CHANNEL CODING

The general term *channel coding* implies a technique by which redundancy symbols are attached to the data by a channel encoder of the system. These redundancy symbols are used to detect and/or correct and/or interpolate erroneous data at the channel decoder. Channel encoding is achieved by imposing relations on the information data and redundancy symbols of the system. These restricting relations make it possible for the decoder to correctly extract the original source signal with high reliability and fidelity from a possibly corrupted received or retrieved signal.

Channel coding is used in digital communication systems for several possible reasons: (1) to increase the reliability of noisy data communications channels or data storage systems; (2) to control errors in such a manner that a faithful reproduction of the data can be obtained; (3) to increase the overall signal-to-noise energy ratio (SNR) of a system; (4) to reduce the noise effects within a system; and (5) to meet the commercial demands of efficiency, reliability, and a high performance of an economically practical digital transmission and storage system. All of these objectives must be tailored to the particular application. Therefore, channel coding is also called *error-control coding, error-correction,* or *detection coding.*

ERROR-HANDLING PROCESSES AND ERROR-CONTROL STRATEGIES

Figure 1 shows a physical layer coding model of a digital communication system. The same model can be used to describe an information storage system if the storage medium is considered to be the channel.

The source information is usually composed of binary or alphanumeric symbols. The encoder converts the information messages into electrical signals acceptable to the channel. Then these signals are sent to the channel (or storage medium), where they may be disturbed by noise. Next, the output of the channel is sent to the decoder, which makes a decision to determine which message was sent. Finally, this message is delivered to the recipient data (sink).

Typical transmission channels are twisted-pair telephone lines, coaxial cable wires, optical fibers, radio links, microwave links, satellite links, and so forth. Typical storage media can be semiconductor memories, magnetic tapes and discs, compact discs (CDs), optical memory units, or digital video discs (DVDs). Each of these channels or media is subject to various types of noise disturbances. For example, the disturbance on a telephone line may come from impulsive circuit switching noise, thermal noise, crosstalk between lines, or a loss of synchronization. The disturbances on a CD often are caused by surface defects, dust, or a mechanical failure. Therefore, the problems a digital communication system faces are the possible message errors that might be caused by these different disturbances. To overcome these problems, "good" encoders and decoders need to be designed to improve the performance of these channels. Figure 2 shows the block diagram of a typical error handling system.

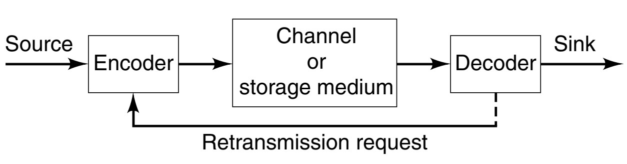


Figure 1. A physical layer model of a communication or storage system.

In an ideal system, the symbols that are obtained from the channel (or storage medium) should match the symbols that originally entered the channel (or storage medium). In any practical system, there often are occasional errors, and the purpose of channel coding is to detect and possibly correct such errors.

The first stage in Fig. 2 is concerned with encoding for error avoidance and the use of redundancy. This includes, for example, such processes as precoding data for modulation, the placing of digital data at an appropriate position on the tape for certain digital formats, the rewriting of a read-after-write error in a computer tape, and error-correction and detection encoding. Following these moves, the encoded data are delivered to the modulator in the form of a signal vector or code. Then the modulator transforms the signal vector into a waveform that matches the channel. After being transmitted through the channel, the waveform often is disturbed by noise. The demodulation of this waveform can produce corrupted signal vectors, which in turn cause possible errors in the data. On receipt of the data, errors are first detected. The detection of an error then requires some course of action. For example, in a bi-directional link a retransmission might be requested. Finally, the correctable error patterns can be eliminated effectively by an error-correction engine.

The error-control strategy for the error-handling system shown in Fig. 2 depends primarily on the application. That is, such a strategy depends on the channel properties of the particular communication link and the type of error-control codes to be used.

Without the feedback line, shown in Fig. 2, communication channels are one-way channels. The codes in this case are mainly designed for error-correction and/or error-concealment. Error control for a one-way system is usually accomplished by the use of *forward error correction* (FEC), that is, by employing error-correcting codes that automatically correct errors which are detected at the receiver. Communication systems frequently employ two-way channels, a fact that must be considered in the design of an error-control system. With a two-way channel, both error-correction and error-detecting codes can be used. When an error-detecting code is used and an error is detected at one terminal, a request for a repeat is given to the transmitting terminal.

There are examples of real one-way channels in which the error probabilities can be reduced by the use of an error-correcting code but not by an error detection and retransmission system. For example, with a magnetic-tape storage system, usually too much time has passed to ask for a retransmission after the tape has been stored for any significant period of time, say a week or sometimes just a day. In such a case, errors are detected when the record is read. Encoding with FEC codes is usually no more complex than it is with error-detecting codes. It is the decoding that requires sophisticated digital equipment.

On the other hand, there are good reasons for using both error detection and retransmission for some applications when possible. Error detection is by its nature a much simpler computational task than error correction and requires much

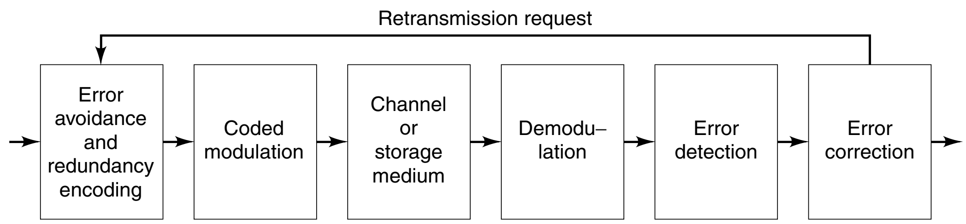


Figure 2. The major processes in an error-handling system.

less complex decoding equipment. Also, error detection with retransmission tends to be adaptive. In a retransmission system, redundant information is utilized only in the retransmitted data when errors occur. This makes it possible, under certain circumstances, to obtain better performance with a system of this kind than is theoreticaly possible over a one-way channel. Error control by the use of error detection and retransmission is called *automatic repeat request* (ARQ). In an ARQ system, when an error is detected at the receiver, a request is sent to the transmitter to repeat the message. This process continues until it is verified that the message was received correctly. Typical applications of ARQ are the protocols for many fax modems.

There is a definite limit to the efficiency of a system that uses simple error detection and retransmission alone. First, short error-detecting codes are not efficient detectors of errors. On the other hand, if very long codes are used, retransmission must be done too frequently. It can be shown that a combination of both the correction of the most frequent error patterns along with detection and retransmission of the less frequent error patterns is not subject to such a limitation. Such a mixed error-control process is usually called a *hybrid error-control* (HEC) strategy. In fact, HEC is often more efficient than either a forward error correction system or a detection and retransmission system. Many present-day digital systems use a combination of forward error correction and detection with or without feedback.

If the error rate demanded by the application cannot be met by the unaided channel or storage medium, some form of error handling may be necessary.

BASIC PRINCIPLES OF ERROR-CONTROL CODES

We have seen that the performance of an error-handling system relies on error-correction and/or detection codes that are designed for the given error-control strategy. There are two different types of codes that are commonly used today: block and convolutional codes. It is assumed for both types of codes that the information sequence is encoded using an alphabet set Q of q distinct symbols, called a q-ary set, where q is a positive integer.

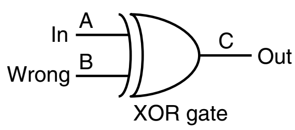
In general, a code is called a block code if the coded information sequence can be divided into blocks of n symbols and each block can be decoded independently. The encoder of a block code divides the information sequence into message blocks of k information symbols each. A message block, called the *message word,* is represented by the k-tuple $\boldsymbol{m} = (m_0, m_1, . . ., m_{k-1})$ of symbols. Evidently, there are a total of q^k different possible message words. The encoder transforms each message word $\boldsymbol{m}$ independently into an n-symbol *codeword* $\boldsymbol{c} = (c_0, c_1, . . ., c_{n-1})$. Therefore, corresponding with the q^k different possible messages, there are q^k different possible codewords at the encoder output. This set of q^k codewords of length n is called an (n, k) block code. The code rate of an (n, k) block code is defined to be $R = k/n$. If $q = 2$, the codes are called binary block codes and can be implemented with a combinational logic circuit.

The encoder for a convolutional code accepts k-bit blocks of an information sequence and produces an encoded sequence of n-bit blocks. (In convolutional coding, the symbols are used to denote a sequence of blocks rather than a single block.) However, each encoded block depends not only on the corresponding k-bit message block, but also on the m previous message blocks. Hence, the encoder is said to have a *memory of order* m. The set of encoded sequences produced by a k-input and n-output encoder of memory of order m is called an (n, k, m) convolutional code. Again, the ratio $R = k/n$ is called the code rate of the convolutional code. Since the encoder contains memory, it is implemented with a sequential logic circuit.

The basic principle of error-control coding is to add redundancy to the message in such a way that the message and its redundancy are related by some set of algebraic equations. When a message is disturbed, the message with such constrained redundancy still can be decoded by the use of these relations. In other words, the error-control capability of a code comes from this relational redundancy which is added to the message during the encoding process.

To illustrate the principle of error-control coding, an example of a binary block code of length 3 is discussed. There are a total of eight different possible binary 3-tuples: (000), (001), (010), (011), (100), (101), (110), (111). First, if all of these 3-tuples are used to transmit messages, one has the example of a (3,3) binary block code of rate 1. In this case, if a one-bit error occurs in any codeword, the received word becomes another codeword. Since any particular codeword may be a transmitted message and there are no redundancy bits in a codeword, errors can neither be detected nor corrected.

The error detection and correction processes are closely related and will be dealt with presently. The actual correction of an error is simplified tremendously by the adoption of binary codes. There are only two symbols, 0 and 1, in this case. Hence, to correct a symbol it is sufficient to know that the symbol is wrong. Figure 3 shows the minimal circuit needed for correction once the bit in error has been identified. The exclusive-OR (XOR) gate shows up extensively in error correction circuits, and the figure also demonstrates its truth table. One way to remember the characteristics of this useful device is that there always is an output "1" when the inputs are different. Inspection of the truth table shows that there is an even number of 1's in each row and, as a consequence, the device is also called an even parity gate.



Truth table of XOR gate

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

Figure 3. Exclusive-OR Gate.

Parity is a fundamental concept in error detection. In the previous example, let only four of the 3-tuples, (000), (011), (101), (110), be chosen as codewords for transmission. These are equivalent to the four 2-bit messages, (00), (01), (10), (11), with the third bit in each 3-tuple equal to the XOR of its first and second bits. This is an example of a (3,2) binary block code of rate 2/3. If a received word is not a codeword, i.e., the third bit does not equal the XOR of the first and second bits, then an error is detected. However, this code cannot correct any error. To illustrate what can happen when there are errors, suppose that the received word is 010. Such an error cannot be corrected even if there is only one bit in error since, in this case, the transmitted codeword has three possibilities: (000), (011), (110).

To achieve error correction, more redundancy bits need to be added to the message words for transmission. Suppose only the two 3-tuples (000), (111) are chosen as codewords. This is a (3,1) binary block code of rate 1/3. The codewords (000), (111) are encoded by duplicating the source bits 0, 1 two additional times, that is, two redundancy bits in each codeword. If this codeword is sent through the channel, and one- or two-bit errors occur, the received word is not a codeword. Errors are detected in this scenario. If the decision (rule of the decoder) is to decide the original source bit as the bit which appears as the majority of the three bits of the received word, a one-bit error is corrected. For instance, if the received word is (010), this decoder would say that 0 was sent.

Consider next the example of a (2,1,2) binary convolutional code. Let the information sequence be $\boldsymbol{m} = (m_0, m_1, \ldots, m_6) = (1011100)$ and the encoded sequence be $\boldsymbol{c} = (c_0^{(1)}c_0^{(2)}, c_1^{(1)}c_1^{(2)}, \ldots, c_6^{(1)}c_6^{(2)}) = (11,10,00,01,10,01,11)$. Also assume the relations between the components of vectors $\boldsymbol{m}$ and $\boldsymbol{c}$ are given by

$$c_i^{(1)} = m_{i-2} + m_{i-1} + m_i$$
$$c_i^{(2)} = m_{i-2} + m_i$$

where $m_{-2} = m_{-1} = 0$ and "+" means sum modulo 2. Suppose the third digit of the received sequence is in error. That is, let the received sequence begin with $\boldsymbol{c} = (11,00,00,\ldots)$. The following are the eight possible beginning code sequences (00,00,00, . . .), (00,00,11, . . .), (00,11,10, . . .), (00,11,01, . . .), (11,10,11, . . .), (11,10,00, . . .), (11,01,01, . . .), (11,01,10, . . .). Clearly, the sixth path, which differs from the received sequence in but a single position, is intuitively the best choice. Thus, a single error is corrected by this observation. Next, suppose digits 1, 2, and 3 were all erroneously received. For this case, the closest code sequence would be (00,00,00, . . .) and the decoder would make an undetectable error. For further understanding of convolutional codes, readers may refer to Refs. 1 and 2.

LINEAR BLOCK CODES

In the previous section, it was shown for a binary block code that the positions of the failed bits can be determined by the use of more parity bits. If these parity bits are generated by a linear combination of message bits, the code is called a *linear block code.* Some important concepts of a linear block code are introduced next by an example of the *Hamming code.*

Consider a binary linear block code of length 7 and rate $R = 4/7$. A four-bit message word $\boldsymbol{m} = (m_0, m_1, m_2, m_3)$ is used to compute three redundancy bits and to make a seven-bit codeword $\boldsymbol{c} = (c_0, c_1, \ldots, c_6)$ from the following set of equations:

$$c_i = m_i, \quad i = 0, 1, 2, 3$$
$$c_4 = m_0 + m_2 + m_3$$
$$c_5 = m_0 + m_1 + m_2$$
$$c_6 = m_0 + m_1 + m_3$$

These equations provide the *parity-check equations* in the matrix form, $\boldsymbol{c}H^T = 0$, where

$$H = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

is called the *parity-check matrix* of the code and "T" denotes matrix transpose. The codewords are generated by $\boldsymbol{c} = \boldsymbol{m} \cdot G$, where

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

is called the *generator matrix* of the code.

In the Hamming code, four message bits are examined in turn, and each bit that is a "1" causes the corresponding row of G to be added to an XOR sum. For example, if the message word is (1001), the top and bottom rows of G are componentwise XORed. The first four columns of G form a submatrix, which is known as an identity matrix. Therefore, the first four data bits in the codeword are identical to the message bits that were to be conveyed. This is useful because the original message bits are encoded in an unmodified form, and the check bits are simply attached to the end of the message to construct the so-called systematic codeword. Almost all channel block coding systems use *systematic codes.*

The redundancy or parity bits are calculated in such a manner that they do not use every message bit. If a message bit is not included in a parity check, it can fail without affecting the outcome of that check. For example, if the second bit of a codeword fails, the outcome of the parity-check equation given by the first row of H is not affected. However, the outcomes of other parity-check equations, given by the second and third rows of H, are affected. The position of the error is deduced from the pattern of these successful and unsuccessful checks in the parity-check matrix. This pattern is known as a syndrome defined by the matrix equation $\boldsymbol{s} = \boldsymbol{r} \cdot H^T$, where $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}$ and $\boldsymbol{e}$ is a seven-bit error vector.

In the previous example of the Hamming code, let a failed second bit be assumed in a received word $\boldsymbol{r}$, i.e., $e_1 = 1$, and $e_i = 0$ for $i \neq 1$. Because this bit is included in only two of the parity-check equations, there are two 1's in the failure pattern, namely, 011. Since considerable care was taken in the design of the matrix pattern for generating the check bits, the syndrome, 011, is actually the address (i.e., $[011]^T$ is the first column of H) of the error bit. This is a fundamental feature of the original Hamming codes, due to Richard Hamming in 1950.

It is useful at this point to introduce the concept of *Hamming distance*. This is the number of positions in which two sequences of length n differ. The *minimum* (Hamming) distance d of a block code is by definition the least Hamming distance between any two distinct codewords of this code. That is, the minimum distance of a binary code equals the minimum number of bits that needs to be changed in order to change any codeword into any other codeword. A linear code of length n, dimension k, and minimum distance d is often denoted by the notation (n, k, d).

If errors corrupt a codeword so that it is no longer a codeword, it is definitely detectable and possibly correctable. If errors convert one codeword into another, it is impossible to detect. Therefore, the minimum distance d indicates the detection and correction capacities of the code. For the Hamming code example it can be found by a direct verification for the $2^4 = 16$ codewords that the minimum distance of the Hamming code is 3. This coincides with the fact that two or fewer bit errors in any codeword of the Hamming code produce a noncodeword. Hence two bit errors are always detectable.

Correction is also possible if the following minimum distance rule is used: Correction (decoding) with the minimum distance rule decodes each received word $\boldsymbol{r}$ to the codeword that is closest to it in Hamming distance. For example, if the received word from the Hamming code is $\boldsymbol{r} = (1111100)$ and an error occurs in the second bit [i.e., $\boldsymbol{e} = (0100000)$], the minimum distance rule always correctly decodes $\boldsymbol{r}$ to the codeword $\boldsymbol{c} = (1011100)$. It can be shown that the Hamming code is able to correct all single-bit errors. Associated with this fact is the important theorem for a linear block code given next.

Theorem A linear block code (n, k, d) has the following minimum distance decoding:

1. If $d \geq e + 1$, then the code can detect e errors.
2. If $d \geq 2t + 1$, then the code can correct t errors.
3. If $d \geq t + e + 1$ for $e \geq t$, then the code can correct t errors and simultaneously detect e errors.

Intuitively, item (2) of this theorem can be explained as follows: If a codeword $\boldsymbol{c}$ is transmitted and errors occur in $\leq t$ positions, then the received word $\boldsymbol{r}$ clearly resembles the transmitted codeword $\boldsymbol{c}$ more than any other codeword.

It has been shown for the Hamming code that the syndromes $\boldsymbol{s}$ are the addresses of the error bits. This concept can be generalized to all linear block codes. That is, each syndrome corresponds to one and only one error vector if the number of errors satisfies $e \leq \rfloor(d-1)/2\lfloor$. Another simpler decoding method, called syndrome decoding, can be shown to be equivalent to the minimum distance decoding rule:

1. Compute the syndrome $\boldsymbol{s}$ of the received word $\boldsymbol{r}$.
2. Determine the error vector $\boldsymbol{e}$ that corresponds to the syndrome $\boldsymbol{s}$ by a lookup table.
3. Decode $\boldsymbol{r}$ by choosing the corrected codeword to be $\boldsymbol{r} - \boldsymbol{e}$.

CYCLIC CODES

The implementation of the encoder of a Hamming code can be made very fast by the use of the parity-check equations. Such an implementation is ideal for some applications, such as computer memory protection, which requires short codes and a fast access time. However, in many other applications, the messages are transmitted and stored serially, and it is desirable to use relatively large data blocks to reduce the memory storage devoted to preambles, addressing, and synchronization. Where large data blocks are to be handled, the use of simple parity-check equations for encoding has to be abandoned because it would become impossibly complex. However, the principle of the generator and the parity-check matrices can still be employed for the encoder, but now these matrices usually are generated algorithmically. For the decoder, the syndromes are used to find the bits in error not by using a simple lookup table, but by solving algebraic equations.

A subclass of linear block codes, called cyclic codes, can provide long codes that have the required encoding and decoding structures. A linear code C of length n is said to be cyclic if every cyclic shift of a codeword c is also a codeword, that is,

$$c = (c_0, c_1, \ldots, c_{n-1}) \in C \Rightarrow c^{\pi} = (c_{n-1}, c_0, c_1, \ldots, c_{n-2}) \in C$$

When messages can be accessed serially, simple circuitry can be used for the encoder since the same gate can be used for many XOR operations. Unfortunately, the reduction in complexity of the encoding process is paralleled by an increase in the difficulty of explaining what takes place. The methodology described so far about how an error-correction system works is mainly in engineering terms. However, it can be described more mathematically so that the encoding and decoding of long codes can be accomplished in a more efficient manner.

Toward this end, codewords of a cyclic code can be represented by the use of polynomials in the following way:

$$c = (c_0, c_1, \ldots, c_{n-1}) \in C \Rightarrow$$
$$c(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1} \in C(x)$$

where $C(x)$ represents the set of polynomials that is associated with the set of all codewords of C. The term $c(x)$ is called a code polynomial. It is clear that $c^{\pi}(x) = x \cdot c(x) - c_{n-1} \cdot (x^n - 1) \in C(x)$, that is, $c^{\pi}(x) \equiv x \cdot c(x) \bmod (x^n - 1)$. Let $R_n[x] = \mathrm{GF}(2)[x]/(x^n - 1)$ denote the polynomial ring of degree at most $n - 1$ over the finite (or Galois) field GF(2) of two elements. Let $g(x)$ be the unitary polynomial of smallest degree in $C(x)$. Then the degree of $g(x)$ equals $n - k$, and every polynomial $c(x) \in C(x)$ can be represented as $c(x) = m(x)g(x)$ for some $m(x) \in R_n[x]$.

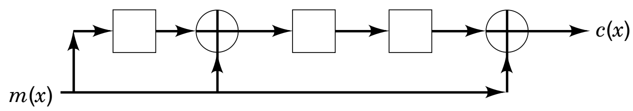


Figure 4. An encoder of the (7,4,3) cyclic Hamming code.

Therefore, a cyclic code encoder can be conceived to be a polynomial multiplier that can be implemented by the use of what is called a shift register. For example, the cyclic Hamming code has the generator polynomial $g(x) = x^3 + x + 1$. Hence, one implementation of the encoder of this code is the shift register device shown in Fig. 4. The register of the encoder is initially set to zero. Then let the message word, $m(x) = m_0 + m_1x + m_2x^2 + m_3x^3$, be the input in sequence from m_3 to m_0. After shifting seven times, a codeword $c(x) = c_0 + c_1x + \cdots + c_6x^6$ is the sequential output of the bits c_6 to c_0.

Many other methods for encoding cyclic codes can be implemented by the use of shift register circuits. The most useful of these techniques is the systematic encoding method. Encoding of an (n, k) cyclic code in systematic form consists of three steps: (1) multiply the message polynomial $m(x)$ by x^{n-k}; (2) divide $x^{n-k}m(x)$ by $g(x)$ to obtain the remainder $b(x)$; and (3) form the codeword $c(x) = b(x) + x^{n-k}m(x)$.

Recall that in the decoding of a linear code, the first step is to compute the syndrome vector $\boldsymbol{s}$ from $\boldsymbol{r}$ by $\boldsymbol{s} = \boldsymbol{r} \cdot H^T$. If the syndrome is zero, the decoder accepts $\boldsymbol{r}$ as a codeword. If the syndrome is not equal to zero, $\boldsymbol{r}$ is not a codeword and the presence of errors is detected. For a cyclic code in systematic form, the syndromes are computed easily. The received word $\boldsymbol{r}$ is treated as the polynomial of degree $n - 1$ or less, i.e., $r(x) = r_0 + r_1x + r_2x^2 + \cdots + r_{n-1}x^{n-1}$. A polynomial division of $r(x)$ by the generator polynomial $g(x)$ yields $r(x) = a(x)g(x) + s(x)$, where the remainder $s(x)$ is a polynomial of degree $n - k - 1$ or less. The $n - k$ coefficients of $s(x)$ form the syndrome $\boldsymbol{s}$. Therefore, $s(x)$ is called the syndrome polynomial of the cyclic code.

In general, the decoding of cyclic codes for error-correction consists of the same three steps used for decoding any linear code: syndrome computation, association of the syndrome with the error pattern, and error correction. However, the limit to this approach is the complexity of the decoding circuit that is needed to determine the error word from the syndrome. Such procedures tend to grow exponentially with code length and the number of errors that need to be corrected. Many cyclic codes have considerable algebraic and geometric properties. If these properties are properly used, then a simplification in the decoding process is usually possible.

Cyclic codes are well suited to error detection, and several have been standardized for use in digital communications. The most common of these have the following generator polynomials:

$$x^{16} + x^{15} + x^2 + 1 \quad (CRC - 16)$$
$$x^{16} + x^{12} + x^5 + 1 \quad (CRC - CCITT)$$

These codes can detect many combinations of errors, and the implementation of both the encoding and error-detecting circuits is quite practical. Since every codeword of a cyclic code can be computed from its generator polynomial $g(x)$ as the product $c(x) = d(x)g(x)$, it is clear that $s(x) \equiv 0 \bmod (g(x))$ if and only if the received polynomial $r(x)$ is a code polynomial. This very useful fact is often employed to design the efficient error-detecting circuits of most cyclic codes.

The results discussed in this section are easily generalized to codes constructed over any finite field GF(q), where q is some power of a prime number p.

BCH CODES

One class of cyclic codes was introduced in 1959 by Hocquenghem, and independently in 1960 by Bose and Ray-Chaudhuri. The codes are known as BCH codes and can be described by means of the roots of a polynomial $g(x)$ with coefficients in a finite field. A cyclic code of length n over GF(2) is called a BCH code of designed distance δ if its generator $g(x)$ is the least common multiple of the minimal polynomials of $\beta^l, \beta^{l+1}, \ldots, \beta^{l+\delta-2}$ for some l, where β is a primitive nth root of unity. If $n = 2^m - 1$, i.e., β is a primitive element of GF(2^m), then the BCH code is called a primitive BCH code. The performance of a BCH code is specified by its designed distance using the following fact: the minimum distance of a BCH code with designed distance δ is at least δ. This fact is usually called the BCH bound. A primitive BCH code of designed distance δ has the minimum distance $d \leq 2\delta - 1$.

To decode BCH codes, let's once again consider a BCH code of length n over GF(2) with designed distance $\delta = 2t + 1$ and let β be a primitive nth root of unity in GF(2^m). Consider a codeword $c(x)$ and assume that the received word is

$$r(x) = r_0 + r_1x + \cdots + r_{n-1}x^{n-1}$$

Let $e(x) = r(x) - c(x) = e_0 + e_1x + \cdots + e_{n-1}x^{n-1}$ be the error vector. Now, define the following:

$M = \{i | e_i \neq 0\}$ is the set of positions where errors occur.

$e = |M|$ is the number of errors.

The polynomial $\sigma(z) = \prod_{i\in M}(1 - \beta^i z)$ in z is called the error-locator polynomial.

Also, let $\omega(z) = \sum_{i\in M} e_i\beta^i z \prod_{j\in M\setminus\{i\}}(1 - \beta^j z)$ be what is known as the error-evaluator polynomial.

It is clear that if one can find $\sigma(z)$ and $\omega(z)$, then the errors can be corrected. In fact, an error occurs in position i if and only if $\sigma(\beta^{-i}) = 0$, and in that case the error is given by $e_i = -\omega(\beta^{-i})\beta^i/\sigma'(\beta^{-i})$, where $\sigma'(\cdot)$ denotes the derivative. Assume that the number of errors $e \leq t$ (if $e > t$, one does not expect to be able to correct the errors). Observe that

$$\frac{\omega(z)}{\sigma(z)} = \sum_{i\in M} \frac{e_i\beta^i z}{1 - \beta^i z} = \sum_{i\in M} e_i \sum_{l=1}^{\infty} (\beta^i z)^l$$
$$= \sum_{l=1}^{\infty} z^l \sum_{i\in M} e_i \beta^{li} = \sum_{l=1}^{\infty} z^l e(\beta^l)$$

where all of these calculations use the operations of what is known as a formal power series over the finite field GF(2^m). For $1 \leq l \leq 2t$, one gets $e(\beta^l) = r(\beta^l)$, i.e., the receiver knows the first $2t$ coefficients on the right-hand side of the equation. Therefore, $\omega(z)/\sigma(z)$ is known as $\bmod z^{2t+1}$. It is claimed that the receiver must determine polynomials $\sigma(z)$ and $\omega(z)$ in such a manner that $\deg[\omega(z)] \leq \deg[\sigma(z)]$ with $\deg[\sigma(z)]$ being as small as possible under the condition,

$$\frac{\omega(z)}{\sigma(z)} \equiv \sum_{l=1}^{2t} z^l r(\beta^l) (\bmod z^{2t+1})$$

In practice, it is very important to find a fast algorithm that actually determines $\sigma(z)$ and $\omega(z)$ by solving these equations. Two commonly used algorithms are the Berlekamp–Massey decoding algorithm introduced by E. R. Berlekamp and J. Massey, and the Euclidean algorithm. Interested readers may refer to (1,3–5).

REED–SOLOMON CODE

A very useful class of nonbinary cyclic codes is called the Reed–Solomon (RS) code. RS codes were first discovered by I. S. Reed and G. Solomon in 1958. An RS code is defined over GF(p^m) with length $p^m - 1$ and minimum distance $d = n - k + 1$. Its generator polynomial is $g(x) = (x - \alpha^u)(x - \alpha^{u+1}) \cdots (x - \alpha^{u+d-2})$, where α is a primitive element in GF(p^m) and where u is some integer. Since RS codes are cyclic, they can be encoded by the product of $g(x)$ and the polynomial associated with the information vector, or by a systematic encoding.

For example, the RS code of length $n = 7$, dimension $k = 5$, and minimum distance $d = 3$ where $p = 2$ is specified by the generator polynomial $g(x) = (x - \alpha^3)(x - \alpha^4) = x^2 + \alpha^6 x + 1$, where α is a root of the irreducible polynomial $x^3 + x + 1$ and is a primitive element of the finite field GF(2^3).

In the RS codes, data bits are assembled into words, or symbols, which become elements of the Galois field upon which the code is based. The number of bits in the symbol determines the size of the Galois field, and hence the number of symbols in a codeword. A symbol length of eight bits is commonly used because it fits in conveniently with modern byte-oriented computers and processors. The Galois, or finite, field with eight-bit symbols is denoted by GF(2^8). Thus, the RS codes defined over GF(2^8) have a length of $2^8 - 1 = 255$ symbols. As each symbol contains eight bits, the codeword is $255 \times 8 = 2040$ bits long. A primitive polynomial commonly used to generate GF(2^8) is $g(x) = x^8 + x^4 + x^3 + x^2 + 1$. The decoders of RS codes are usually implemented by the Euclidean algorithm and the Berlekamp–Massey algorithm.

NOISY CHANNEL CODING THEOREM

Several important classes of codes have been discussed in the previous sections. In this section the performance to be expected from channel coding is discussed briefly. Some commonly used quantities for measuring performance improvement by channel coding include the error probabilities from the decoder, such as the bit-error rate of the system, the probability of an incorrect decoding of a codeword, and the probability of an undetected error. In the physical layers of a communication system, these error probabilities usually depend on the particular code, the decoder, and, more importantly, on the underlying channel/medium error probabilities.

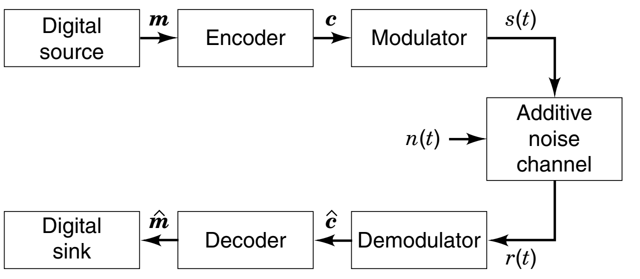


Figure 5. Coded system on an additive noise channel.

Figure 5 shows a block diagram of a coded system for an additive noise channel. In such a system, the source output $\boldsymbol{m}$ is encoded into a code sequence (codeword) $\boldsymbol{c}$. Then $\boldsymbol{c}$ is modulated and sent to the channel. After demodulation the decoder receives a sequence $\boldsymbol{r}$ which satisfies $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}$, where $\boldsymbol{e}$ is the error sequence and "+" usually denotes component-wise vector XOR addition. The final decoder output $\hat{m}$ represents the recovered message.

The primary purpose of a decoder is to produce an estimate $\hat{m}$ of the transmitted information sequence $\boldsymbol{m}$ that is based on the received sequence $\boldsymbol{r}$. Equivalently, since there is a one-to-one correspondence between the information sequence $\boldsymbol{m}$ and the codeword $\boldsymbol{c}$, the decoder can produce an estimate $\hat{c}$ of the codeword $\boldsymbol{c}$. Clearly, $\hat{m} = \boldsymbol{m}$ if and only if $\hat{c} = \boldsymbol{c}$. A decoding rule is a strategy for choosing an estimated codeword $\hat{c}$ for each possible received sequence $\boldsymbol{r}$. If the codeword $\boldsymbol{c}$ was transmitted, a decoding error occurs if and only if $\hat{c} \neq \boldsymbol{c}$. On the assumption that $\boldsymbol{r}$ is received, the conditional error probability of the decoder is defined by

$$P(E|\boldsymbol{r}) = P(\hat{c} \neq \boldsymbol{c}|\boldsymbol{r}) \tag{1}$$

The error probability of the decoder is then given by

$$P(E) = \sum_r P(E|\boldsymbol{r})P(\boldsymbol{r}) \tag{2}$$

where $P(\boldsymbol{r})$ denotes the probability of receiving the codeword $\boldsymbol{r}$ and the summation is over all possible received words. Evidently, $P(\boldsymbol{r})$ is independent of the decoding rule used since $\boldsymbol{r}$ is produced prior to the decoding process. Hence, the optimum decoding rule must minimize $P(E|\boldsymbol{r}) = P(\hat{c} \neq \boldsymbol{c}|\boldsymbol{r})$ for all $\boldsymbol{r}$. Since minimizing $P(\hat{c} \neq \boldsymbol{c}|\boldsymbol{r})$ is equivalent to the maximization of $P(\hat{c} = \boldsymbol{c}|\boldsymbol{r})$, $P(E|\boldsymbol{r})$ is minimized for a given $\boldsymbol{r}$ by choosing $\hat{c}$ to be some codeword $\boldsymbol{c}$ that maximizes

$$P(\boldsymbol{c}|\boldsymbol{r}) = \frac{P(\boldsymbol{r}|\boldsymbol{c})P(\boldsymbol{c})}{P(\boldsymbol{r})} \tag{3}$$

That is, $\hat{c}$ is chosen to be the most likely codeword, given that $\boldsymbol{r}$ is received. If all codewords are equally likely, i.e., $P(\boldsymbol{c})$ is the same for all $\boldsymbol{c}$, then maximizing Eq. (3) is equivalent to the maximization of the conditional probability $P(\boldsymbol{r}|\boldsymbol{c})$.

If each received symbol in $\boldsymbol{r}$ depends only on the corresponding transmitted symbol, and not on any previously transmitted symbol, the channel is called a discrete memoryless channel (DMC). For a DMC, one obtains

$$P(\boldsymbol{r}|\boldsymbol{c}) = \prod_i P(r_i|c_i) \tag{4}$$

since for a memoryless channel each received symbol depends only on the corresponding transmitted symbol. A decoder that chooses its estimate to maximize Eq. (4) is called a maximum likelihood decoder (MLD).

One of the most interesting problems in channel coding is to determine for a given channel how small the probability of error can be made in a decoder by a code of rate $\boldsymbol{R}$. A complete answer to this problem is provided to a large extent by a specialization of an important theorem, due to Claude Shannon in 1948, called the *noisy channel coding theorem* or the *channel capacity theorem.* Roughly speaking, Shannon's noisy channel coding theorem states: For every memoryless channel of capacity C, there exists an error-correcting code of rate $R < C$ such that the error probability $P(E)$ of the maximum likelihood decoder for a power-constrained system can be made arbitrarily small. If the system operates at a rate $R > C$, the system has a high probability of error, regardless of the choice of the code or decoder. The capacity C of a channel defines the maximum number of bits that can be reliably sent per second over the channel.

CODING PERFORMANCE AND DECODING COMPLEXITY

The *noisy channel coding* theorem states that there exist "good" error-correcting codes for any rate $R < C$ such that the probability of error in an ML decoder is arbitrarily small. However, the proof of this theorem is nonconstructive, which leaves open the problem of the search for specific "good" codes. Also, Shannon assumed exhaustive ML decoding that has a complexity that is proportional to the number of words in the code. It is clear that long codes are required to approach capacity and, therefore, that more practical decoding methods are needed. These problems, left by Shannon, have kept researchers searching for good codes for almost 50 years until the present time.

Gallagher (6) showed that the probability of an error of a "good" block code of length n and rate $R < C$ is bounded exponentially with block length as follows:

$$P(E) \leq e^{nE_b(R)} \tag{5}$$

where what is known as the error exponent $E_b(R)$ is greater than zero for all rates $R < C$. Like Shannon, Gallagher continued to assume a randomly chosen code and an exhaustive ML decoding. The decoding complexity $\hat{K}$ is then of the order of the number of codewords, i.e., $\hat{K} \cong e^{nR}$, and therefore the decreasing of $P(E)$ is bounded only algebraically, with a decoding complexity given by

$$P(E) \leq \hat{K}^{-E_b(R)/R} \tag{6}$$

The exponential error bound given in Eq. (6) for block codes also extends to convolutional codes of memory order m with the form,

$$P(E) \leq e^{-(m+1)nE_c(R)} \tag{7}$$

where $(m + 1)n$ is called the constraint length of the convolutional code, and the convolutional error exponent $E_c(R)$ is greater than zero for all rates $R < C$. Both $E_b(R)$ and $E_c(R)$ are positive functions of R for $R < C$ and are completely determined by the channel characteristics.

It is shown in (2) and (6) that the complexity of an implementation of ML decoding algorithm called the Viterbi algorithm is exponential in the constraint length, i.e., $\hat{K} \cong e^{(m+1)nR}$. Thus, the probability of error is again only an algebraic function of its complexity, as follows:

$$P(E) \leq \hat{K}^{-E_c(R)/R} \tag{8}$$

Both of the bounds Eq. (5) and Eq. (7) imply that an arbitrarily small error probability is achievable for $R < C$ either by increasing the code length n for block codes or by increasing the memory order m for convolutional codes. For codes to be very effective, they must be long in order to average the effects of noise over a large number of symbols. Such a code may have as many as 2^{200} possible codewords and many times the number of possible received words. While an exhaustive ML decoding still conceptually exists, such a decoder is impossible to implement. It is very clear that the key obstacle to an approach to channel capacity is not only in the construction of specific "good" long codes, but also in the problem of its decoding complexity.

Certain simple mathematical constructs enable one to determine the most important properties of "good" codes. Even more importantly, such criteria often make it feasible for the encoding and decoding operations to be implemented in practical electronic equipment. Thus, there are three main aspects of the channel coding problem: (1) to find codes that have the error-correcting ability (this usually demands that the codes be long); (2) a practical method of encoding; and (3) a practical method of making decisions at the receiver, that is, performing the error correction process. Interested readers should refer to the literature (1–9).

BIBLIOGRAPHY

1. S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications,* Englewood Cliffs, NJ: Prentice-Hall, 1983.
2. A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding,* New York: McGraw-Hill, 1979.
3. W. W. Peterson and E. J. Weldon Jr., *Error-Correcting Codes,* 2nd ed., Cambridge, MA: The MIT Press, 1972.
4. F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes,* New York: North-Holland Publishing Company, 1977.
5. E. R. Berlekamp, *Algebraic Coding Theory,* New York: McGraw-Hill, 1968.
6. R. G. Gallager, *Information Theory and Reliable Communications,* New York: Wiley, 1968.
7. G. D. Forney Jr., *Concatenated Codes,* Cambridge, MA: The MIT Press, 1966.
8. R. Blahut, *Theory and Practice of Error Control Codes,* Reading, MA: Addison-Wesley, 1983.
9. G. C. Clark and J. B. Cain, *Error Correction Coding for Digital Communications,* New York: Plenum, 1981.

IRVING S. REED
University of Southern California
XUEMIN CHEN
General Instrument Corporation

CHAOS. See CHAOS, BIFURCATIONS, AND THEIR CONTROL.