Introduction to Error-Control Coding

Jay M. Jacobsmeyer, P.E.
Pericle Communications Company

Abstract

This paper is an introduction to the subject of error-control coding. The usual mathematical approach is bypassed to appeal to a wider audience. The operation, proper use, and limitations of error-control codes are described. Several coding techniques are investigated and the popular compact disc (CD) player is used to illustrate the successful application of error-control coding.

1.0 Introduction

Many of us have heard the terms *error-control coding* in connection with the latest commercial and military communications systems, but few of us understand what it is, what it does, and why we should be concerned about it. This tutorial describes what error-control codes do, how they work, and when they can be used successfully. The field is growing rapidly and a working knowledge of the topic will soon become a prerequisite for anyone involved in the design of modern communications systems.

Error-control coding is a discipline under the branch of applied mathematics called Information Theory, discovered by Claude Shannon in 1948 [6]. Prior to this discovery, conventional wisdom said that channel noise prevented error-free communications. Shannon proved otherwise when he showed that channel noise limits the transmission rate, not the error probability. Shannon showed that every communications channel has a capacity, C (measured in bits per second), and as long as the transmission rate, R (also in bits per second), is less than C, it is possible to design a virtually error-free communications system using error-control codes. Shannon's contribution was to prove the existence of such codes. He did not tell us how to find them.

After the publication of Shannon's famous paper [6], researchers scrambled to find codes that would produce the very small probability of error that he predicted. Progress was disappointing in the 1950s when only a few weak codes were found. In the 1960s, the field split between the algebraists who concentrated on a class of codes called *block codes* and the probabilists, who were concerned with understanding encoding and decoding as a random process. The probabilists eventually discovered a second class of codes, called *convolutional codes*, and designed powerful decoders for them. In the 1970s, the two research paths merged and several efficient decoding algorithms were developed. With the advent of cheap microelectronics, decoders finally became practical and in 1981, the entertainment industry

adopted a very powerful error control scheme for the new compact disc (CD) players [9]. Today, error-control coding in its many forms is used in almost every new military communications system including the Joint Tactical Information Distribution System (JTIDS) and the Jam Resistant Secure Communications (JRSC) system employed on the Defense Satellite Communications System (DSCS).

The purpose of this paper is to bring the basics of error-control coding within reach of the communications professional. It is merely an introduction to the subject and further study is required before one should attempt to design error-control systems. However, if this paper does no more than convince the reader of the merits of error-control coding, it will have served its purpose.

2.0 Preliminaries

In this section we define the terms that we will need to handle the later topics. Definitions are tailored to suit this paper and may differ from those presented in the literature.

Digital communications systems are often partitioned as shown in Figure 1. The following paragraphs describe the elements of Figure 1 and define other terms common to error-control coding.

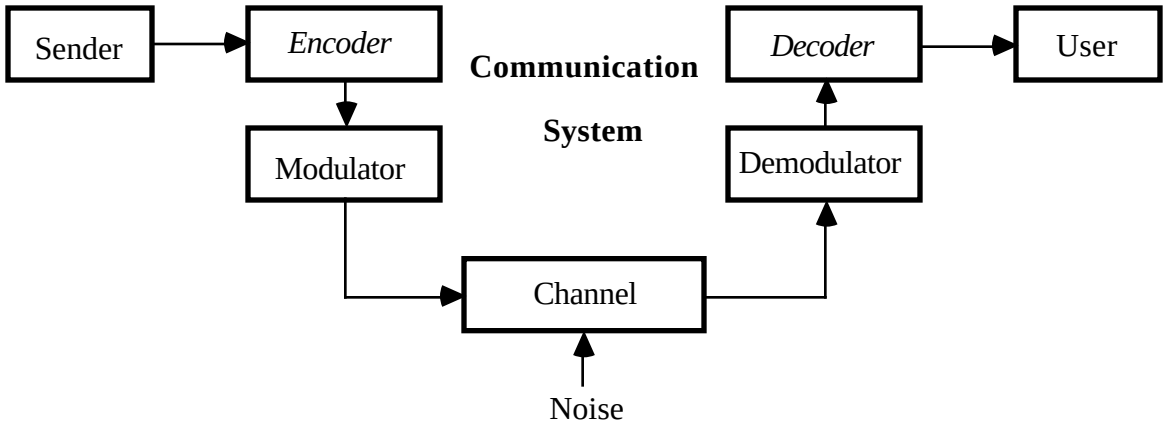

Figure 1 - The Digital Communications System

Encoder and Decoder - The encoder adds redundant bits to the sender's bit stream to create a *codeword*. The decoder uses the redundant bits to detect and/or correct as many bit errors as the particular error-control code will allow.

Modulator and Demodulator - The modulator transforms the output of the encoder, which is digital, into a format suitable for the channel, which is usually analog (e.g., a telephone channel). The demodulator attempts to recover the correct channel symbol in the presence of

noise. When the wrong symbol is selected, the decoder tries to correct any errors that result.

Communications Channel - The part of the communication system that introduces errors. The channel can be radio, twisted wire pair, coaxial cable, fiber optic cable, magnetic tape, optical discs, or any other noisy medium.

Error-Control Code – A set of codewords used with an encoder and decoder to detect errors, correct errors, or both detect and correct errors.

Bit-Error-Rate (BER) - The probability of bit error. This is often the figure of merit for a error-control code. We want to keep this number small, typically less than 10^{-4}. Bit-error-rate is a useful indicator of system performance on an independent error channel, but it has little meaning on bursty, or dependent error channels.

Message-Error-Rate (MER) - The probability of message error. This may be a more appropriate figure of merit because the smart operator wants all of his messages error-free and could care less about the BER.

Undetected Message Error Rate (UMER) - The probability that the error detection decoder fails and an errored message (codeword) slips through undetected. This event happens when the error pattern introduced by the channel is such that the transmitted codeword is converted into another valid codeword. The decoder can't tell the difference and must conclude that the message is error-free. Practical error detection codes ensure that the UMER is very small, often less than 10^{-16}.

Random Errors - Errors that occur independently. This type of error occurs on channels that are impaired solely by thermal (Gaussian) noise. Independent-error channels are also called *memoryless* channels because knowledge of previous channel symbols adds nothing to our knowledge of the current channel symbol.

Burst Errors - Errors that are not independent. For example, channels with deep fades experience errors that occur in bursts. Because the fades make consecutive bits more likely to be in error, the errors are usually considered dependent rather than independent. In contrast to independent-error channels, burst-error channels have *memory*.

Energy Per Bit - The amount of energy contained in one information bit. This is not a parameter that can be measured by a meter, but it can be derived from other known parameters. Energy per bit (E_b) is important because almost all channel impairments can be overcome by increasing the energy per bit. Energy per bit (in joules) is related to transmitter power P_t (in watts), and bit rate R (in bits per second), in the following way:

$$E_b = \frac{P_t}{R} \tag{1}$$

If transmit power is fixed, the energy per bit can be increased by lowering the bit rate. Thus, the reason why lower bit rates are considered more robust. The required energy per bit to maintain reliable communications can be decreased through error-control coding as we shall see in the next section.

Coding Gain - The difference (in dB) in the required signal-to-noise ratio to maintain reliable communications after coding is employed. Signal-to-noise ratio is usually given by E_b/N_0, where N_0 is the noise power spectral density measured in watts/Hertz (joules). For example, if a communications system requires a E_b/N_0 of 12 dB to maintain a BER of 10^{-5}, but after coding it requires only 9 dB to maintain the same BER, then the coding gain is 12 dB minus 9 dB = 3 dB. (Recall that dB (decibels) = 10 $\log_{10} X$, where X is a ratio of powers or energies.)

Code Rate - Consider an encoder that takes k information bits and adds r redundant bits (also called *parity* bits) for a total of $n = k + r$ bits per codeword. The code rate is the fraction k/n and the code is called a (n, k) error-control code. The added parity bits are a burden (i.e. overhead) to the communications system, so the system designer often chooses a code for its ability to achieve high coding gain with few parity bits.

3.0 What Coding Can and Cannot Do

The traditional role for error-control coding was to make a troublesome channel acceptable by lowering the frequency of error events. The error events could be bit errors, message errors, or undetected errors. Coding's role has expanded tremendously and today coding can do the following:

Reduce the occurrence of undetected errors. This was one of the first uses of error-control coding. Today's error detection codes are so effective that the occurrence of undetected errors is, for all practical purposes, eliminated.

Reduce the cost of communications systems. Transmitter power is expensive, especially on satellite transponders. Coding can reduce the satellite's power needs because messages received at close to the thermal noise level can still be recovered correctly.

Overcome Jamming. Error-control coding is one of the most effective techniques for reducing the effects of the enemy's jamming. In the presence of pulse jamming, for example, coding can achieve coding gains of over 35 dB [8].

Eliminate Interference. As the electromagnetic spectrum becomes more crowded with man-made signals, error-control coding will mitigate the effects of unintentional interference.

Despite all the new uses of error-control coding, there are limits to what coding can do. On the Gaussian noise channel, for example, Shannon's capacity formula sets a lower limit on the signal-to-noise ratio that we must achieve to maintain reliable communications. Shannons's

lower limit depends on whether the channel is power-limited or bandwidth-limited. The deep space channel is an example of a power-limited channel because bandwidth is an abundant resource compared to transmitter power. Telephone channels, on the other hand, are considered bandwidth-limited because the telephone company adheres to a strict 3.1 kHz channel bandwidth.

For strictly power-limited (unlimited bandwidth) channels, Shannon's lower bound on E_b/N_0 is 0.69, or –1.6 dB [3]. In other words, we must maintain an E_b/N_0 of at least -1.6 dB to ensure reliable communications, no matter how powerful an error-control code we use.

For bandwidth-limited channels with Gaussian noise, Shannon's capacity formula can be written as the following [3]:

$$\frac{E_b}{N_0} \geq \frac{2^r - 1}{r} \tag{2}$$

where r is the spectral bit rate in bits/s/Hz. For example, consider a bandwidth-limited channel operating with uncoded quadrature phase shift keying (a common modulation technique with 2 bits/symbol and a maximum spectral bit rate of r = 2 bit/s/Hz) and a required BER of 10^{-5}. We know that without coding, this communications system requires an E_b/N_0 of 9.6 dB [5]. Shannon's formula above says that to maintain reliable communications at an arbitrarily low BER, we must maintain (for r = 2 bits/s/Hz) an E_b/N_0 of at least 1.5 (1.8 dB). Therefore, if we need to lower the required E_b/N_0 by more than 7.8 dB, *coding can't do it*. We must resort to other measures, like increasing transmitter power. In practice, the situation is worse because we have no practical code that achieves Shannon's lower bound. A more realistic coding gain for this example is 3 dB rather than 7.8 dB.

Another limitation to the performance of error-control codes is the modulation technique of the communication system. Coding must go hand-in-hand with the choice of modulation technique for the channel. Even the most powerful codes cannot overcome the consequences of a poor modulation choice.

4.0 How Error-Control Codes Work

A full understanding of the structure and performance of error-control codes requires a foundation in modern algebra and probability theory, which is beyond the scope of this paper. Instead, we appeal to the reader's intuition and common sense. Let's begin by showing how the encoder and decoder work for binary block codes.

The block encoder takes a block of k bits and replaces it with a n-bit codeword (n is bigger than k). For a binary code, there are 2^k possible codewords in the *codebook*. The channel introduces errors and the received word can be any one of 2^n n-bit words of which only 2^k are valid codewords. The job of the decoder is to find the codeword that is closest to the received n-bit word. How a practical decoder does this is beyond the scope of this paper, but

our examples will use a brute force look-up table method. The decoding spheres of Figure 2 will be used to illustrate the decoding process. In Figure 2, each valid codeword is represented by a point surrounded by a sphere of radius *t* where *t* is the number of errors that the code can correct [3]. Note that codewords *A* and *B* of Figure 2 are separated by a distance d_{min}, called the *minimum distance* of the code. The minimum distance of a code is defined as the smallest number of places that any two codewords in the codebook differ. Usually, codes with large minimum distance are preferred because they can detect and correct more errors. First let's consider a decoder that can only detect errors, not correct them.

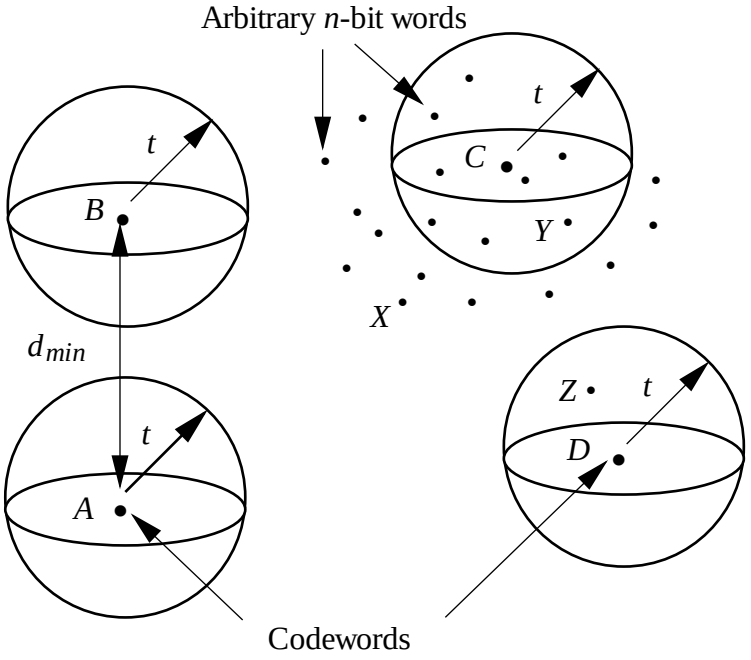

Figure 2 - Decoding Spheres of Radius *t*

A. Error Detection Only

The minimum distance of a code gives a measure of its error detection capability. An error control code can be used to detect all patterns of *u* errors in any codeword as long as $d_{min} = u + 1$. The code may also detect many error patterns with more than *u* errors, but it is guaranteed to detect *all* patterns of *u* errors or less. We'll assume that the error detection decoder comprises a look-up table with all 2^k valid codewords stored. When an *n*-bit word is received by the decoder, it checks the look-up table and if this word is one of the allowable codewords, it flags the *n*-bit word as error-free and sends the corresponding information bits to the user. We'll use Figure 2 to illustrate three cases: no errors, a detectable error pattern, and an undetectable error pattern.

Case # 1: No errors. Let's assume that the encoder sends codeword *C* (see Figure 2) and the channel introduces no errors. Then codeword *C* will also be received, the decoder

will find it in the look-up table, and decoding will be successful.

Case #2: Detectable error pattern. This time we send codeword *C* and the channel introduces errors such that the *n*-bit word *Y* is received. Because *Y* is not a valid codeword, the decoder will not find it in the table and will therefore flag the received *n*-bit word as an errored codeword. The decoder does not necessarily know the number or location of the errors, but that's acceptable because we only asked the decoder to *detect* errors. Since the decoder properly detected an errored codeword, decoding is successful.

Case #3: Undetectable error pattern. We send codeword *C* for the third time and this time the channel introduces the unlikely (but certainly possible) error pattern that converts codeword *C* into codeword *D*. The decoder can't know that codeword *C* was sent and must assume that codeword *D* was sent instead. Because codeword *D* is a valid codeword, the decoder declares the received *n*-bit word error-free and passes the corresponding information bits on to the user. This is an example of decoder failure.

Naturally, we want the decoder to fail rarely, so we choose codes that have a small probability of undetected error. One of the most popular error detection codes is the shortened Hamming code, also known as the cyclic redundancy check (CRC). Despite its widespread use since the 1960s, the precise performance of CRCs was not known until Fujiwara et al. [7] published their results in 1985.

B. Forward Error Correction (FEC)

Comparing the spheres surrounding codewords *A* and *B* in Figure 2, we see that the error correcting capability of a code is given by $d_{min} = 2t + 1$ (this is the minimum separation that prevents overlapping spheres). Or in other words, a code with $d_{min} = 3$ can correct all patterns of 1 error, one with $d_{min} = 5$ can correct all patterns of 2 errors, and so on. A code can be used to correct *t* errors and detect *v* additional errors as long as $d_{min} \geq 2t + v + 1$. Now refer to Figure 2 and consider three error decoding cases for the error correction decoder: correct decoding, decoding failure, and error detection without correction.

Case #1: Correct decoding. Assume that codeword *C* is sent and the *n*-bit word Y is received. Because *Y* is inside *C*'s sphere, the decoder will correct all errors and error correction decoding will be successful.

Case #2: Decoding failure. This time we send codeword *C* and the channel gives us *n*-bit word *Z*. The decoder has no way of knowing that codeword *C* was sent and must decode to *D* since *Z* is in *D*'s sphere. This is an example of error correction decoder failure.

Case #3: Error detection without correction. This case shows one way that an error correction code can be used to also detect errors. We send codeword *C* and receive *n*-bit word *X*. Since *X* is not inside any sphere, we won't try to correct it. We do, however, recognize that it is an errored codeword and report this information to the user.

In the last example, we could try to correct *n*-bit word *X* to the nearest valid codeword, even though *X* was not inside any codeword's sphere. A decoder that attempts to correct all received *n*-bit words whether they are in a decoding sphere or not is called a *complete* decoder. On the other hand, a decoder that attempts to correct only *n*-bit words that lie inside a decoding sphere is called an incomplete, or *bounded distance* decoder. Bounded distance decoders are much more common than complete decoders. Now that we understand the basics of encoding and decoding, let's investigate a simple error correction code.

C. The Repetition Code

Consider a (5, 1) repetition code that repeats each bit four times. The encoder looks like this:

$0 \rightarrow 0\,0\,0\,0\,0$

$1 \rightarrow 1\,1\,1\,1\,1$

The decoder takes 5 bits at a time and counts the number of 1's. If there are three or more, the decoder selects 1 for the decoded bit. Otherwise, the decoder selects 0. The minimum distance of this code is 5, so it can correct all patterns of two errors. To compute the error performance of this code, consider a random error channel with probability of bit error, *p*. After decoding, the probability of bit error is simply the probability of three or more bit errors in a 5 bit codeword. This probability is computed for several values of *p* with results listed in Table 1.

Table 1 - Post -Decoding Probability of Bit Error For (5, 1) Repetition Code

Input BER	Output BER
10^{-2}	9.9×10^{-6}
10^{-3}	1.0×10^{-8}
10^{-4}	1.0×10^{-11}
10^{-5}	1.0×10^{-14}
10^{-6}	1.0×10^{-17}

The values listed in Table 1 show that even this simple code offers dramatic improvements in error performance, but at the price of a 400% overhead burden.

5.0 Popular Coding Techniques

In this section we highlight six of the most popular error-control coding techniques. We will discuss automatic repeat request (ARQ), forward error correction (FEC), hybrid ARQ, interleaving, erasure decoding, and concatenation.

A. Automatic Repeat Request (ARQ)

An error detection code by itself does not control errors, but it can be used to request repeated transmission of errored codewords until they are received error-free. This technique is called *automatic repeat request,* or *ARQ*. In terms of error performance, ARQ outperforms forward error correction because codewords are *always* delivered error-free (provided the error detection code doesn't fail). This advantage does not come free – we pay for it with decreased throughput. The chief advantage of ARQ is that error detection requires much simpler decoding equipment than error correction. ARQ is also adaptive since it only re-transmits information when errors occur. On the other hand, ARQ schemes require a feedback path which may not be available. They are also prone to duping by the enemy. A pulse jammer can optimize its duty cycle to increase its chances of causing one or more errors in each codeword. Ideally (from the jammer's point of view), the jammer forces the communicator to retransmit the same codeword over and over, rendering the channel useless.

There are two types of ARQ: *stop and wait* ARQ and *continuous* ARQ.

Stop-and-wait ARQ. With stop-and-wait ARQ, the transmitter sends a single codeword and waits for a positive acknowledgement (ACK) or negative acknowledgement (NAK) before sending any more codewords. The advantage of stop-and-wait ARQ is that it only requires a half duplex channel. The main disadvantage is that it wastes time waiting for ACKs, resulting in low throughput.

Continuous ARQ. Continuous ARQ requires a full duplex channel because codewords are sent continuously until a NAK is received. A NAK is handled in one of two ways: With *go back-N* ARQ, the transmitter retransmits the errored codeword plus all codewords that followed until the NAK was received. The parameter *N* is determined from the round trip channel delay. For geosynchronous satellite channels, *N* can be very large because of the 540 millisecond round trip delay. The transmitter must store *N* codewords at a time and large values of *N* result in expensive memory requirements. With *selective-repeat* ARQ, only the errored codeword is retransmitted, thus increasing the throughput over go back-*N* ARQ. Both types of continuous ARQ offer greater throughput efficiency than stop-and-wait ARQ at the cost of greater memory requirements.

B. Forward Error Correction (FEC)

Forward error correction is appropriate for applications where the user must get the message right the *first* time. The one-way or broadcast channel is one example. Today's error correction codes fall into two categories: *block codes* and *convolutional codes.*

Block Codes. The operation of binary block codes was described in Section 4.0 of this paper. All we need to add here is that not all block codes are binary. In fact, one of the most popular block codes is the *Reed-Solomon* code which operates on *m*-bit symbols, not bits. Because Reed-Solomon codes correct symbol errors rather than bit errors, they are very

effective at correcting burst errors. For example, a 2-symbol error correcting Reed-Solomon code with 8 bit-symbols can correct all bursts of length 16 bits or less. Reed Solomon Codes are used in JTIDS, a new deep space standard, and compact disc (CD) players.

Convolutional Codes. With convolutional codes, the incoming bit stream is applied to a K-bit long shift register. For each shift of the shift register, b new bits are inserted and n code bits are delivered, so the code rate is b/n. The power of a convolutional code is a function of its constraint length, K. Large constraint length codes tend to be more powerful. Unfortunately, with large constraint length comes greater decoder complexity. There are several effective decoding algorithms for convolutional codes, but the most popular is the Viterbi algorithm, discovered by Andrew Viterbi in 1967. Viterbi decoders are now available on single integrated circuits (VLSI) from several manufacturers. Viterbi decoders are impractical for long constraint length codes because decoding complexity increases rapidly with constraint length. For long constraint length codes ($K > 9$), a second decoding algorithm called *sequential decoding* is often used. A third decoding technique, *feedback decoding*, is effective on burst-error channels, but is inferior on random error channels. In general, convolutional codes provide higher coding gain than block codes for the same level of encoder/decoder complexity.

One drawback of the codes we have looked at so far is that they all require bandwidth expansion to accommodate the added parity bits if the user wishes to maintain the original unencoded information rate. In 1976, Gottfried Ungerboeck discovered a class of codes that integrates the encoding and modulation functions and does not require bandwidth expansion [4]. These codes are called *Ungerboeck code*s or *trellis coded modulation (TCM)*. Virtually every telephone line modem on the market today operating above 9.6 k bits/s uses TCM.

C. Hybrid ARQ

Hybrid ARQ schemes combine error detection and forward error correction to make more efficient use of the channel. At the receiver, the decoder first attempts to correct any errors present in the received codeword. If it cannot correct all the errors, it requests retransmission using one of the three ARQ techniques described above. *Type I hybrid ARQ* sends all the necessary parity bits for error detection and error correction with each codeword. *Type II hybrid ARQ*, on the other hand, sends only the error detection parity bits and keeps the error correction parity bits in reserve. If the decoder detects errors, the receiver requests the error correction parity bits and attempts to correct the errors with these parity bits before requesting retransmission of the entire codeword. Type II ARQ is very efficient on a channel characterized by a "good" state that prevails most of the time and a "bad" state that occurs infrequently.

D. Interleaving

One of the most popular ways to correct burst errors is to take a code that works well on random errors and *interleave* the bursts to "spread out" the errors so that they appear random to the decoder. There are two types of interleavers commonly in use today, *block interleavers* and *convolutional interleavers*. Figure 3 illustrates the operation of a block interleaver.

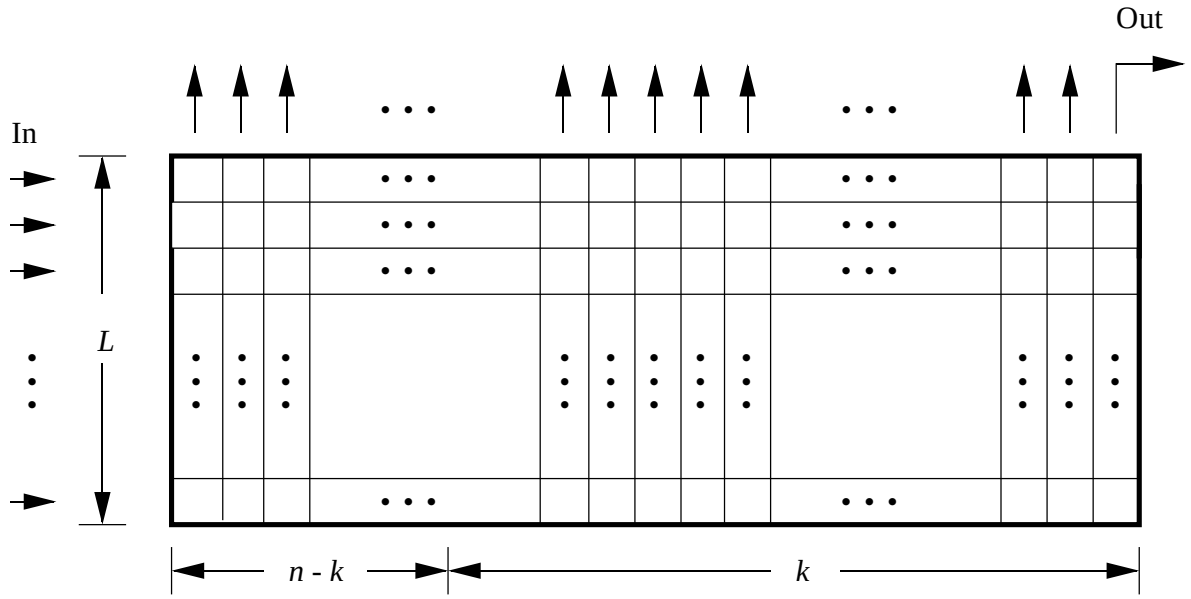

Figure 3 - Block Interleaver

The block interleaver is loaded row by row with *L* codewords, each of length n bits. These *L* codewords are then transmitted column by column until the interleaver is emptied. Then the interleaver is loaded again and the cycle repeats. At the receiver, the codewords are *deinterleaved* before they are decoded. A burst of length *L* bits or less will cause no more than 1 bit error in any one codeword. The random error decoder is much more likely to correct this single error than the entire burst.

The parameter *L* is called the *interleaver degree*, or *interleaver depth*. The interleaver depth is chosen based on worst case channel conditions. It must be large enough so that the interleaved code can handle the longest error bursts expected on the channel. The main drawback of block interleavers is the delay introduced with each row-by-row fill of the interleaver. *Convolutional interleavers* eliminate the problem except for the delay associated with the initial fill. Convolutional interleavers also reduce memory requirements over block interleavers by about one-half [5]. The big disadvantage of either type of interleaver is the interleaver delay introduced by this initial fill. The delay is a function of the interleaver depth and the data rate and for some channels it can be several seconds long. This long delay may be unacceptable for some applications. On voice circuits, for example, interleaver delays confuse

the unfamiliar listener by introducing long pauses between speaker transitions. Even short delays of less than one second are sufficient to disrupt normal conversation. Another disadvantage of interleavers is that a smart jammer can choose the appropriate time to jam to cause maximum damage. This problem is overcome by randomizing the order in which the interleaver is emptied.

In practice, interleaving is one of the best burst-error correcting techniques. In theory, it is the *worst* way to handle burst errors. Why? From a strict probabilistic sense, we are converting "good" errors into "bad" errors. Burst errors have structure and that structure can be exploited. Interleavers "randomize" the errors and destroy the structure. Theory differs from reality, however. Interleaving may be the only technique available to handle burst errors successfully. For example, Viterbi [8] shows that, for a channel impaired by a pulse jammer, exploiting the burst structure is not enough. Interleaving is still required. This does not mean that we should be careless about our choice of code and take up the slack with long interleavers. Codes designed to correct burst errors can achieve the same performance with much shorter interleavers. Until the coding theorists discover a better way, interleaving will be an essential error control coding technique for bursty channels.

E. Erasure Decoding

When the receiver detects the presence of jamming, fading, or some transient malfunction, it may choose to declare a bit or symbol *erased*. For example, the receiver may erase the symbols "A" and "L" from the message SIGNAL to get SIGN– –. This is not the same as *deletion*, which would give SIGN. Because the location of the erased bits is known, erasure decoding usually requires fewer parity bits than error correction decoding. A code with minimum distance d_{min} can correct e erasures if $d_{min} = e + 1$. An error correction code can be used to correct t errors and e erasures as long as $d_{min} \geq 2t + e + 1$ [3]. For example, an error-control code with minimum distance 7 can be used to correct 2 errors and 2 erasures.

F. Concatenation

When two codes are used in series, the combination is called a *concatenated* code. Concatenated codes are often used when a single code cannot correct all types of errors encountered on the channel. The operation of concatenated codes is best illustrated by example. Figure 4 shows the Consultative Committee for Space Data Systems (CCSDS) Blue Book standard for Telemetry Channel Coding (interleaving is omitted for clarity) [2].

The inner code, a rate 1/2, K=7, convolutional code with Viterbi decoding, corrects most random errors and the outer code, a 16-symbol error correcting Reed-Solomon code, cleans up any burst errors that slip through the Viterbi decoder. The Reed-Solomon code operates on 8-bit symbols and is therefore a very powerful burst-error correcting code. The overall code rate is just the product of the two individual code rates, i.e. (1/2)(223/255) ≈ .44. Coding gains for concatenated codes are very large. In fact, some concatenated codes allow operation so

near the thermal noise level that synchronization becomes a problem. Despite their power, concatenated codes are difficult to implement and their typically low code rates (high overhead) may be undesirable on bandlimited channels.

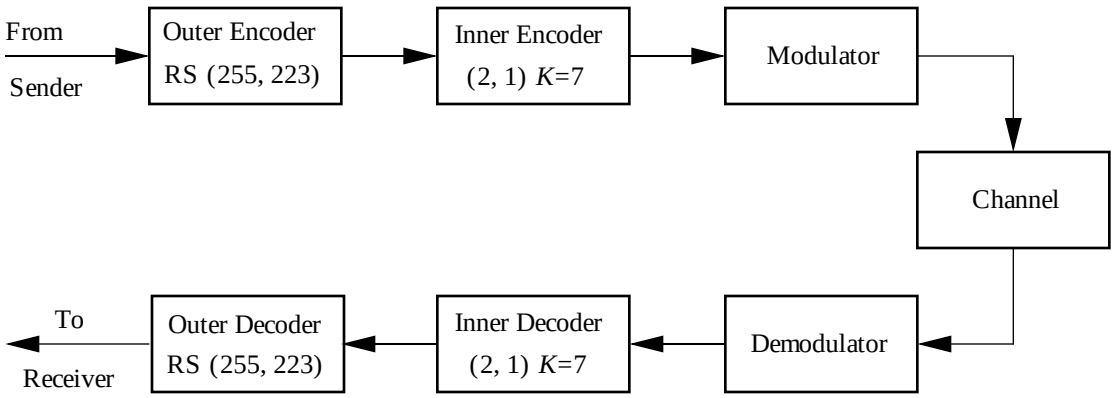

Figure 4 - Concatenated Code for Deep Space Standard

In the next section we will investigate a sophisticated error-control coding system that uses almost all of the error control techniques discussed above. Oddly, this system is not military; it is not expensive; and it is not made in the USA. It is the $200.00 compact disc (CD) player.

6.0 Coding In Compact Disc Players

In 1979, Philips Corp. of the Netherlands and Sony Corp. of Japan joined forces to establish a standard for the new compact disc (CD) digital audio system. Philips had been working on the audio compact disc since 1969 and was looking for a partner. The company wanted to prevent the proliferation of incompatible designs and figured that if the two giants could agree on a standard, the rest of the industry would follow their lead. Philips was right. Today, all compact disc players use the Philips-Sony standard.

At the time of the first meetings between Philips and Sony, Philips had nearly perfected the optics of its compact disc players, but Philips' engineers were having serious problems controlling errors. The slightest scratch or dust particle would obliterate hundreds of bits and cause a "blast like a cannon shot or thunderclap" [9].
To appreciate Philips error-control problems, it helps to understand the basics of how compact disc players work. The recording medium for CD players is a plastic disc 120 mm in diameter used to store digitized audio in the form of minute *pits* that are optically scanned by a laser. The audio waveform is sampled at 44.1 k samples/s to provide a listening bandwidth of 20 kHz. Each audio sample is uniformly quantized to one of 2^{16} levels. (Compare this to 2^8 levels for conventional PCM.) The resulting dynamic range is 96 dB and the total harmonic distortion is less than 0.005 %. Channel errors usually come from two sources: (1) small unwanted particles, air bubbles, or pit inaccuracies from the manufacturing process, and (2)

fingerprints, scratches, or dust particles from handling. Before it collaborated with Sony, Philips was very careful in its demonstrations to play only exceptionally clean discs because its prototype could not tolerate the smallest dust particles, much less scratches and fingerprints. The average consumer is not nearly as careful, so a much more rugged system was required. That was Sony's contribution.

Sony was an ideal partner for Philips because it had practical experience in error correction that the Philips engineers lacked. Philips had tried a convolutional code because of the high coding gain promised by this type of code. Unfortunately, the compact disc was basically a burst-error channel and when the convolutional code failed to correct bursts, it actually made things worse by introducing more errors [9]. After a year of design meetings, Sony and Philips adopted a powerful error control scheme that used multiple layers of interleaving and Reed-Solomon codes. This error-control scheme is called the *cross-interleave Reed-Solomon code (CIRC)*. The CIRC controls errors through a hierarchy of four techniques [5]:

- The decoder provides a level of forward error correction.
- If the error burst exceeds the capability of forward error correction, the decoder provides a level of erasure correction.
- If the error burst exceeds the capability of erasure correction, the decoder hides unreliable samples by *interpolating* between reliable neighbors.
- If the error burst exceeds the capability of the interpolator, the decoder blanks out, or *mutes* the system for the duration of the unreliable samples.

The resulting performance of the CIRC scheme is summarized in Table 2 [5].

Table 2 - Error Control Performance of the CIRC

Max correctable burst	~4 000 bits (2.5 mm on disc)
Max interpolatable burst	~ 12,000 bits (8 mm)
Undetected error samples (clicks)	Less than one every 750 hours at BER of 10^{-3}
New discs typically have BER of	~ 10^{-4}

The CIRC encoder uses three levels of interleaving and two concatenated Reed-Solomon codes. The inner Reed-Solomon code is a (32, 28) code. The outer code is a (28, 24) code. Both codes use 8-bit symbols and the overall code rate is 3/4. The inner code has minimum distance $d_{min} = 5$, so it can correct at most 2 symbol errors. The outer code also has minimum distance of 5 and it will attempt error correction or erasure correction, depending on how successful the inner decoder was at correcting the burst. For a complete description of CIRC encoding and decoding, the reader should consult pages 366-373 of [5]. The CIRC encoding scheme was a tremendous success. It converted a channel that could not tolerate minute dust particles to one that is immune to all but the roughest handling. The compact disc can endure 8 mm holes punched through the disc without noticeable effect.

7.0 Conclusions

Error-control coding can be a cost effective technique for meeting the user's information reliability requirements, provided his or her requirements are specified completely. Unfortunately, performance specifications for communications systems tend to overlook the user's real information requirements. Too often, bit-error-rate is the only information requirement specified. We noted earlier that bit-error-rate has little meaning to the user, who wants error-free *messages*. A seemingly good bit-error-rate on a full duplex channel with ARQ may result in an unacceptable message-error-rate on a broadcast channel. Even if the message error rate is specified, the user still may not get what he or she needs. A system designer constrained only by message length and message-error-rate can trade off time (via low bit rate and long interleaver depth) to meet the user's specifications. Obviously the user can't wait forever, so there should be a specification for maximum message delivery time as well. To completely specify his information requirements, the user must state

- what information he or she wants (completeness),
- how fast he or she wants it delivered (timeliness), and
- how much of it can be in error (reliability).

The answer to the question "How many messages can be in error?" is usually – "None!" This presents a problem for the communications engineer, who knows this requirement is impossible (in practice) to satisfy. However, a message-error-rate of say, 10^{-6} (one in a million on the average) is achievable and is low enough to make the channel error-free for all practical purposes.

8.0 To Find Out More

References on coding for the non-engineer are practically non-existent. This paper was written to partially fill that void. On the other hand, the practicing engineer has several choices. For years, the classic textbook on error-control coding was Peterson and Weldon's Error Correcting Codes (MIT Press, 1971). Twenty-five-year-old textbooks are long since obsolete, so the author recommends newer texts by Blahut (Ref 3, 1983) and Lin and Costello (Ref 1, 1983). The author also recommends Michelson's and Levesque's Error Control Techniques For Digital Communication (Wiley, 1985) which is not a textbook and was written especially for the practicing engineer. Berlekamp (Ref. 2, 1987) summarizes the latest applications of error control coding and proposes some common sense approaches to solving error-control problems. One of the best sources available on fading channels is Bogusch's Digital Communications in Fading Channels: Modulation and Coding (Mission Research Corporation, 1987 – unclassified report prepared for AF Weapons Lab under contract F29601-82-C-0023). An excellent history of the development of the compact disc player is found in *IEEE Spectrum* (Ref 9, 1988). These recommendations reflect the author's personal prejudices and should not be taken as the only satisfactory references on the subject. There are dozens of excellent books on error-control coding.

9.0 References

[1] S. Lin and D. J. Costello, Error Control Coding: Fundamentals and Applications, Englewood Cliffs, New Jersey: Prentice Hall, 1983.

[2] E. R. Berlekamp et al., "The application of error control to communications," *IEEE Communications Magazine*, vol. 25, No. 4, pp. 44-57, April 1987.

[3] R. E. Blahut, Theory and practice of error control codes, Reading, Massachusetts: Addison-Wesley Publishing Company, 1983.

[4] G. Ungerboeck, "Trellis coded modulation with redundant signal sets Parts I and II," *IEEE Communications Magazine*, vol. 25, No. 2, pp. 5-21, February 1987.

[5] B. Sklar, Digital Communications Fundamentals and Applications, Englewood Cliffs, New Jersey: Prentice Hall, 1988.

[6] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, pp. 379-423, 1948.

[7] T. Fujiwara, T. Kasami, A. Kitai, and S. Lin, "On the undetected error probability for shortened Hamming codes," *IEEE Transactions on Communications*, vol. COM-33, pp. 570-574, June 1985.

[8] A. J. Viterbi, "Spread Spectrum Communications – Myths and Realities," *IEEE Communications Magazine*, vol. 17, No. 5, pp. 11-18, May 1979.

[9] F. Guterl, "Compact Disc," *IEEE Spectrum*, vol. 25, No. 11, pp. 102-108, 1988.