line as shown.

Let us now summarize the action of the encoder. The input sequence X is of length L, the number of delays is m, and the number of output values for each input message bit is v. The resulting total number of binary signals sent by the transmitter is:

$$n = (L+m)v \qquad (5.9)$$

In the example above, this was:

$$n = (4+2)3 = 18 \qquad (5.10)$$

Thus the decoder receives n signals which it uses to estimate which L binary message bits were sent. Note that while the transmitted signals are binary, the received signals are corrupted by noise, and are hence analog. Figure 5.7 suggests a possible set of five received signals.

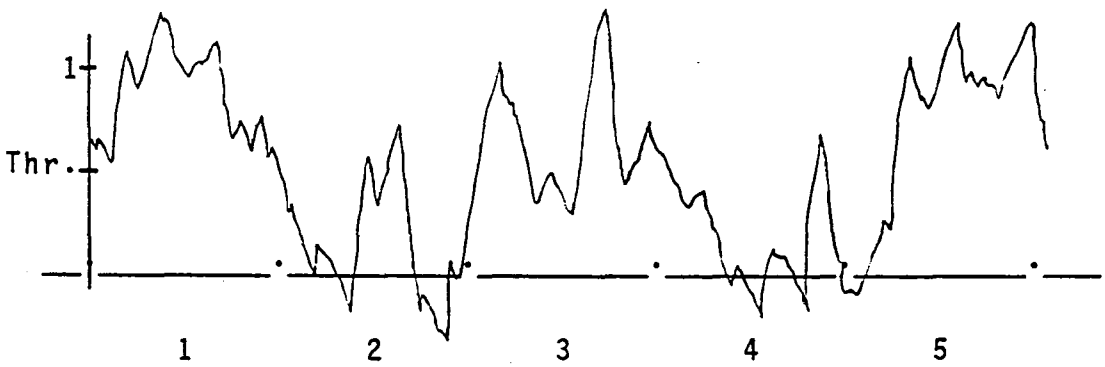


Figure 5.7 Typical Received Binary Signal with Noise

The encoder must decide whether the received signals are 0 or 1. It is quite possible that the noise may have corrupted the signal so badly so as to deceive the decoder into incorrectly estimating a given bit value. The purpose of the encoding process is to correct any such errors as may occur. One way in which the decoder can operate is to make a binary decision on each bit as it arrives. It might do this by deciding whether the average value of the signal over the bit duration was above or below some threshold. In the case of the signal of Figure 5.7, the original sequence appears to have been 10101. A decoder which makes binary (0 or 1) decisions about each bit as it occurs is called a "hard decoder."

But let us take another look at the signal in Figure 5.7. We might well have a good deal of confidence in our decision about elements 1,2,4, and 5, since there doesn't appear to be much question that these signals spend most of their time near either 0 or 1. But element 3 is much less certain. It might well be a 0 with a lot of noise. If the decoder makes hard decisions, and it makes a mistake on number 3, then it makes 1 error in 5 in this case, not an impressive record. But notice that in making hard decisions, we throw away information, namely the question of how close we are to the threshold, that is, how confident we are about our decision. Suppose that instead of making hard decisions we quantize the received level into more that two levels, with the quantization giving information about how close

we are to the threshold. If the signal elements are sent as a coded block, and if an error has been made in one of the elements, the receiver fails to find the received block in its codebook (assuming of course that the Hamming distance is more than one). The decoder can then go back and consider the quantization levels, and change its decision into that codeword which is most likely over the quantization values. Such a device is called a "soft decoder." Its performance is about 2 dB better than that of a hard decoder. That is, the signal power could be reduced by about 2 dB without decreasing the final bit error probability. Soft decision decoding can be used on convolutional decoders if the cost of the additional complexity is acceptable.

This completes our discussion of convolutional encoding. We turn next to the more challenging problem of decoding. Three major approaches to decoding have developed over the years. They are: Viterbi decoding, sequential decoding, and threshold decoding. We shall consider each in turn.

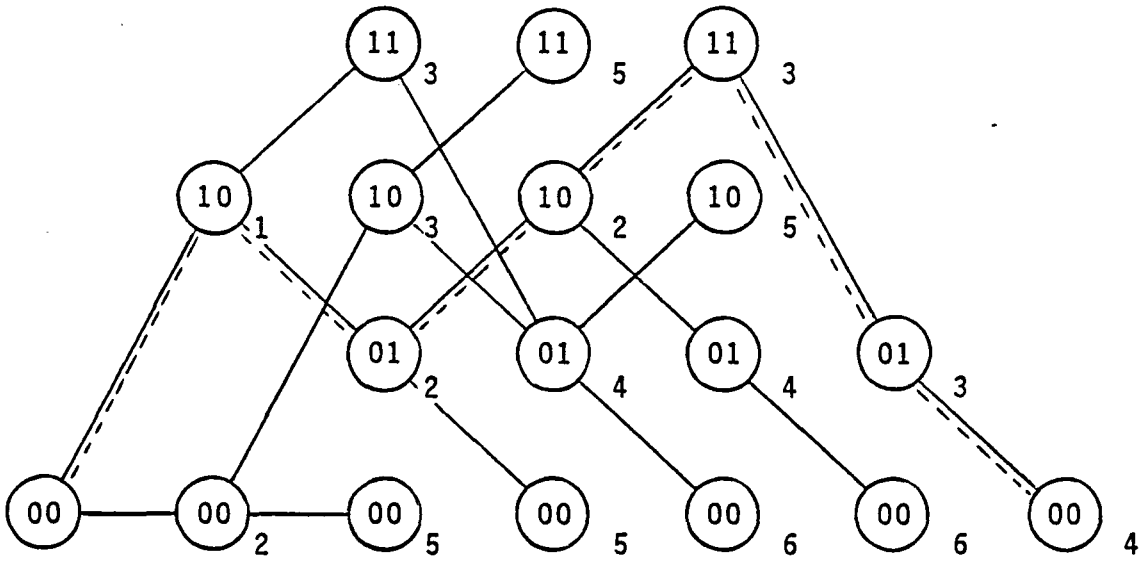
Viterbi Decoding

Refer again to the trellis diagram of Figure 5.6. The Viterbi decoding algorithm finds the one path through the trellis which is closest to the received signal in Hamming distance. That is, it finds the codeword in the book of possible codewords (trellis path) which is most like the received signal. This is called the "maximum likelihood path."

We start in the lower lefthand corner of the trellis diagram, redrawn in Figure 5.8, with some new information. At the bottom of the diagram the original message, labeled "t", is given. Below it is the received signal, called "r". Note that there are four bit errors in r.

The initial state is of course 00. If the first message bit is a one, the encoder produces the string 111, as shown. If a zero, the string is 000. The decoder does not see a clean binary signal, but rather a noisy signal as in Figure 5.7. It must estimate what was actually sent. Let us assume for the time that it makes hard decisions. It then produces an estimate of the three-bit string, perhaps it estimates 110.

The first step in the algorithm is to compute the Hamming distance between the received signal and the signal corresponding to each path element. For example, suppose that 110 was estimated by the decoder. In the case of the upper path



t	111	101	100	010	110	011
r	101	111	100	110	110	001

Figure 5.8 Decoder Trellis Diagram - Hard Decision Decoding

leaves the state in the lower lefthand corner, the distance between 111 and 110 is 1. For the lower (horizontal) path the distance between 000 and 110 is 2. For each state in the trellis we store the path closest to the received signal (this path is called the "survivor"). The other path can be eliminated. We also store the Hamming distance (called the "metric").

The second step in the algorithm is to select the complete path with the smallest cumulative metric or Hamming distance. An example is shown in Figure 5.8. Before we start through this example, we make some general observations which apply to all trellisses and are quite important. As noted above we always start in the lower lefthand corner because we know that the encoder was initialized to the 00 state. Furthermore we know that we must always end up in the 00 state because the last two inputs to the decoder are 0, which forces the decoder to end in the 00 state. We also know from this that in going from state 5 to 6, and 6 to 7 the path can only be the lower path, corresponding to the 0 input. Thus, no upper paths are seen on

the right end of the trellis.

In the example shown in Figure 5.8 the first state was, of course, 00, and the first message bit was a 1, and so the encoder generated the string 111 (see Figure 5.5). But the receiver made a bit error and estimated 101. Since the Hamming Distance between 101 and 000 is 2, the metric at the 00 state at position 2 is 2, as indicated to the lower right of the state circle. The metric at the 10 state is 1. We proceed through the entire trellis in this manner, eliminating any paths which generate a higher metric than that indicated at any given state node. We do not lose anything by dropping any path element which causes a higher metric than that shown since such an element could not be a part of an overall path which had a smaller cumulative metric than the chosen path.

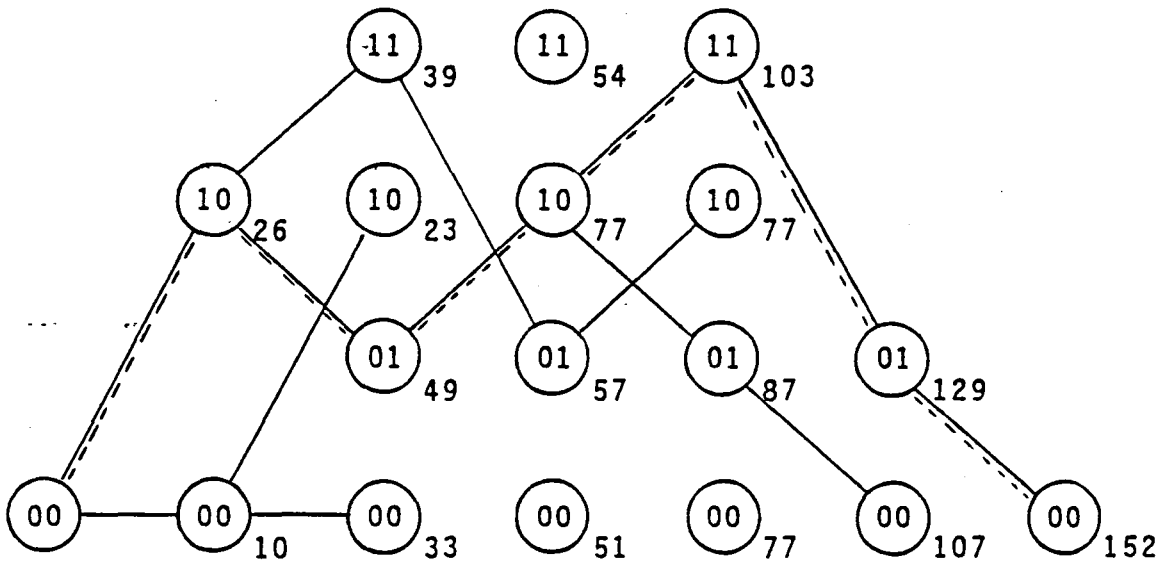
After the trellis has been completely traversed, the decoder selects the path which has the lowest metric, and deduces the signal which was probably sent. In the case shown the path with the lowest metric has the structure "up-down-up-up-down-down." The last two are ignored since they are always down-down (00). The resulting estimated message is 1011, which is correct. The decoder has succeeded in correcting 4 bit errors. Selection of the path with the lowest metric does not assure us that the estimate is correct, only that it is the most likely message given the receiver input.

So far the discussion has been limited to hard decisions, suggested by the binary received signal r in Figure 5.8. If we wish to take advantage of the additional gain of about 2 dB due to quantization, we design the receiver to quantize the incoming data. Two major questions arise. How many quantization levels are desired, and what should re the nature or law of the quantization? To retain a large amount of information, the number of quantization levels should be large. However, this would require a very complex decoder. As few as four levels can increase the effectiveness of the receiver very significantly. More levels are often not worth the cost. The nature of the quantization is often logarithmic, but little is lost by using integers which approximate a logarithmic law. An example of an integer quantization metric law is given in Figure 5.9.

		r			
		0	0	1	1
y	0	10	8	5	0
	1	0	5	8	10

Figure 5.9 Metric Values for Quantization Data

The table is used in the following way. The y value is the value which we hypothesize was transmitted in order to follow one of the trellis paths. Hence, if we are in the lower left hand corner, and want to obtain the metric for the upper path, we hypothesize that the y-string 111 was sent. The r values represent the quantized data. We use underlined values to indicate that the data was above the high threshold, and hence we have high confidence in this data. For example, if we hypothesize that a 111 was sent, and in fact a $\underline{1}01$ was received, the total metric is 10 + 5 + 8 = 23. We now repeat the trellis diagram of Figure 5.8, this time using quantized data. A set of hypothetical data is shown along with the resulting survivor/metric solution.



111	101	100	010	110	011
$\underline{1}11$	$0\underline{0}1$	$\underline{10}0$	$0\underline{1}0$	$\underline{1}10$	$00\underline{1}$

Figure 5.10 Decoder Trellis Diagram - Soft Decisions

The primary disadvantage of coding schemes in general is that implementation is more complex than for uncoded systems. Sometimes the cost of this added complexity can be justified, sometimes it cannot. In the case of the Viterbi algorithm the decoder must remember 2^m L-bit survivors and their metrics, and, for each message bit, 2^m calculations must be made to determine the dnew metric and the survivor. If m is large, implementation

can be very costly. In practice, m is limited to about 10.

The value of a coder is often expressed in terms of its "coding gain" which is the amount by which the signal to noise ratio can be reduced and still retain the same error probability as without coding. Practical hard decision Viterbi decodersachieve a coding gain of about 4 - 5 dB. Soft decision decoding adds another 2 dB or so to this.

Sequential Detection

The most serious problem with the Viterbi decoding algorithm is the 2^m calculations and memory storages which must be carried out for each message bit. Sequential decoding uses an alternate approach which does not have this problem. In this section we briefly discuss the sequential detection approach, and its advantages. A detailed discussion of the the process, and the various associated algorithms is left to the literature.

Consider again the trellis diagram of Figure 5.6. The diagram is relevant to sequential decoding as well as Viterbi decoding, but the algorithm is different. In this case, the decoder receives an input and initiates a path search, starting from the lower left hand corner. Each time the decoder reaches a node, it computes a metric based on the received data and the path followed so far. If this metric exceeds a certain metric, the encoder continues to search forward from the given node. If the metric is below the threshold, the decoder backs up and searches along an alternate path. This continues until a net path length of L + m is reached. If the metric is still above the set threshold, the path is declared to be correct. If there is little noise, we can expect the decoder to trace out the correct path with little or no backtracking, and thus fairly rapidly. If, however, there is a great deal of noise, the decoder may have to backtrack often, leading to a long search time. Usually the expected number of calculations required for sequential decoding is much less than for Viterbi decoding. However, if we have some "bad luck", or if the system becomes quite noisy, we may search for such a long time that the search must be aborted, with a consequent "data erasure".

The principle advantage of sequential decoding is that it permits relatively large values of memory length m, with manageable complexity.

Threshold Decoding

The primary advantage of threshold decoding is that it can be implemented with relatively simple threshold logic gates. It has the disadvantage that it is suboptimum compared with Viterbi and sequential decoding, but the advantage of simple electronics often justifies this advantage. In this section we only very

briefly discuss this implementation, again leaving the details to the literature.

The major difference between threshold decoding and both Viterbi and sequential decoding is that the decoding does not involve a search process. Of the v sequences created by the encoder, one is the actual message sequence itself (hence the code is systematic), and the other v-1 sequences are parity check sequences. In this sense the coding is somewhat like that of block codes. The decoder calculates a "syndrome" on the basis of the parity bits, which gives evidence of errors. If the syndrome calculation produces a value above some set threshold, the decoder modifies the data sequence in accordance with the error suggested by the syndrome.

6. APPLICATIONS OF CODING

The purpose of this section is to briefly discuss the application of coding in particular situations. We begin with some general comments on where coding can be effectively applied, and then review a few actual examples. No attempt is made here to present an exhaustive survey of coding applications.

6.1 General Considerations

Channel coding is used in situations where errors are made, and it is considered important or necessary to detect and/or correct these errors. However, coding is not necessarily the desirable approach in all situations in which errors are made. As a general rule, coding tends to be effective in low signal to noise ratio (SNR) environments, where the channel is power limited. A classic example is the space-to-earth channel in which the power available in the space-borne transmitter is limited, often by the launch weight of the solar panels. It is not surprising that some of the first applications of error correction coding were in space channels, beginning in the late 1960's. The space channel is usually quite accurately modeled as having additive white Gaussian Noise (AWGN). Such channels tend to make individual independent errors, which are particularly susceptible to correction by convolutional codes.

Coding does not tend to be effective in high SNR environments, such as on telephone and other hardwire lines, which are typically bandwidth limited. In addition, channels such as switched telephone lines are not characterized by Gaussian noise, but rather tend to be dominated by impulse noise, due primarily to mechanical switching. This noise has a relatively long duration (perhaps a few milliseconds) per burst, and hence can last far longer than one message bit. Hence bit errors are not longer individual and independent. Coding tends to be less effective against bursts of errors than single errors, although some codes are designed to handle bursts up to some maximum length. Because of the high SNR and the presence of noise bursts on telephone lines, coding has not often been applied in this environment. However, with the evolution of telephone switching from mechanical to electronic devices, coding may begin to appear in some telephone line applications, particularly those dedicated to digital data.

As we saw earlier, feedback (ACK/NACK) error control tends to be used in situations where the transmission time is short, such as on earth-based hardwire lines. Forward error correction tends to be used in situations where the transmission time is so long as to make feedback delays unacceptable.

6.2 Examples of Applications

In this section we point out a few specific cases in which coding has been used or is planned.

One of the earliest applications of channel coding was to NASA's Pioneer deep space probes, which used convolutional codes with sequential decoding to combat the effects of noise on the very weak signals returned from space (Forney(1970)). Subsequent space communication systems have benefited from the application of both sequential and Viterbi decoding to convolutional encoded signals. Continuing in this tradition, the Tracking and Data Relay Satellite System (TDRSS) in one application will use a rate 1/2, constraint length 7, non-systematic, convolutional code (Tracking(1980)). In this case the coding will be on user vehicles which transmit to the TDRSS satellite receiver. The coding permits the user to radiate less power, and reduces the interference to any other user which may be in the receiver's antenna beam. The Galileo probe will use Viterbi decoding concatenated with Reed-Solomon coding designed to correct for bursts of errors which tend to be self-generated by Viterbi decoders under some circumstances.

Not all applications of coding are convolutional codes. The Joint Tactical Information Distribution System (JTIDS) has adopted the Reed-Solomon (31, 15) code, which is particularly effective against bursts of errors. A (127,50) BCH code has been proposed for use in the "Space Operations Center" (Space (1982)).

Block codes have been applied in connection with computers and their associated memories. Single bit error correcting codes have been used for more than 25 years on computer systems such as the IBM 650, Whirlwind, LARC, and many more (Wakerly(1982)).

7. CONCLUSION AND RECOMMENDATIONS

The purpose of this document has been to study the use of coding in the space station data system, and to recommend additional studies which are necessary and should be carried out in the next one to three years to define more accurately how coding can be used most effectively in the space station environment.

As a result of this study, five specific recommendations are made.

1. A survey of existing applications of coding, and of the development of new hardware and software for coding should be carried out.

2. As the data needs of the space station are further defined, system studies should include conisderation of the ways in which coding can improve system performance.

3. A study should be made of the possibility of using coding to obtain channel separation in the multiple user environment.

4. Some attention should be paid to the question of whether coding might be useful for fault tolerance or for synchronization purposes.

5. Coding should be incorporated in simulation studies of the space station data system.

REFERENCES

Ahlswede, R., "Multi-Way Communication Channels," in Proc. 2nd Int. Symp. Inf. Th.(Tsahkadsor, Armenian S.S.R.), pp.23-52, 1971 (Publishing House of the Hungarian Academy of Sciences, 1973)

Berlekamp, E.R., "The Technology of Error-Correcting Codes", IEEE Proc., Vol.68, No.5, May, 1980, pp.564-593

Chang, S., and J.K.Wolf, "On the T-User M-Frequency Noiseless Multiple-Access Channel with and without Intensity Information", IEEE Trans.Inf.Th., Vol.IT-27, No.1, pp.41-48, Jan., 1981

Cohen, A.R., J.A.Heller, and A.J.Viterbi, "A New Coding Technique for Asynchronous Multiple Access Communication," IEEE Trans. Comm. Tech., Vol. Com.-19, No. 5, pp.849-855, Oct., 1971

El Gamal, A., and T.M.Cover, "Multiple User Information Theory," Proc. IEEE, Vol. 68, No. 12, pp. 1466-1483, Dec., 1980

Farrell, P.G., "Survey of Channel Coding for Multi-User Systems," in New Concepts in Multi-User Communication, Sijthoff and Noordhoff, Rockville, MD., 1981

Forney, G.D., "Coding and Its Application in Space Communication", IEEE Spectrum, June, 1970, PP.47-58

Goodman, D.J., P.S.Henry, and V.K.Prabhu, "Frequency-Hopped Multilevel FSK for Mobile Radio", BSTJ, Vol.59, No.7, pp.1257 - 1275, Sept. 1980

Gyorfi, L., and I. Kerekes, "A Block Code for Noiseless Asynchronous Multiple Access OR Channel," IEEE Trans. Inf. Th., Vol. IT-27, No. 6, pp. 788-791, Nov., 1981

Healy, T.J., "The Capacity Region of a Spread Spectrum Multiple Access Channel", 1982 IEEE Military Communication Conference, Boston, Oct.17-20,1982

Lin, S., and D.J. Costello, "Coding for Reliable Data Transmission and Storage", in Protocols and Techniques for Data Communication Networks, Prentice-Hall, Englewood Cliffs, N.J., 1981

Matsumoto, M., and G.R.Cooper, "Multiple Narrow-Band Interferers in an FH-DPSK Spread-Spectrum Communication System", IEEE Trans. Veh. Tech., Vol.VT-30, No.1, pp.37-42, Feb., 1981

McEliece, R.J., The Theory of Information and Coding, Addison Wesley Publ. Co., Reading, Mass., 1977

Scales, W.C., "Potential Use of Spread Spectrum Techniques in Non-Government Applications," NTIS: FCC-0320, Dec., 1980

Shannon, C. E., "A Mathematical Theory of Communication", BSTJ, Vol.27, 1948

Space Operations Center Communication and Tracking System, preliminary concept design, EE8-4/82-047, NASA, Johnson Space Center, Houston, Texas

Timor, U., "Multistage Decoding of Frequency-Hopped FSK System", BSTJ, Vol.60, No.4, pp.471-483, April, 1981

Tracking and Data Relay Satellite System(TDRSS) Users' Guide, Revision 4, STDN No. 101.2, NASA, Goddard Space Flight Center, Greenbelt, Md., Jan., 1980

Van Der Meulen, E.C., " A Survey of Multi-Way Channels in Information Theory: 1961-1976," IEEE Trans. Inf. Th., Vol.IT-23, No.1, pp.1-37, Jan., 1977

Wakerly, J., Error Detecting Codes, Self-Checking Circuits and Applications, North-Holland, New York, 1978

Wiggert, D., Error-Control Coding and Applications, Artech House, Dedham, Mass., 1978

Wolf, J.K., "Coding Techniques for Multiple Access Communication Channels," in New Concepts in Multi-User Communication, Sijthoff and Noordhoff, Rockville, MD., 1981

Yue, O., "Frequency-Hopping, Multiple-Access, Phase-Shift-Keying System Performance in a Rayleigh Fading Environment", BSTJ, Vol. 59, No.6, pp.861-879, July-Aug., 1980

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-166402		

4. Title and Subtitle	5. Report Date
CHANNEL CODING IN THE SPACE STATION DATA SYSTEM NETWORK	May 1982
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
T. Healy	
9. Performing Organization Name and Address	10. Work Unit No.
University of Santa Clara Department of Electrical Engineering Santa Clara, CA 95053	K-1637
	11. Contract or Grant No.
	NCC2-128
12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
National Aeronautics and Space Administration Washington, D.C. 20546	Contractor Report
	14. Sponsoring Agency Code
	506-61-01

15. Supplementary Notes

Point of Contact: Technical Monitor, Mr. Terry L. Grant, MS 244-7, Ames Research Center, Project Technology Branch, Moffett Field, CA 94035

16. Abstract

Channel coding, in one form or another, is an established and common element in data systems today. No analysis and design of a major new system would fail to consider ways in which channel coding could make the system more effective. The presence of channel coding on TDRS, Shuttle, the Advanced Communication Technology Satellite Program system, the JSC-proposed Space Operations Center, and the proposed 30/20 GHz Satellite Communication System strongly support the requirement for the utilization of coding for the communications channel. The designers of the Space Station data system will have to consider the use of channel coding. A detailed discussion of the use of channel coding for error correction, privacy/secrecy, channel separation, and synchronization is presented.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Space Station Technology Channel Coding Data Systems Communications Systems	Unlimited Subject Category 62

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	54	

*For sale by the National Technical Information Service, Springfield, Virginia 22161

End of Document