Coding

Errors in a digital communication system can be reduced by using

- **Automatic repeat request (ARQ):** When a receiver circuit detects errors in a block of data, it requests that the block be retransmitted. This requires a duplex (two-way) channel.
- **Forward error correction (FEC):** The transmitted data are encoded so that the receiver can correct as well as detect errors.

These methods represent *channel coding* techniques, in contrast to *source coding* which attempts to reduce redundancy in a signal representation.

From a theoretical viewpoint, Shannon's channel capacity theorem states that a finite value for S/N limits only the *rate* of transmission of information. The probability of error can approach zero, however, as long as the information rate is less than the channel capacity. The topic of coding deals with methods of trying to get channels to operate at their capacity.

Coding provides improved performance due to:

- **Redundancy:** Extra bits are added by the coder to accentuate the uniqueness of each message
- **Noise averaging:** The code is designed so that the receiver can average out noise over long timespans T_0, where T_0 becomes large.

Two broad categories of codes are discussed here:

- **Block codes:** k input symbols are mapped directly to n output symbols. Since $n > k$, the code can be selected to provide redundancy, such as **parity bits**, which are used by the decoder to provide some error detection or correction. These codes are called (n, k)-codes, with a **code rate** $R = k/n$.
- **Tree codes:** A tree code is produced by a coder that has memory. For example, the convolutional coder takes k binary symbols at its input and

produces n binary symbols at its output, but here the n output symbols are affected by $v + k$ input symbols. The code rate is $R = k/n$.

1 Block codes

The **Hamming distance** d between two codewords is the number of positions by which they differ. For example, the codewords 110101 and 111001 have a distance of $d = 2$.

If valid codewords are always at least a distance $d = 2$ apart, then it is always possible to detect a single error. If codewords are a distance of $d = 3$ apart, then a single error can be detected and corrected (or two errors detected). In general, some errors can be detected and corrected if $d \geq s + t + 1$, where s is the number of errors that can be detected, and t the number that can be corrected ($s \geq t$). Thus t or fewer errors can be detected and corrected if $d \geq 2t + 1$.

In block coding, codewords are expressed in the form

$$i_1 i_2 i_3 \cdots i_k p_1 p_2 \cdots p_r,$$

where k is the number of information bits, r is the number of parity check bits, and n is the total word length in the (n, k) block code, where $n = k + r$. There are $M = 2^k$ valid codewords. For the parity bits to provide error detection and correction, they must be functionally related to the information bits. This relationship is expressed as follows:

$$\begin{aligned} p_1 &= z_{11} i_1 \oplus z_{12} i_2 \oplus \cdots \oplus z_{1k} i_k \\ p_2 &= z_{21} i_1 \oplus z_{22} i_2 \oplus \cdots \oplus z_{2k} i_k \\ &\vdots \\ p_r &= z_{r1} i_1 \oplus z_{r2} i_2 \oplus \cdots \oplus z_{rk} i_k. \end{aligned}$$

Note that these equations are all expressed in modulo-2 arithmetic, where multiplication is defined as

$$0 \cdot 0 = 0, \quad 1 \cdot 0 = 0, \quad 0 \cdot 1 = 0, \quad 1 \cdot 1 = 1,$$

and addition as

$$0 \oplus 0 = 0, \quad 1 \oplus 0 = 1, \quad 0 \oplus 1 = 1, \quad 1 \oplus 1 = 0.$$

Defining

$$\mathbf{Z} = \begin{pmatrix} z_{11} & z_{12} & \cdots & z_{1k} \\ z_{21} & z_{22} & \cdots & z_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ z_{r1} & z_{r2} & \cdots & z_{rk} \end{pmatrix},$$

the encoded vector

$$\mathbf{c} = [i_1, i_2, \cdots, i_k, p_1, p_2, \cdots, p_r]^T$$

can therefore be obtained from the information word

$$\mathbf{m} = [i_1, i_2, \cdots, i_k]^T$$

using the linear relation $\mathbf{c} = \mathbf{G}^T\mathbf{m}$, where $\mathbf{G}$ is the **generator** matrix

$$\mathbf{G} = \left(\mathbf{I} \;\middle|\; \mathbf{Z}^T \right)$$

The conditions on the parity check bits can equivalently be expressed as follows:

$$\begin{aligned} 0 &= z_{11}i_1 \oplus z_{12}i_2 \oplus \cdots \oplus z_{1k}i_k \oplus p_1 \\ 0 &= z_{21}i_1 \oplus z_{22}i_2 \oplus \cdots \oplus z_{2k}i_k \oplus p_2 \\ &\vdots \\ 0 &= z_{r1}i_1 \oplus z_{r2}i_2 \oplus \cdots \oplus z_{rk}i_k \oplus p_r. \end{aligned}$$

A $r \times n$ parity check matrix

$$\mathbf{H} = \begin{pmatrix} z_{11} & z_{12} & \cdots & z_{1k} & 1 & 0 & \cdots & 0 \\ z_{21} & z_{22} & \cdots & z_{2k} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ z_{r1} & z_{r2} & \cdots & z_{rk} & 0 & 0 & \cdots & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{Z} & \bigg| & \mathbf{I} \end{pmatrix}.$$

can therefore be defined, and for a codeword **c** to be valid it must be true that

$$\mathbf{Hc} = \mathbf{0}.$$

The function of the receiver is to try to recover the original information word **m** from a received codeword $\tilde{\mathbf{c}}$. The tilde denotes that the received vector may have some errors, caused by noise in the channel and other impairments. The first role of the receiver is therefore to determine if there are any errors in $\tilde{\mathbf{c}}$, and if so to correct them to produce **c**. This is done by evaluating the **syndrome**

$$\mathbf{s} = \mathbf{H}\tilde{\mathbf{c}}.$$

If $\mathbf{s} = \mathbf{0}$, no errors are detected. However, if $\mathbf{s} \neq 0$ then an error has occurred.

To find out how to correct a single error, suppose the received codeword $\tilde{\mathbf{c}}$ has an error in the jth position. Then

$$\tilde{\mathbf{c}} = \mathbf{c} \oplus \mathbf{e},$$

where **e** is an error vector $\mathbf{e} = [0, \cdots, 1, \cdots, 0]^T$, with the 1 in the jth position. In this case the syndrome is

$$\mathbf{s} = \mathbf{H}\tilde{\mathbf{c}} = \mathbf{H}(\mathbf{c} \oplus \mathbf{e}) = \mathbf{He}.$$

This is exactly the jth column of the **H** matrix. The bit in error can therefore be determined by comparing the syndrome to the columns of **H**. Once the location of the incorrect bit is determined, the received codeword can be corrected.

Finding good long codes for error correction is not trivial. A **Hamming code** is a block code that has a Hamming distance of $d = 3$, so a single error can be detected and corrected. For this code, the **H** matrix is determined by choosing the **Z** matrix such that the columns of **H** contain all the possible r-bit binary words except for the all-zero vector. However, only certain (n, k) codes are allowable, namely those where

$$(n, k) = (2^m - 1, 2^m - 1 - m)$$

with m an integer and $m \geq 3$. Thus some allowable codes are (7, 4), (15, 11), (63, 57), and (127, 120). As m becomes large the code rate tends towards 1, and the information throughput efficiency of the code becomes high.

Example:

Consider the (7, 4) code with parity check matrix

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

This represents a Hamming code, since the columns of **H** are the binary-coded decimal numbers 1 through 7.

The generator matrix for this code can be shown to be

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

For $\mathbf{m} = [1, 0, 1, 1]^T$ the transmitted codeword is therefore

$$\mathbf{c} = \mathbf{G}^T \mathbf{m} = [1, 0, 1, 1, 0, 1, 0]^T.$$

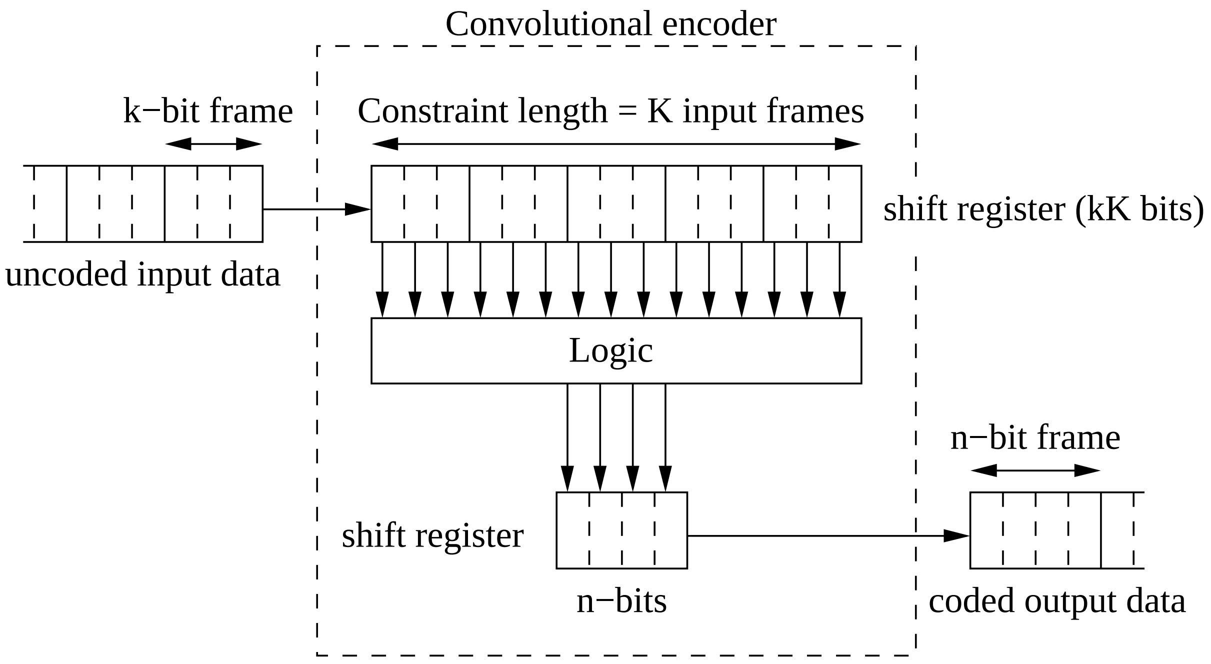
Suppose now that due to noise the received codeword is $\tilde{\mathbf{c}} = [1, 0, 1, 0, 0, 1, 0]^T$. The syndrome is

$$\mathbf{s} = \mathbf{H}\tilde{\mathbf{c}} = [1, 1, 1]^T$$

which is the 4th column of **H**. We therefore conclude that the 4th element of $\tilde{\mathbf{c}}$ is in error, and that the received codeword should in fact be
$[1, 0, 1, 1, 0, 1, 0]^T$.

2 Convolutional codes

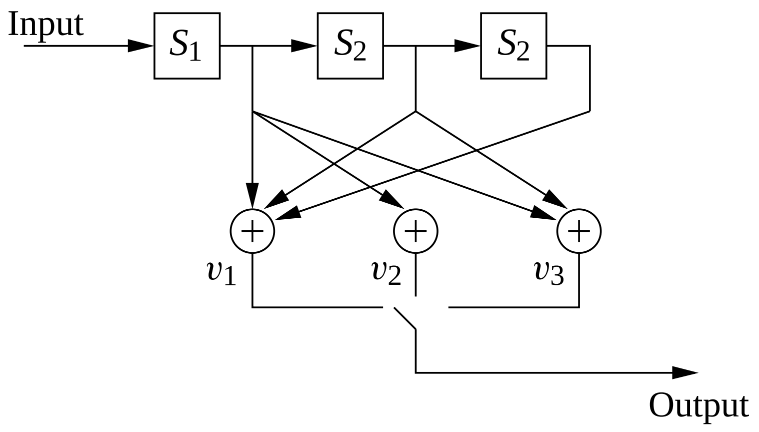
In a convolutional encoder, k bits (one input frame) are shifted in and concurrently n bits (one output frame) are shifted out during each encoding cycle.



Every k-bit input frame therefore results in an n-bit output frame, with redundancy provided by making $n > k$. Also, memory is provided in the coder, since the output frame depends on the previous K input frames, where $K > 1$. The code rate is $R = k/n$. In the example above, $k = 3$, $n = 4$, $K = 5$, and $R = 3/4$.

Depending on the particular convolutional code, data from the kK stages of the shift register are added (modulo-2) and used to set the bits in the output register.

A rate 1/3 convolutional encoder is shown below:

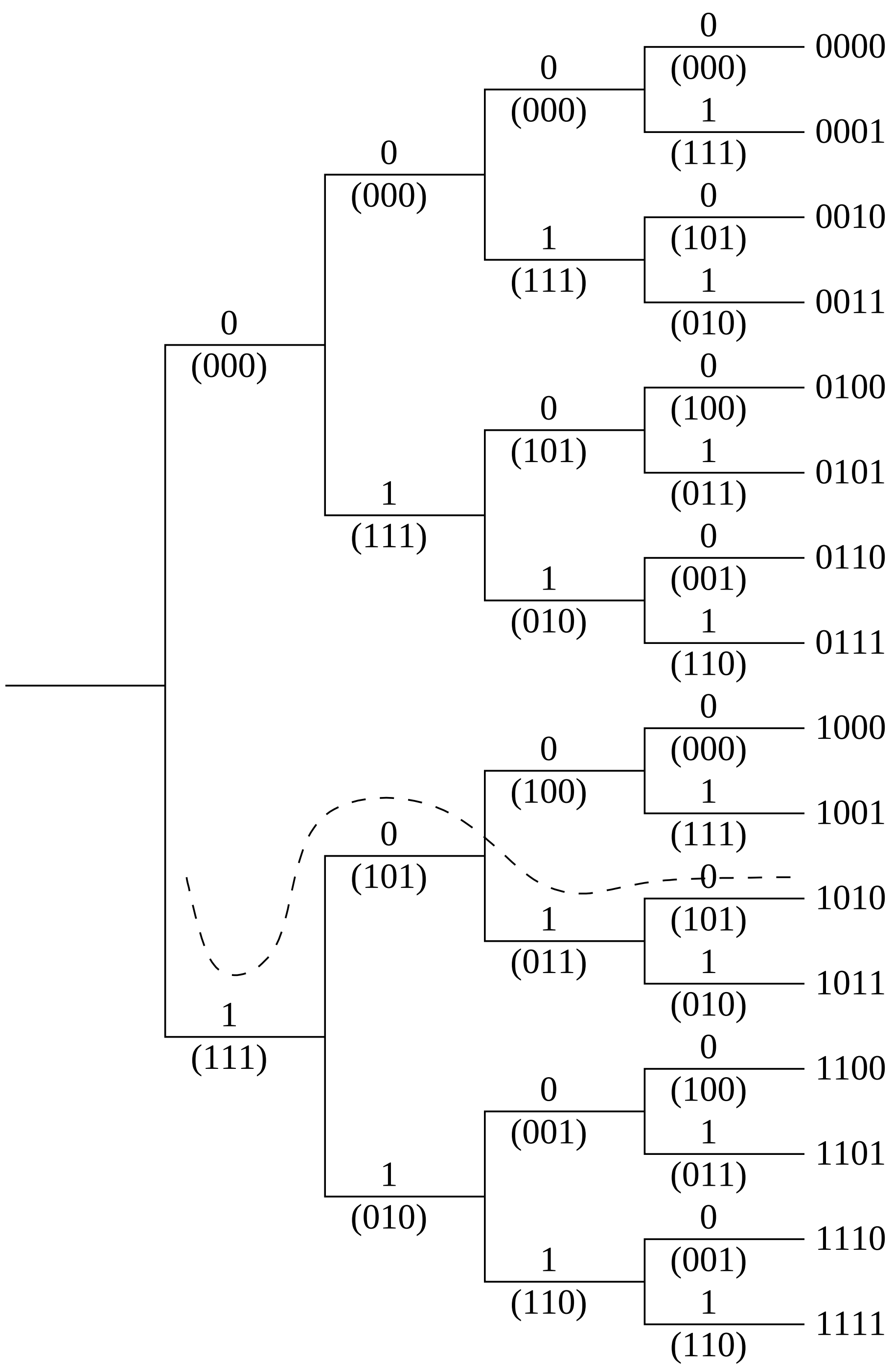


Here $k = 1$, $K = 3$, and $n = 3$. A commutator with 3 inputs performs the function of the shift register — the convolutional code is generated by inputting a bit of data and then rotating the commutator to produce $n = 3$ output bits. The encoder shown implements the code

$$v_1 = S_1 \oplus S_2 \oplus S_3$$
$$v_2 = S_1$$
$$v_3 = S_1 \oplus S_2.$$

Thus the input sequence 101001 results in the output sequence 111101011101100111.

Convolutional codes can be decoded by tree-searching techniques. A portion of the code tree for the encoder above is



If the input to the encoder was 1010, the output would be 111101011101. Decoding this sequence involves following the dashed line in the figure — at each fork in the tree we follow the path closest in Hamming distance to the input sequence. Thus it is evident that the input sequence 110101011111 will be decoded as 1010, indicating an error in the third and eleventh positions of the input sequence.

For N information symbols the tree code requires 2^N branches, which becomes impractical in terms of storage requirements when N is large. The most popular technique for decoding uses the Viterbi algorithm.