

Binary Linear Hamming Codes

A Hamming code is a binary linear error-correcting code that can correct single-bit errors. For each integer $m > 2$, there is a $[2^m - 1, 2^m - m - 1, 3]$ Hamming code. This implies that all Hamming codes have a minimum distance of 3, which means that the code can detect and correct a single error and detect double-bit errors. By including an extra parity bit, it is possible to increase the minimum distance of the Hamming code to 4. This gives the code the ability to detect up to 3 errors but not correct any. Because of the simplicity of Hamming codes, they are popular in computer memory systems (RAM, Random Access Memory), where they are known as *SECDED* ("Single Error Correction, Double Error Detection"). Particularly popular code is the $[72, 64]$ code, a truncated $[127, 120]$ Hamming code plus an additional parity bit.

The parity-check matrix of a Hamming code is constructed by listing all rows of length m , where each row is a binary representation of a number from 1 to $2^m - 1$ (not particularly in ascending or descending order).

Although any number of algorithms can be created, the following general algorithm positions the parity bits at powers of two to ease calculation of which bit was flipped upon detection of incorrect parity.

1. All bit positions that are powers of two are used as parity bits. (positions 1, 2, 4, 8, 16, 32, 64, etc.)
2. All other bit positions are for the message to be encoded. (positions 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, etc.)
3. Each parity bit calculates the parity for some of the bits in the code word. The position of the parity bit determines the sequence of bits that it alternately checks and skips.

♣ **[7,4,3] Hamming Codes.** In a 7-bit message, there are seven possible single bit errors, so three error control bits could potentially specify not only that an error occurred but also which bit caused the error. In Shannon's paper, the following algorithm (due to Richard Hamming) describes a $[7, 4, 3]$ binary Hamming code together with its decoding scheme:

Let $v = (v_1, v_2, v_3, v_4, v_5, v_6, v_7) \in \mathbb{K}^7$, where v_3, v_5, v_6, v_7 are message bits and v_1, v_2, v_4 are parity bits calculated as follows:

1. v_1 is chosen to make $\alpha = v_1 + v_3 + v_5 + v_7 \equiv 0 \pmod{2}$.
2. v_2 is chosen to make $\beta = v_2 + v_3 + v_6 + v_7 \equiv 0 \pmod{2}$.
3. v_4 is chosen to make $\gamma = v_4 + v_5 + v_6 + v_7 \equiv 0 \pmod{2}$.

Notice that α and β share v_3 and v_7 , α and γ share v_5 and v_7 , and β and γ share v_6 and v_7 .

When a word is received, then α , β , and γ are calculated; zero indicates no error occurred and one gives the v_i that is incorrect.

Here is a systematic generating matrix of the [7, 4, 3] Hamming code using the above algorithm:

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

It generates the following Hamming code:

$$C = \left\{ \begin{array}{cccccccc} 0000000 & 1110000 & 1001100 & 0101010 & 1101001 & 0111100 & 1100110 & 1000011 \\ 0011001 & 1011010 & 0100101 & 0010110 & 1010101 & 0110011 & 0001111 & 1111111 \end{array} \right\}$$

Here is how to correct errors for word received by any systematic Hamming code using the above algorithm, assuming that only one error bit occurred:

(α, β, γ)	$error \longleftrightarrow bit$
$1 \longleftrightarrow (1, 0, 0)$	$1000000 \longleftrightarrow 1$
$2 \longleftrightarrow (0, 1, 0)$	$0100000 \longleftrightarrow 2$
$4 \longleftrightarrow (0, 0, 1)$	$0001000 \longleftrightarrow 4$
$3 \longleftrightarrow (1, 1, 0)$	$0010000 \longleftrightarrow 3$
$5 \longleftrightarrow (1, 0, 1)$	$0000100 \longleftrightarrow 5$
$6 \longleftrightarrow (0, 1, 1)$	$0000010 \longleftrightarrow 6$
$7 \longleftrightarrow (1, 1, 1)$	$0000001 \longleftrightarrow 7$

Example. In all that follows, we can see how the above decoding scheme detects, corrects, and decodes words:

<i>Error :</i>	<i>Zero Error</i>	<i>One Error</i>	<i>Two Errors</i>	<i>Three Errors</i>	<i>Four Errors</i>
<i>Sent :</i>	$v_0 = 0001111$	$v_1 = 0011001$	$v_2 = 1010101$	$v_3 = 0111100$	$v_4 = 1001100$
<i>Received :</i>	$w_0 = 0001111$	$w_1 = 0011011$	$w_2 = 0011101$	$w_3 = 1001100$	$w_4 = 1000011$
$(\alpha \beta \gamma) :$	$(0 \ 0 \ 0) \longleftrightarrow 0$	$(0 \ 1 \ 1) \longleftrightarrow 6$	$(1 \ 0 \ 1) \longleftrightarrow 5$	$(0 \ 0 \ 0) \longleftrightarrow 0$	$(0 \ 0 \ 0) \longleftrightarrow 0$
<i>Decoded :</i>	$v'_0 = 0001111$	$v'_1 = 0011001$	$v'_2 = 0011001$	$v'_3 = 1001100$	$v'_4 = 1000011$
<i>Detected :</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>No</i>
<i>Corrected :</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>No</i>

Notice that the received word w_2 with two error-bits was decoded as v'_2 , but $v'_2 \notin C$, so we know that v'_2 is not the right word and it is impossible for us to guess the right codeword.

An interesting fact about this algorithm is that a received word can be decoded without the use of a parity check matrix or coset leaders.

One possibility for a parity-check matrix H_s and a generator matrix G_s for a $[7,4,3]$ Hamming code is:

$$H_s = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad G_s = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

Notice that G_s is in standard form. By permuting columns of G_s or rows of H_s , we obtain systematic Hamming code. We shall see cyclic Hamming codes with non-systematic generator.

♣ **Binary Linear Cyclic Hamming Codes.** The $[7, 4, 3]$ binary cyclic code generated by the polynomial $g(x) = 1 + x + x^3$ is a Hamming code:

$$H_c = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \quad \text{with} \quad G_c = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

This generator matrix is not in a systematic form and does not use the previous algorithm. The encoding and decoding algorithms for Hamming cyclic codes are based on multiplication and division of polynomials which is discussed in the binary cyclic code section.

Notice that

$$[2^3 - 1, 2^3 - 3 - 1, 3] = [15, 11, 3];$$

therefore the binary linear cyclic code in \mathbb{K}^{15} , generated by the polynomial $g(x) = 1 + x + x^4$ must be a Hamming code.

♣ **Extended Hamming Codes.** An extension of a binary Hamming code results from adding at the beginning or at the end of each codeword a new digit that checks the parity of the codeword. Therefore, every word in an Extended Hamming code has an even number of ones. This way, the minimum distance of the Hamming code is increased

from 3 to 4. This gives the code the ability to detect and correct a single error and it could also be used to detect up to 3 errors but not correct any.

It is important to note that by extending a code, we increase the length of the codewords but we keep the dimension of the code.

By adding a parity bit at the end of our first $[7,4,3]$ Hamming code, we obtain an $[8,4,4]$ Extended code which changes the generator matrix G into \hat{G} :

$$G = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \quad [7,4,3] \text{ Hamming Code}$$

$$\hat{G} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \quad [8,4,4] \text{ Extended Hamming Code}$$

The following Extended Hamming code generated by \hat{G} still has sixteen codewords:

$$\hat{C} = \left\{ \begin{array}{cccccccc} 00000000 & 11100001 & 10011001 & 01010101 & 1101001 & 01111000 & 11001100 & 10000111 \\ 00110011 & 10110100 & 01001011 & 00101101 & 10101010 & 01100110 & 00011110 & 11111111 \end{array} \right\}$$

Example. In all that follows, we can see how the above decoding scheme detects, corrects, and decodes words:

<i>Error :</i>	<i>Zero Error</i>	<i>One Error</i>	<i>Two Errors</i>	<i>Three Errors</i>	<i>Four Errors</i>
<i>Sent :</i>	$v_0 = 00011110$	$v_1 = 00110011$	$v_2 = 10101010$	$v_3 = 01111000$	$v_4 = 10011001$
<i>Received :</i>	$w_0 = 00011110$	$w_1 = 00110110$	$w_2 = 00111010$	$w_3 = 10011000$	$w_4 = 10000111$
$(\alpha \beta \gamma) :$	$(0 \ 0 \ 0) \longleftrightarrow 0$	$(0 \ 1 \ 1) \longleftrightarrow 6$	$(1 \ 0 \ 1) \longleftrightarrow 5$	$(0 \ 0 \ 0) \longleftrightarrow 0$	$(0 \ 0 \ 0) \longleftrightarrow 0$
<i>Decoded :</i>	$v'_0 = 00011110$	$v'_1 = 00110011$	$v'_2 = 00110011$	$v'_3 = 10011001$	$v'_4 = 10000111$
<i>Detected :</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>
<i>Corrected :</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>No</i>

Notice that the received word w_2 with two error-bits was corrected by changing the fifth bit, based on the value of $(\alpha \beta \gamma)$; the parity bit was also changed to one. Unfortunately, the resulting decoded word v'_2 which is a codeword is not the same as the codeword v_2 which was sent. In the case of v'_3 , the error was detected, only because of the wrong parity bit, but $v'_3 \neq v_3$. Finally, the received word w_4 with an even number of ones showed no error; the evaluation of $(\alpha \beta \gamma)$ produced $(0 \ 0 \ 0)$; but as we can see $v'_4 = w_4 \neq v_4$.