Weight Distribution of Linear Error Correcting Codes

Chaos Golubitsky

Abstract

In this paper, i will discuss a class of error-correcting codes known as BCH codes. I will give a general description of these codes, and name some properties of error-correcting codes in general, and BCH codes specifically, in an attempt to justify the usefulness of this class of codes. I will then discuss an interesting theorem about the weight distribution of codes which holds for large BCH codes.

1 Introduction to Linear Codes

There are many operations for which it is necessary to transmit information over some channel. The desired outcome of such a transmission is that the information obtained at the receiving end is the same as the information which was placed in the channel at the sending end. In reality, this is not always the case, since it is possible for the channel to mistransmit some of the information.

An error-correcting code is a set of rules which can be used to encode a message so that the original message can be retrieved even if a small number of errors occur in transmission. In this paper, i focus exclusively on codes for *binary symmetric* channels. "Binary" means that these channels are used to transmit strings consisting of the symbols 0 and 1, and "symmetric" means that the probability p that a 0 is incorrectly transmitted as a 1 is the same as the probability that a 1 is incorrectly transmitted as a 0. We also assume $p < .5$, since if $p > .5$, we can simply switch 0s and 1s on the receiving end, making the error probability $1 - p$. If $p = .5$, then the channel is useless.

1.1 Definition of a Code

A code works by breaking the information to be transmitted into *messages* of length k. Each message $u = u_1 \dots u_k$ is translated into a *codeword* $x = x_1 \dots x_n$ of length n. The codeword x is then passed through the channel, and, if all goes well, is retranslated to the correct message at the other end. The number k is referred to as the dimension of the code, n is the length of the code, and the code is called an $[n, k]$ code.

Thus, a binary code is simply the set of the 2^k possible message symbols u and the codeword x which corresponds to each. There is no inherent restriction on the method used to associate u and x. However, if some restrictions are placed on the code, making it a linear code, then some nice properties result.

An $[n, k]$ linear code $\mathcal{C}$ is defined by an $(n-k) \times n$ nonzero matrix H, called the parity check matrix. Given this matrix, the codewords of $\mathcal{C}$ are all vectors x such that $Hx^{\mathrm{t}} = 0$. Note that H can be rewritten as $H = [A|I_{n-k}]$, where A is some $(n-k) \times k$ matrix. This form has the property that the first k symbols of each codeword x are the message symbols $u_1 \dots u_k$. Thus, given any codeword, we can find the associated message by simply removing the last $n - k$ symbols.

In order to obtain codewords from message symbols, we compute the *generator matrix* of $\mathcal{C}$, which is the $k \times n$ matrix $G = [I_k| - A^{\mathrm{t}}] = [I_k|A^{\mathrm{t}}]$ (since $A = -A$ for binary matrices.) Then, for any message word u, the associated codeword is $x = uG$. When x is sent through the channel, the word $y = y_1 \dots y_n$ is received. If $Hy = 0$, then y is a codeword, and we assume that $y = x$. Note that $Hx = 0$ and $x = uG$ imply $HG^t = 0$.

Suppose $Hy \neq 0$. Then there has been an error in transmission, and we must try to find the intended codeword. The simplest strategy in this case, called *maximum likelihood decoding*, is based on the idea that, since $p < .5$, it is more likely that a single received symbol y_i is correct than that y_i is wrong. Thus, the most likely codeword x to have been transmitted is the one for which the fewest errors need to have occurred in order for y to have been received.

The *Hamming distance* between two vectors, denoted $\text{dist}(x, y)$, is the number of i such that $x_i \neq y_i$. The *weight* of a vector, $\text{wt}(x)$, is the number of nonzero x_i. Obviously, $\text{dist}(x, y) = \text{wt}(x - y)$. Maximum likelihood decoding computes the distance between the received word y and each codeword x, and decodes y as the message word corresponding to the x for which $\text{dist}(x, y)$ is smallest. Maximum likelihood decoding can be implemented by simply comparing the received word y to every x in the code, but this is not very efficient. I will introduce another implementation later in this section.

An example of a linear code which may be familiar is the binary even-weight code. This code adds a 1 or 0 to the end of each message word, such that the codeword has even weight. For instance, take the even-weight code of length $n = 5$. It is a $[5, 4]$ code with parity check matrix $H = [\ 1 \quad 1 \quad 1 \quad 1 \quad 1\ ]$. Thus, the generator matrix is

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

So the message word $u = 1011$ would be encoded as $x = 10111$ for transmission. If a single error in transmission occurred and the received vector were $y = 10101$, then the decoder would know an error had occurred, but would be unable to determine which symbol was incorrect.

1.2 Usefulness of a Code

Not all codes are equally useful. There are two main qualities a good code should have. First, the code should be *efficient*. For linear codes, efficiency is defined as the ratio $\frac{k}{n}$ of message symbols to code symbols, and we would like this value to be as high as possible. Second, the code should be able to decode transmissions with as few decoding errors as possible. (A decoding error occurs when maximum likelihood decoding returns the wrong codeword.)

A code is called t-error-correcting if any received word which had t or fewer errors in transmission is correctly decoded by some maximum likelihood decoding scheme. For a good code, t should be large relative to the length n of the code.

For a given code $\mathcal{C}$, t can be determined by examining the *minimum distance* d of the code. We define $d = \min(\text{dist}(x, y))$, $x, y \in \mathcal{C}$. One of the properties of linear codes is that if x and y are in $\mathcal{C}$, then $x + y \in \mathcal{C}$, and $cx \in \mathcal{C}$ for any constant c. (This property is clear from linear algebra and the construction of a linear code.) Thus, the word $x - y$ is a codeword with weight $\text{wt}(x - y) = \text{dist}(x, y)$, and we can write $d = \min(\text{wt}(x))$, $x \in \mathcal{C}$, $x \neq 0$. An $[n, k]$ code with minimum distance d is denoted an $[n, k, d]$ code.

Theorem 1 *A code with minimum distance d can correct $\lfloor \frac{1}{2}(d-1) \rfloor$ errors and can detect $\lfloor \frac{d}{2} \rfloor$ errors.*

Since the number of errors corrected by a code is a function of minimum distance, it is possible to create a t-error-correcting code by designing a code which has minimum distance $2t + 1$. This idea will be used in the construction of BCH codes.

Returning to the previous example of the even-weight code: all codewords have even weight, so the minimum distance is at least $d = 2$, and it is exactly $d = 2$ because, for example, 11000 is a codeword. Thus, by Theorem 1, the even-weight code cannot correct any errors, though it can detect one error.

The even-weight code is an extreme example of a very efficient code which does not have good error-correcting capability. An example in the opposite extreme is the binary repetition code, which is an $[n, 1, n]$ code with two codewords of length n. The message word 0 is encoded as $0 \cdots 0$, and the message word 1 as $1 \cdots 1$. Thus, the generator matrix for the $[4, 1, 4]$ repetition code is simply $G = [\ 1 \quad 1 \quad 1 \quad 1\ ]$. This code can correct $\lfloor \frac{1}{2}(n-1) \rfloor$ errors, but has very poor efficiency.

1.3 Faster Decoding Methods

If error-correcting codes are to be used in real applications, it is almost always important that they provide quick decoding of messages. BCH codes are important because they have this property, when used with appropriate decoding methods. In the remainder of this section, i will introduce an implementation of maximum likelihood decoding which is faster than the guess-and-check method. In the next section, i will describe the Hamming and BCH codes which benefit from this new method.

This method makes direct use of the fact that $\mathcal{C}$ is a linear code. Since $\mathcal{C}$ is closed under addition and scalar multiplication, we can consider it to be a subgroup of $\mathbf{Z}_2^n$, the group consisting of all binary n-tuples. Then, we can define cosets in $\mathcal{C}$ of elements $a \in Z_2^n$, where the coset

$$a + \mathcal{C} = \{a + x | x \in \mathcal{C}\}.$$

Then $b \in a + \mathcal{C}$ if and only if $a - b \in \mathcal{C}$. This property shows how we can use cosets for decoding. Suppose at least one transmission error occurs, and some received word y is not in $\mathcal{C}$. Then there is some vector e such that $x = y - e \in \mathcal{C}$. In this case, e is the transmission error that occurred if x was sent. Now, we can choose e to be any element of the coset $y + \mathcal{C}$. However, maximum likelihood decoding dictates that we choose the smallest possible number of errors. So e should be the element in $y + \mathcal{C}$ of least weight. This element is called the *coset leader* of $y + \mathcal{C}$.

So, we prepare for decoding by finding the $\frac{2^n}{|\mathcal{C}|}$ coset leaders in $\mathcal{C}$. Then, whenever a vector y is received, we assume the transmission error to be the coset leader of $y + \mathcal{C}$, and decode the codeword $x = y - e$. (If the coset leader is $e = 0$, then $x = y$ and we assume no error occurred.)

For instance, suppose $\mathcal{C}$ is the $[6, 3, 3]$ code with

$$H = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Then the 8 codewords are 000000, 001110, 010101, 011011, 100011, 101101, 110110, and 111000. There are $\frac{64}{8} = 8$ cosets. The coset leaders of the first seven cosets are 000000 (this is the coset corresponding to $\mathcal{C}$ itself), and the vectors of length 6 of weight 1. These six vectors are all in distinct cosets, since if two of them were in the same coset, their sum would be in the code, and no codeword has weight less than 3.

The eighth coset consists of the eight words 100100, 101010, 110001, 111111, 000111, 001001, 010010, and 011100. This coset contains no words of weight less than 2, and three words of weight 2, so there is

no clear coset leader. The decoder can either choose of one 100100, 001001, or 010010 at random as the coset leader (likely causing a decoding error), or simply return an error if a word in this coset is received.

Once we have identified the cosets, there is an even faster method of finding which coset contains a given vector y. Compute the vector $S = Hy^{\mathrm{t}}$, where H is the parity check matrix of the code. This vector is called the *syndrome* of y. By definition of a linear code, if $S = 0$, then $y \in \mathcal{C}$. If $S \neq 0$, then $S = Hy^{\mathrm{t}} = Hx^{\mathrm{t}} + He^{\mathrm{t}} = He^{\mathrm{t}}$, so every element of a coset has the same syndrome.

Now, e is the vector which is 1 at any location where an error occurred and 0 otherwise. So $S = He^{\mathrm{t}}$ is equal to the sum of the columns of H where errors occurred. If S is the sum of t or fewer columns of H (for a t-error-correcting code $\mathcal{C}$), then we assume the coset leader of $y + \mathcal{C}$ is the vector e with 1 in those locations, and decode $x = y - e$ accordingly. If S is not the sum of any t or fewer columns, then the coset leader of $y + \mathcal{C}$ has weight greater than t, and y cannot be decoded.

Using the example from earlier in this section, if the vector $x = 100011$ is received, then the syndrome is $S = Hx^t = 0$ because x is a codeword. But if the vector $y = 011110$ is received, then the syndrome is $S = Hy^t = 101$. This is the same as the second column of H, so the decoder concludes that an error occurred in the second column and interprets the received vector as the codeword 001110. If the vector $z = 010010$ is received, then the syndrome is $S = Hz^t = 111$, which is not a column of H, so more than one error must have occurred. Since $\mathcal{C}$ is a single-error correcting code, the decoder reports failure.

2 Introduction to Specific Codes

BCH codes are a class of codes designed to take advantage of this decoding method. That is, for a given BCH code, any received word whose number of transmission errors falls within the error correction bounds can be transformed to the correct codeword by finding the syndrome and doing a minimal amount of additional algebra.

The algebraic justification for BCH codes is best understood by first introducing Hamming codes. These are a special case of BCH codes designed to correct one error, but are much simpler in construction.

2.1 Hamming Codes

The binary Hamming code $\mathcal{H}_r$ is a code of length $n = 2^r - 1$ for some integer $r \geq 2$. The parity check matrix H of $\mathcal{H}_r$ has columns consisting of all nonzero binary vectors of length r. Since H is an $r \times (2^r - 1)$ matrix, the dimension of this code must be $k = 2^r - 1 - r$.

Now, suppose the word y is received in transmission. Then the syndrome $S = Hy^{\mathrm{t}}$ is some vector of length r. But any vector of length r is one of the columns of H, by definition, so if $S = H_i$ (the ith column of H), then the coset leader of y is e_i, the vector with a 1 at location i and zeroes otherwise. So $\mathcal{H}_r$ is a single-error-correcting code. In fact, the minimum distance of any Hamming code is $d = 3$.

If more than one transmission error occurs, then the decoder will return an incorrect codeword, rather than signaling failure. This is because Hamming codes are *perfect* codes — they correctly decode all vectors with $t = 1$ or fewer errors, and incorrectly decode all vectors with more than t errors. Note that for large values of r, the efficiency $\frac{2^r-1-r}{2^r-1}$ of $\mathcal{H}_r$ is high, but $\mathcal{H}_r$ still corrects only a single error. Thus, Hamming codes are most useful when the probability p of a single error is very low.

2.2 Cyclic Codes

By viewing Hamming codes as a subset of a larger class of codes, called cyclic codes, it is possible to extend the idea of Hamming codes to create codes which correct multiple errors while remaining easy to decode.

By definition, an $[n, k, d]$ code $\mathcal{C}$ is a cyclic code if $\mathcal{C}$ is an ideal of the ring R_n of all binary words of length n. An ideal is a set $\mathcal{C} \subset R_n$ such that $rc \in \mathcal{C}$ whenever $c \in \mathcal{C}$ and $r \in R_n$.

However, this definition involves the product rc of binary words, and the view of codewords developed in Section 1 gave no algorithm for multiplying two codewords. In order to treat R_n as a ring, we need to introduce such an algorithm. This is done by associating a polynomial with each binary vector, treating the leftmost bit as least significant. So, for instance, in R_4, 1101 is represented by the polynomial $1 + x + x^3$.

Then two vectors can be multiplied as polynomials. However,

$$1101 \cdot 0110 = (1 + x + x^3)(x + x^2) = x + x^3 + x^4 + x^5,$$

which cannot be represented as a binary vector of length 4. So, instead, we let $F[x]$ be the field of all binary polynomials of any length, and set R_n to be the residue field $R_n = F[x]/(x^n - 1)$. (Since $F[x]$ is a binary field, $x^n - 1 = x^n + 1$.)

In our example, $x^4 + 1 = 0$, so we can compute

$$(x + x^3 + x^4 + x^5)/(x^4 + 1) = 1 + x^3 = 1001 \in R_4.$$

We can now return to the code $\mathcal{C}$, which is defined to be an ideal of this ring. Since R_n is a binary ring, the definition of an ideal over R_n is simpler than previously stated. To see this, note that any polynomial $r(x) \in R_n$ is a sum of x^j for some values $0 \leq j < n$. As long as $\mathcal{C}$ is a linear subspace of R_n (which any linear code is), it is closed under addition. So, we need only verify that $x^j c(x) \in \mathcal{C}$ for every codeword $c(x) \in \mathcal{C}$ and every j. But $x^j = x \cdot x^{j-1}$, so, by induction, $\mathcal{C}$ is an ideal if $xc(x) \in \mathcal{C}$.

Now, since $x^n = 1$, $xc(x)$ is equivalent to a *cyclic shift* of $c(x)$ such that if

$$c(x) = c_0 \dots c_{n-1} = c_0 + \dots + c_{n-1}x^{n-1},$$

then

$$xc(x) = c_{n-1}c_0 \dots c_{n-2} = c_{n-1} + c_0 x + \dots + c_{n-2}x^{n-1}.$$

Thus, a code is cyclic if and only if any cyclic shift of a codeword is also a codeword.

We now need to know how to generate cyclic codes. In fact, every binary cyclic code is a primitive ideal, that is, an ideal generated by some polynomial $g(x)$. We write this as $\mathcal{C} = \langle g(x) \rangle$. This is shown by the following multi-part theorem, which is easily proved by definition of ideals in R_n.

Theorem 2 *If $\mathcal{C} \in R_n$ is a nonzero ideal, then:*

(a) *$\mathcal{C}$ contains a unique polynomial $g(x)$ of minimal degree r.*

(b) *This polynomial $g(x)$ generates $\mathcal{C}$.*

(c) *$g(x)$ is a factor of $x^n + 1$.*

(d) *Any codeword $c(x) \in \mathcal{C}$ can be written uniquely as $c(x) = f(x)g(x)$, where the degree of $f(x)$ is less than $n - r$.*

(e) *Given $g(x)$, the generator matrix of the code is the $(n - r) \times n$ matrix*

$$G = \begin{bmatrix} g(x) \\ xg(x) \\ \vdots \\ x^{n-r-1}g(x) \end{bmatrix}.$$

So, if we have a polynomial $g(x)$ which is a factor of $x^n + 1$ of degree $r < n$, then we can generate an $[n, n - r]$ cyclic code. But in order to generate specific kinds of codes, we need the concept of minimal polynomials.

2.3 Minimal Polynomials and Hamming Codes

Thus far, we have considered elements of the ring R_n, which consists of all binary polynomials of degree less than n. Now, we consider another, smaller field.

The field $GF(2^r)$ is the unique finite field with 2^r elements. It is defined similarly to R_n, but R_n is the residue after division by x^n+1, while $GF(2^r)$ consists of polynomials modulo some *irreducible* polynomial $\pi(x)$ of degree r. Clearly, this field contains all binary polynomials of degree less than r.

Now, any element $\beta \in GF(2^r)$ satisfies the identity $\beta^{2^r-1} = 1$. An element α is primitive if it has order $2^r - 1$, that is, if $\alpha^m \neq 1$ for any positive $m < 2^r - 1$. If α is a primitive element, then any nonzero element of $GF(2^r)$ can be written as α^j for some j. The field $GF(2^r)$ has the property that it always contains a primitive element.

Given an element $\beta \in GF(2^r)$, there is some binary polynomial $M(x)$ of lowest degree such that $M(\beta) = 0$. In fact, it is always the case that $\deg(M(x)) \leq r$. If $\alpha \in GF(2^r)$ is a primitive element, then its minimal polynomial has degree r and is called a primitive polynomial. In fact, every primitive element of $GF(2^r)$ has the same minimal polynomial, and this polynomial is the same $\pi(x)$ which was used to construct $GF(2^r)$.

We can now construct the Hamming code $\mathcal{H}_r$ as a cyclic code. Find a primitive element of $GF(2^r)$ and call it α. Then the first $2^r - 1$ powers of α are distinct, and represent all nonzero binary vectors of length r. So the parity check matrix of $\mathcal{H}_r$ can be written $H = (1, \alpha, \alpha^2, \ldots, \alpha^{2^r-2})$. Alternatively, the generator polynomial $g(x)$ of $\mathcal{H}_r$ is the minimal polynomial of α, which is $\pi(x)$, so $\mathcal{H}_r$ can be defined by its generator matrix G in the manner suggested by Theorem 2.

As an example, take the case $r = 3$. Define the field $GF(2^3)$ by setting the irreducible polynomial $\pi(x) = 1 + x^2 + x^3$ equal to zero, and letting $\alpha = 010$ be a root of $\pi(x)$. Then the parity check matrix of $\mathcal{H}_3$ is:

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix},$$

which is the matrix containing all nonzero vectors of length 3, as desired.

Using the alternative construction, note that the primitive polynomial of α is simply $\pi(x) = 1 + x^2 + x^3 = 1011$. So the generator polynomial of $\mathcal{H}_3$ can be written:

$$G = \begin{bmatrix} \pi(x) \\ x\pi(x) \\ x^2\pi(x) \\ x^3\pi(x) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

It is sufficient to use one of these methods, since either H or G can be derived from the other. Matrix multiplication confirms that $HG^t = GH^t = 0$, so the two methods have produced consistent results.

2.4 Construction of BCH Codes

We can now proceed with the construction of multiple-error-correcting cyclic codes. The idea of Hamming codes is that each column of the parity check matrix H is distinct (and together they represent all

possibilities), so it is possible to immediately identify the position of a single error — just compare the syndrome to the columns of the matrix.

BCH codes expand this idea. Abstractly, decoding a transmission sent with a t-error correcting BCH code works as follows: suppose t or fewer errors occur in the transmission of a given vector x. Then the syndrome S of the received vector y will be a sum of the columns of H in which the errors occurred. (This is true for any code.) In a BCH code, it is possible to find out which columns would have S as a sum by solving t linear equations. If there is no solution, then more than t errors occurred, and the codeword cannot be decoded. (That is, BCH codes are not perfect.)

The expansion of a Hamming code into a 2-error correcting BCH code would proceed as follows: given a parity check matrix H of the Hamming code, add r more rows to H (so H is now a $2r \times (2^r - 1)$ matrix). The top half of the matrix still consists of the elements $1, \alpha, \ldots, \alpha^{2m-2}$, while the bottom half is $f(1), f(\alpha), \ldots, f(\alpha^{2m-2})$ for some function $f(x)$. This $f(x)$ is chosen so that we can solve the equation

$$\begin{pmatrix} \alpha^i + \alpha^j \\ f(\alpha^i) + f(\alpha^j) \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = S$$

uniquely for a given z_1 and z_2.

Specifically, if transmission errors have occurred in columns i and j of the received word, then there may be several choices of i and j for which $\alpha^i + \alpha^j = z_1$. We would like to define $f(x)$ such that the true values of i and j can be determined from these possibilities by the equation $f(\alpha^i) + f(\alpha^j) = z_2$. There are some choices of $f(x)$ which do not work. For instance, suppose $f(\alpha^i) + f(\alpha^j) = f(\alpha^i + \alpha^j)$, which is the case if, for example, $f(\alpha^j) = \alpha^{2j}$. Then

$$f(\alpha^i) + f(\alpha^j) = f(\alpha^i + \alpha^j) = f(z_1),$$

and we have not gained any additional information.

Now, the only nonzero elements in the field $GF(2^r)$ are the $2^r - 1$ distinct powers of α, so it must be the case that $f(\alpha) = \alpha^j$ for some j. It can be shown that the addition of the extra rows to H is helpful if and only if α^j does not have the same minimal polynomial as α. Dividing the field $GF(2^r)$ into subsets of elements which share a minimal polynomial is useful, and is made possible by the following theorem:

Theorem 3 *If $\beta \in GF(2^r)$, then β and β^2 have the same minimal polynomial.*

The subsets of $GF(2^r)$ obtained in this fashion are called *cyclotomic cosets*. The minimal polynomial of the coset C_i with smallest element i is referred to as $M^{(i)}(x)$. It is necessary to choose $f(x)$ such that $f(x) = \alpha^j$ and j is not in the coset C_1. A choice which works in any $GF(2^r)$ is $j = 3$. The generator polynomial of the 2-error correcting BCH code with $f(\alpha) = \alpha^3$ is $g(x) = M^{(1)}(x)M^{(3)}(x)$.

Suppose we wish to expand the Hamming code $\mathcal{H}_3$ into a 2-error correcting BCH code by this method. The parity check matrix of this Hamming code is $(1, \alpha, \alpha^2, \ldots, \alpha^6)$. We can compute the parity check matrix of the 2-error correcting BCH code with $f(\alpha) = \alpha^3$. Recall that $\alpha^7 = 1 = \alpha^0$.

$$H = \left[\begin{array}{ccccccc} 1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ \hline 1 & \alpha^3 & \alpha^6 & \alpha^9 & \alpha^{12} & \alpha^{15} & \alpha^{18} \end{array}\right] = \left[\begin{array}{ccccccc} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ \hline 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

If we wanted to generate codewords for this code, we would use the method suggested by Theorem 2 in combination with any irreducible polynomial $g(x)$ of degree 6 to obtain a generator matrix. However, without explicitly generating codewords, we can demonstrate the error-correcting properties of this code.

Suppose one transmission error occurs in column 3. Then the syndrome is $Hy = 001011$, which is simply column 3 of H. Suppose two errors occur, in columns 3 and 6. Then the syndrome is 111001. Now, $z_1 = 111$ could imply either a single error in column 5 or some combination of errors. But if only one error occurred, then z_2 would be equal to $z_1^3 = 110$. Since this is not the case, there must have been more than one error.

The value of z_1 is consistent with errors in columns 1 and 7, columns 2 and 4, or columns 3 and 6. In the first case, z_2 would be equal to $1 + \alpha^{18} = 011$. In the second case, z_2 would be equal to $\alpha^3 + \alpha^9 = 100$. In the third case, z_2 would be equal to $\alpha^6 + \alpha^{15} = 001$. Since $z_2 = 001$, correct errors in positions 3 and 6, then decode.

Further, suppose three errors occur, in columns 2, 5, and 6. Then $z_1 = 011$ and $z_2 = 001$. Now, z_2 is not consistent with a single error in column 7, so more than one error occurred. Double errors could have occurred in columns 1 and 5, 2 and 3, or 4 and 6, but in these cases, z_2 would have been 010, 110, or 011, respectively. Since none of these is the case, there must have been more than two errors, so the decoder reports failure.

Something else noteworthy about this code is its minimum distance. Since it was designed to correct 2 errors, we would expect it to have minimum distance 5. However, the dimensions of H imply $k = 1$, so the actual minimum distance is $d = 7$.

In general, BCH codes are created with a *designated minimum distance* δ. The true minimum distance of the code will be $d \geq \delta$, and the dimension of the code will be $k = n - \deg(g(x)) \geq n - r(\delta - 1)$. (The length of the code is always $n = 2^r - 1$ and the code corrects $\lfloor \frac{1}{2}(d-1) \rfloor$ errors, as described in Theorem 1. The narrow-sense[1] BCH code with designated minimum distance δ has generator polynomial

$$g(x) = \text{lcm}\{M^{(1)}(x), M^{(2)}(x), \ldots, M^{(\delta-1)}\}.$$

3 Weight Distribution of Codes

The minimum distance of a linear code is the smallest weight of any nonzero codeword, and is important because it directly affects the number of errors the code can correct, as demonstrated in Theorem 1. However, the minimum distance says nothing about the weights of other codewords in the code; for that, it is necessary to look at the weight distribution of the entire code.

It turns out that the weight distribution of many types of codes, including Hamming and BCH codes, is asymptotically normal (that is, it approaches the normal distribution for large codes of a given variety). This fact is demonstrated in a theorem developed by V.M. Sidel'nikov.

[1] "Narrow-sense" means that the first factor of the generator polynomial is $M^{(1)}(x)$.

3.1 Theorem on Weight Distribution of Large Codes

Theorem 4 *Let $\mathcal{C}$ be an $[n, k, d]$ binary code with $n > 3$, and let $d' \geq 3$ be the minimum distance of the dual code $\mathcal{C}^\perp$. Then*

$$|A(z) - \Phi(z)| \leq \frac{20}{\sqrt{d'}}. \tag{1}$$

The statement of this theorem requires further explanation, since it includes some concepts not introduced in the first two sections of this paper.

3.1.1 Dual Codes

The dual code $\mathcal{C}^\perp$ of $\mathcal{C}$ is the code consisting of all vectors h such that, if $g \in \mathcal{C}$, then $h \cdot g = 0$ (where the dot product is computed in the binary group $\mathbf{Z}_2$). The code $\mathcal{C}^\perp$ is an $[n, n-k]$ code. The construction of $\mathcal{C}^\perp$ from $\mathcal{C}$ is very simple: if $\mathcal{C}$ has parity check matrix H and generator matrix G, then $\mathcal{C}^\perp$ is the code with parity check matrix G and generator matrix H.

The quantity d' is the minimum distance of the dual code. There is no explicit relationship between the code $\mathcal{C}$ and the minimum distance of $\mathcal{C}^\perp$, but i will explore the relationship for a few specific codes in the final section of this paper.

3.1.2 Definitions of Weight Distribution of Codes

The function $A(z)$ is defined for any real number z in the range $(-\infty, \infty)$. It is called the *cumulative distribution function*, and is defined by:

$$A(z) = \sum_{j=\lceil \mu - \sigma z \rceil}^{n} a_j.$$

This needs to be broken down further. For an $[n, k, d]$ code $\mathcal{C}$, a_j is the number $a_j = \frac{A_j}{2^k}$, where A_j is the number of codewords in $\mathcal{C}$ which have weight j. Thus, the sum of the elements of the vector $a = (a_0, \ldots, a_n)$ is $\sum a_j = 1$, and a_j is the ratio of codewords with weight j to total codewords.

Now, $\mu = \mu(a)$ is the mean weight of the codewords in $\mathcal{C}$ and σ is the square root of the variance in weight of the codewords. Mean and variance are defined by:

$$\begin{aligned} \mu(a) &= \sum_{j=0}^{n} j a_j, \\ \sigma^2(a) &= \sum_{j=0}^{n} (\mu - j)^2 a_j. \end{aligned}$$

So, the cumulative distribution function $A(z)$ measures the fraction of total codewords which have weight greater than or equal to $\mu - \sigma z$. For any code, $A(-\infty) = 0$ and $A(\infty) = 1$.

The cumulative distribution function of the normal distribution is well known: it is

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-t^2/2} dt.$$

We can now restate Theorem 4. It says that, at any point z, the difference between the cumulative distribution function of any code and the cumulative distribution function of the normal distribution is bounded above by the number $\frac{20}{\sqrt{d'}}$. However, both $A(z)$ and $\Phi(z)$ range in value from 0 to 1, so this statement is meaningless unless $d' > 400 = 20^2$.

3.2 Proof of Theorem 4

This proof consists of several steps, which i outline below.

(A) Demonstrate some useful statistical properties of the distribution $\vec{a}$ of a code $\mathcal{C}$:

(i) Show that the distribution $\vec{a}$ has mean $\mu = \frac{n}{2}$ and variance $\sigma^2 = \frac{n}{4}$.

(ii) Show that the characteristic function $\alpha(t)$ of $\vec{a}$ can be approximated by the first $r = 2\lfloor \frac{1}{2}(d'-1) \rfloor$ terms of its Taylor series

$$\alpha(t) = \sum_{r=0}^{\infty} \mu_r(\vec{a}) \frac{(it)^r}{r!}, \tag{2}$$

and that the accuracy of the approximation is governed by the binomial distribution $\vec{b}$:

$$\left| \alpha(t) - \sum_{k=0}^{r-1} \mu_k(\vec{a}) \frac{(it)^k}{k!} \right| \leq \mu_r(\vec{b}) \frac{|t|^r}{r!}. \tag{3}$$

(B) Introduce the inequality

$$|A(z) - \Phi(z)| \leq \frac{1}{\pi} \int_{-T}^{T} \left| \frac{\alpha(t) - e^{-t^2/2}}{t} \right| dt + \frac{24}{T\pi\sqrt{2\pi}}, \tag{4}$$

from probability theory. This holds for any $T > 0$. Find a bound on (4) as a function of T and r as follows:

(i) Show that the integral in (4) satisfies

$$\int_{-T}^{T} \left| \frac{\alpha(t) - e^{-t^2/2}}{t} \right| dt \leq \int_{-T}^{T} \frac{1}{|t|} \left| \sum_{k=0}^{\infty} \mu_k(\vec{b}) \frac{(it)^k}{k!} - e^{-t^2/2} \right| dt + 2 \int_{-T}^{T} \mu_r(\vec{b}) \frac{|t|^{r-1}}{r!} dt. \tag{5}$$

(ii) Show that, for $T < \sqrt{n}$,

$$\int_{-T}^{T} \frac{1}{|t|} \left| \sum_{k=0}^{\infty} \mu_k(\vec{b}) \frac{(it)^k}{k!} - e^{-t^2/2} \right| dt \leq \frac{168}{25n}. \tag{6}$$

(iii) Show that

$$\int_{-T}^{T} \mu_r(\vec{b}) \frac{|t|^{r-1}}{r!} dt \leq \frac{2T^r e^{1/6}}{r!\sqrt{r}} \left(\frac{r}{e} \right)^{\frac{r}{2}}. \tag{7}$$

(C) Show that the substitution

$$T = \left(\frac{r}{e}\right)^{\frac{1}{2}} \left(\frac{6}{e^{1/6}}\right)^{\frac{1}{r}},$$

is valid in the resulting inequality

$$|A(z) - \Phi(z)| \leq \frac{168}{25\pi n} + \frac{4T^r e^{1/6}}{\pi r! \sqrt{r}} \left(\frac{r}{e}\right)^{\frac{r}{2}} + \frac{24}{T\pi\sqrt{2\pi}} \tag{8}$$

and that (1) follows from this choice.

Note that the bound on $A(z)$ stated in Theorem 4 is approximate; better bounds exist, and, in fact, Sidel'nikov has proven a similar theorem with a bound of $\frac{9}{\sqrt{d'}}$. The bounds established at each step in the proof given here are not necessarily the tightest ones which exists. However, the idea of the theorem — that many linear codes are asymptotically normal in the size of the code — can be shown to be true merely by establishing that some upper bound exists.

3.2.1 Proof of Theorem 4, Section (A)

Section (A), Part (i):

The restrictions on the mean and variance of $\vec{a}$ follow from two lemmas, which i state without proof:

Lemma 1 *If a linear code $\mathcal{C}$ has a dual code $\mathcal{C}^\perp$ such that $A'_0 = 1$ and $A'_1 = 0$, then $\mathcal{C}$ has mean weight $\mu = \frac{n}{2}$.*

Lemma 2 *If a linear code $\mathcal{C}$ has a dual code $\mathcal{C}^\perp$ such that $d' \geq 3$, then*

$$\sum_{j=0}^{n} j^2 a_j = \frac{1}{4} n(n+1).$$

Restating Lemma 2 gives:

$$\frac{n}{4} = \sum_{j=0}^{n} j^2 a_j - \frac{1}{4} n^2.$$

Since $\sum a_j = 1$,

$$\frac{n}{4} = \sum_{j=0}^{n} \left(j^2 a_j - \frac{1}{4} n^2 a_j\right) = \sum_{j=0}^{n} \left(j^2 - \frac{1}{4} n^2\right) a_j.$$

Now, from Lemma 1, $\mu = \frac{1}{2} n$, so

$$\frac{n}{4} = \sum_{j=0}^{n} \left(j^2 - \mu^2\right) a_j = \sum_{j=0}^{n} (j-\mu)^2 a_j + 2\mu \sum_{j=0}^{n} (j-\mu) a_j.$$

By definition of mean,

$$\sum_{j=0}^{n} (j-\mu) a_j = \mu - \mu \sum_{j=0}^{n} a_j = 0.$$

So,

$$\frac{n}{4} = \sum_{j=0}^{n} (j-\mu)^2 a_j = \sigma^2.$$

Section (A), Part (ii):

As noted above, the characteristic function $\alpha(t)$ of the weight distribution $\vec{a}$ is defined

$$\alpha(t) = \sum_{r=0}^{\infty} \mu_r(\vec{a}) \frac{(it)^r}{r!},$$

where $\mu_r(\vec{a})$ is the rth central moment of $\vec{a}$:

$$\mu_r(\vec{a}) = \sum_{j=0}^{n} \left(\frac{\mu - j}{\sigma}\right)^r a_j,$$

for some $r \geq 0$. Thus, (2) is the Taylor series expansion of

$$\alpha(t) = \sum_{j=0}^{n} a_j e^{it(\mu - j)/\sigma}. \tag{9}$$

We would like to put an upper bound on the difference between $\alpha(t)$ and the first r terms of (2), for some $r \leq d'$. In order to do this, first demonstrate:

Lemma 3 *If r is any positive integer and $t > 0$, then*

$$\left| e^{it} - \sum_{k=0}^{r-1} \frac{(it)^k}{k!} \right| \leq \frac{t^r}{r!}.$$

Given Lemma 3, we can establish a bound on the difference between $\alpha(t)$ and the first r terms of its Taylor expansion. The next step in doing this is to show that the equation

$$\left| \alpha(\xi + t) - \sum_{k=0}^{r-1} \frac{t^k}{k!} \alpha^{(k)}(\xi) \right| \leq \mu_r(\vec{a}) \frac{|t|^r}{r!} \tag{10}$$

holds for any even integer $r > 0$ and any $t \in \mathbf{R}$.

To see that this is true, take the value for $\alpha(t)$ given in (9). Now, we know $\mu = \frac{n}{2}$ and $\sigma^2 = \frac{n}{4}$, so, from this, we get

$$\alpha(t) = \sum_{j=0}^{n} a_j e^{it(\frac{n}{2} - j)/\frac{\sqrt{n}}{2}} = \sum_{j=0}^{n} a_j e^{itq}$$

if we set $q = \frac{n-2j}{\sqrt{n}}$. Since q is constant with respect to ξ, the kth derivative of $\alpha(\xi)$ is:

$$\alpha^{(k)}(\xi) = \left(\frac{d}{d\xi}\right)^k \left(\sum_{j=0}^{n} a_j e^{iq\xi}\right) = \sum_{j=0}^{n} a_j \left(\frac{d}{d\xi}\right)^k \left(e^{iq\xi}\right) = \sum_{j=0}^{n} a_j (iq)^k \left(e^{iq\xi}\right).$$

So the left hand side of (10) becomes

$$\left| \alpha(\xi + t) - \sum_{k=0}^{r-1} \frac{t^k}{k!} \alpha^{(k)}(\xi) \right| = \sum_{j=0}^{n} a_j \left| e^{iq\xi} \right| \left| e^{iqt} - \sum_{k=0}^{r-1} \frac{t^k}{k!} (iq)^k \right|.$$

Now, $iq\xi$ is purely imaginary, so, by Euler's formula, $|e^{iq\xi}| = 1$. Also, $q = \frac{n-2j}{\sqrt{n}}$, and since $0 \leq j \leq n$, $\max(q) = \sqrt{n}$. So,

$$\sum_{j=0}^{n} a_j \left|e^{iq\xi}\right| \left|e^{iqt} - \sum_{k=0}^{r-1} \frac{t^k}{k!}(iq)^k\right| \leq \sum_{j=0}^{n} a_j \left|e^{i\sqrt{n}t} - \sum_{k=0}^{r-1} \frac{(i\sqrt{n}t)^k}{k!}\right|. \tag{11}$$

Now, we look at the right hand side of (10). Given $\mu = \frac{n}{2}$ and $\sigma = \sqrt{\frac{n}{4}}$, we can expand $\mu_r(\vec{a})$, obtaining:

$$\mu_r(\vec{a})\frac{|t|^r}{r!} = \sum_{j=0}^{n} \left(\frac{\mu - j}{\sigma}\right)^r a_j \frac{|t|^r}{r!} = \sum_{j=0}^{n} a_j \frac{|t|^r}{r! n^{r/2}}(n-2j)^r.$$

Again, we know $0 \leq j \leq n$, so $\min(n-2j) = -n$. Thus,

$$\mu_r(\vec{a})\frac{|t|^r}{r!} \geq \sum_{j=0}^{n} a_j \frac{|t|^r}{r! n^{r/2}}(-n)^r = \sum_{j=0}^{n} a_j \frac{|t|^r n^{r/2}}{r!} \tag{12}$$

if r is even.

Now, we would like to show that (11) is less than or equal to (12). It is clear that, if this is true for every value of j, $0 \leq j \leq n$, then it is true for the sum of these values. For a given j, if $a_j = 0$, then both (11) and (12) are identically 0. Otherwise, we can divide through by a_j, at which point Lemma 3 gives us the desired result:

$$\left|e^{i\sqrt{n}t} - \sum_{k=0}^{r-1} \frac{(i\sqrt{n}t)^k}{k!}\right| \leq \frac{(\sqrt{n}t)^r}{r!} = \frac{t^r n^{r/2}}{r!}.$$

Thus, (10) is true for $t > 0$. But if $t < 0$, then $-t > 0$, and $(-t)^r = (-1)^r t^r$, so, since r is even, the formula still holds.

Now, note that

$$\alpha^{(k)}(0) = \sum_{j=0}^{n} a_j \left(\frac{i(\mu - j)}{\sigma}\right)^k = i^k \sum_{j=0}^{n} a_j \left(\frac{\mu - j}{\sigma}\right)^k = i^k \mu_k(\vec{a}).$$

So, if $\xi = 0$, (10) becomes

$$\left|\alpha(t) - \sum_{k=0}^{r-1} \mu_k(a)\frac{(it)^k}{k!}\right| \leq \mu_r(\vec{a})\frac{|t|^r}{r!}. \tag{13}$$

To relate this to the normal distribution, we need the following lemma:

Lemma 4 *If $\mathcal{C}$ is a linear code whose dual code has minimum distance d' and if $\vec{a}$ is the weight distribution vector of $\mathcal{C}$, then $\mu_r(\vec{a}) = \mu_r(\vec{b})$ for $0 \leq r \leq d' - 1$.*

Given Lemma 4, we can choose $r = 2\lfloor \frac{1}{2}(d'-1) \rfloor$ to satisfy both $r < d'$ and $2|r$, and (13) implies

$$\left|\alpha(t) - \sum_{k=0}^{r-1} \mu_k(\vec{a})\frac{(it)^k}{k!}\right| \leq \mu_r(\vec{b})\frac{|t|^r}{r!}. \tag{14}$$

3.2.2 Proof of Theorem 4, Part (B)

The equation (4) comes from probability theory. I will restate it below as a lemma and use it without proof.

Lemma 5 *Suppose $\alpha(t)$ is the characteristic function corresponding to some vector $\vec{a}$ of unit weight, and $A(z)$ is the cumulative distribution function of $\vec{a}$. Then, for any $T > 0$,*

$$|A(z) - \Phi(z)| \leq \frac{1}{\pi} \int_{-T}^{T} \left| \frac{\alpha(t) - e^{-t^2/2}}{t} \right| dt + \frac{24}{T\pi\sqrt{2\pi}}.$$

Section (B), Part (i):

By the triangle inequality,

$$\begin{aligned} \int_{-T}^{T} \left| \frac{\alpha(t) - e^{-t^2/2}}{t} \right| dt \quad \leq \quad & \int_{-T}^{T} \frac{1}{|t|} \left| \sum_{k=0}^{\infty} \mu_k(\vec{b}) \frac{(it)^k}{k!} - e^{-t^2/2} \right| dt \\ & + \int_{-T}^{T} \frac{1}{|t|} \left| \alpha(t) - \sum_{k=0}^{\infty} \mu_k(\vec{b}) \frac{(it)^k}{k!} \right| dt. \end{aligned} \tag{15}$$

Consider the second integrand on the right hand side of (15). From the definition of the characteristic function $\alpha(t)$:

$$\left| \alpha(t) - \sum_{k=0}^{\infty} \mu_k(\vec{b}) \frac{(it)^k}{k!} \right| = \left| \sum_{k=0}^{\infty} \mu_k(\vec{a}) \frac{(it)^k}{k!} - \sum_{k=0}^{\infty} \mu_k(\vec{b}) \frac{(it)^k}{k!} \right|.$$

Lemma 4 states that $\mu_k(\vec{a})$ and $\mu_k(\vec{b})$ are equal (and thus interchangeable) for $k < r$, so

$$\left| \sum_{k=0}^{\infty} \mu_k(\vec{a}) \frac{(it)^k}{k!} - \sum_{k=0}^{\infty} \mu_k(\vec{b}) \frac{(it)^k}{k!} \right| = \left| \sum_{k=r}^{\infty} \mu_k(\vec{a}) \frac{(it)^k}{k!} - \sum_{k=r}^{\infty} \mu_k(\vec{b}) \frac{(it)^k}{k!} \right|.$$

By another application of the triangle inequality,

$$\left| \sum_{k=r}^{\infty} \mu_k(\vec{a}) \frac{(it)^k}{k!} - \sum_{k=r}^{\infty} \mu_k(\vec{b}) \frac{(it)^k}{k!} \right| \leq \left| \sum_{k=r}^{\infty} \mu_k(\vec{a}) \frac{(it)^k}{k!} \right| + \left| \sum_{k=r}^{\infty} \mu_k(\vec{b}) \frac{(it)^k}{k!} \right|.$$

We can apply the bound obtained in (3) to both of these expressions, since it is valid for any characteristic function which meets its hypotheses, including that of the binomial distribution $\vec{b}$. So

$$\left| \sum_{k=r}^{\infty} \mu_k(\vec{a}) \frac{(it)^k}{k!} \right| + \left| \sum_{k=r}^{\infty} \mu_k(\vec{b}) \frac{(it)^k}{k!} \right| \leq 2\mu_r(\vec{b}) \frac{|t|^r}{r!}.$$

Substitute this new inequality into (15) to obtain (5), as desired.

Section (B), Part (ii):

To get a more explicit bound on the first integral in (5), first note that $\sum_{k=0}^{\infty} \mu_k(\vec{b}) \frac{(it)^k}{k!}$ is the Taylor series expansion of the characteristic function of $\vec{b}$. So

$$\sum_{k=0}^{\infty} \mu_k(\vec{b}) \frac{(it)^k}{k!} = \sum_{j=0}^{n} b_j e^{it(\mu-j)/\sigma}.$$

The binomial coefficients are $b_j = 2^{-n} \left(\begin{array}{c} n \\ j \end{array} \right)$ and they have $\mu = \frac{n}{2}$, and $\sigma^2 = \frac{n}{4}$, so

$$\begin{aligned} \sum_{j=0}^{n} b_j e^{it(\mu-j)/\sigma} &= 2^{-n} \sum_{j=0}^{n} \left(\begin{array}{c} n \\ j \end{array} \right) e^{it(\mu-j)/\sigma} \\ &= 2^{-n} \sum_{j=0}^{n} \left(\begin{array}{c} n \\ j \end{array} \right) e^{it\left(\sqrt{n} - \frac{2j}{\sqrt{n}}\right)} \\ &= 2^{-n} \sum_{j=0}^{n} \left(\begin{array}{c} n \\ j \end{array} \right) e^{it\sqrt{n}} e^{\frac{-2itj}{\sqrt{n}}} \\ &= 2^{-n} e^{it\sqrt{n}} \sum_{j=0}^{n} \left(\begin{array}{c} n \\ j \end{array} \right) e^{\frac{-2it}{\sqrt{n}} j} \end{aligned}$$

By the binomial theorem,

$$2^{-n} e^{it\sqrt{n}} \sum_{j=0}^{n} \left(\begin{array}{c} n \\ j \end{array} \right) 1^{(n-j)} e^{-2itn^{-1/2}j} = 2^{-n} e^{it\sqrt{n}} \left(1 + e^{-2itn^{-1/2}}\right)^n.$$

Rearranging terms gives

$$2^{-n} e^{it\sqrt{n}} \left(1 + e^{-2itn^{-1/2}}\right)^n = 2^{-n} \left(e^{itn^{-1/2}} \left(1 + e^{-2itn^{-1/2}}\right)\right)^n = 2^{-n} \left(e^{itn^{-1/2}} + e^{-itn^{-1/2}}\right)^n.$$

By Euler's formula, $\cos x = \frac{e^{ix} + e^{-ix}}{2}$, so

$$2^{-n} \left(e^{itn^{-1/2}} + e^{-itn^{-1/2}}\right)^n = 2^{-n} \left(2\cos\left(tn^{-1/2}\right)\right)^n = \cos^n\left(tn^{-1/2}\right).$$

Thus,

$$\int_{-T}^{T} \frac{1}{|t|} \left| \sum_{k=0}^{\infty} \mu_k(\vec{b}) \frac{(it)^k}{k!} - e^{-t^2/2} \right| dt = \int_{-T}^{T} \frac{1}{|t|} \left| \cos^n\left(tn^{-1/2}\right) - e^{-t^2/2} \right| dt. \tag{16}$$

And, in fact, since cosine is an even function,

$$\int_{-T}^{T} \frac{1}{|t|} \left| \cos^n\left(tn^{-1/2}\right) - e^{-t^2/2} \right| dt = 2\int_{0}^{T} \frac{1}{t} \left| \cos^n\left(tn^{-1/2}\right) - e^{-t^2/2} \right| dt.$$

One way to get a bound on this integral is by converting its argument to an exponential:

$$\begin{aligned} 2\int_{0}^{T} \frac{1}{t} \left| \cos^n\left(tn^{\frac{-1}{2}}\right) - e^{-t^2/2} \right| dt &= 2\int_{0}^{T} \frac{1}{t} \left| e^{n\ln(\cos(tn^{-1/2}))} - e^{-t^2/2} \right| dt \\ &= 2\int_{0}^{T} \frac{e^{-t^2/2}}{t} \left| e^{t^2/2 + n\ln(\cos(tn^{-1/2}))} - 1 \right| dt \\ &= 2\int_{0}^{T} \frac{e^{-t^2/2}}{t} \left| e^{n\left(\frac{1}{2}(tn^{-1/2})^2 + \ln(\cos(tn^{-1/2}))\right)} - 1 \right| dt. \end{aligned} \tag{17}$$

Substitute $\tau = tn^{-1/2}$ into this equation. Then the exponent on e is

$$n\left(\frac{1}{2}(tn^{-1/2})^2 + \ln(\cos(tn^{-1/2}))\right) = n\left(\frac{\tau^2}{2} + \ln(\cos(\tau))\right).$$

Since $n \geq d' \geq 3 > 0$, we can choose $T < \sqrt{n}$, so that $|\tau| < 1$ always holds. We can then use the expansion $\ln(1-x) = -\sum_{k=1}^{\infty} \frac{x^k}{k}$, as follows:

$$\begin{aligned}\left|\frac{\tau^2}{2} + \ln(\cos(\tau))\right| &= \left|\frac{\tau^2}{2} + \ln(1-(1-\cos(\tau)))\right| \\ &= \left|\frac{\tau^2}{2} - \sum_{k=1}^{\infty}\frac{(1-\cos(\tau))^k}{k}\right| \\ &= \left|\frac{\tau^2}{2} + \cos(\tau) - 1 - \sum_{k=2}^{\infty}\frac{(1-\cos(\tau))^k}{k}\right|.\end{aligned}$$

Using the triangle inequality on this last term gives

$$\left|\frac{\tau^2}{2} + \ln(\cos(\tau))\right| \leq \left|\cos(\tau) - 1 + \frac{\tau^2}{2}\right| + \left|\sum_{k=2}^{\infty}\frac{(1-\cos(\tau))^k}{k}\right|.$$

The following bounds on the value of cosine come from the Taylor series expansion:

$$1 - \frac{\tau^2}{2!} \leq \cos(\tau) \leq 1 - \frac{\tau^2}{2!} + \frac{\tau^4}{4!}.$$

Thus,

$$0 \leq \cos(\tau) - 1 + \frac{\tau^2}{2} \leq \frac{\tau^4}{4!}.$$

The second term satisfies

$$\left|\sum_{k=2}^{\infty}\frac{(1-\cos(\tau))^k}{k}\right| \leq \left|\sum_{k=2}^{\infty}\frac{1}{k}\left(\frac{\tau^2}{2}\right)^k\right| = \left|\frac{\tau^4}{8} + \sum_{k=3}^{\infty}\frac{1}{k}\left(\frac{\tau^2}{2}\right)^k\right| \leq \left|\frac{\tau^4}{8} + \frac{1}{3}\sum_{k=3}^{\infty}\left(\frac{\tau^2}{2}\right)^k\right|.$$

The sum is now simply a power series. It converges to $\frac{\tau^6/8}{1-\tau^2/2}$. If $|\tau| < 1$, then $1 - \frac{\tau^2}{2} > \frac{1}{2}$ and $\tau^6 \leq \tau^4$, so

$$\left|\frac{\tau^4}{8} + \frac{1}{3}\left(\frac{\tau^6/8}{1-\tau^2/2}\right)\right| < \left|\frac{\tau^4}{8} + \frac{2}{3}\left(\frac{\tau^6}{8}\right)\right| \leq \left|\frac{\tau^4}{8} + \frac{2}{3}\left(\frac{\tau^4}{8}\right)\right| = \frac{3\tau^4}{16} \leq \frac{\tau^4}{4}.$$

Thus,

$$n\left(\frac{\tau^2}{2} + \ln(\cos(\tau))\right) \leq n\left(\frac{\tau^4}{4!} + \frac{\tau^4}{4}\right) = \frac{7n\tau^4}{24} = \frac{7t^4}{24n}.$$

Substituting this value back into the integrand of (17) gives

$$\frac{e^{-t^2/2}}{t}\left|e^{n\left(\frac{1}{2}(tn^{-1/2})^2 + \ln(\cos(tn^{-1/2}))\right)} - 1\right| \leq \frac{e^{-t^2/2}}{t}\left(e^{\frac{7t^4}{24n}} - 1\right).$$

For any positive x, $e^x - 1 \leq xe^x$, so

$$\frac{e^{-t^2/2}}{t}\left(e^{\frac{7t^4}{24n}} - 1\right) \leq \frac{e^{-t^2/2}}{t}\left(\frac{7t^4}{24n}e^{\frac{7t^4}{24n}}\right) = \frac{7t^3}{24n}e^{\frac{7t^4}{24n} - \frac{t^2}{2}}.$$

As long as $T < \sqrt{n}$ (which was required earlier), $C\frac{t^2}{n} \leq C$ for any $C > 0$, so

$$\frac{7t^3}{24n}e^{\frac{7t^4}{24n} - \frac{t^2}{2}} \leq \frac{7t^3}{24n}e^{\frac{7t^2}{24} - \frac{t^2}{2}} = \frac{7t^3}{24n}e^{-5t^2/24}.$$

Now, before we can return to (17) and integrate, the limits of integration need to be adjusted to ensure that $T < \sqrt{n}^2$, but, at any rate, $T > 0$ and the integrand is always nonnegative, so integrating from 0 to ∞ will suffice. Integrate by substitution to obtain:

$$\frac{7}{12n}\int_0^\infty t^3 e^{-5t^2/24}dt = \frac{7}{12n}\lim_{L\to\infty} e^{-5t^2/24}\left(\frac{-24t^2}{10} - \frac{(24)^2}{50}\right)\Bigg|_{x=0}^{x=L} = \frac{7}{12n}\left(\frac{(24)^2}{50}\right) = \frac{168}{25n}.$$

Section (B), Part (iii):

Now consider the second integral on the right hand side of (5). Since $\mu_r(\vec{b})$ is constant with respect to t,

$$\int_{-T}^{T}\mu_r(\vec{b})\frac{|t|^{r-1}}{r!}dt = \frac{2\mu_r(\vec{b})}{r!}\int_0^T t^{r-1}dt = \frac{2\mu_r(\vec{b})}{r!}\left(\frac{T^r}{r}\right).$$

The following lemma establishes a bound on $\mu_r(\vec{b})$. Its proof comes from probability theory.

Lemma 6 *If $\vec{b}$ is the binomial distribution, then, given any positive even integer r,*

$$\mu_r(\vec{b}) \leq \left(\frac{r}{e}\right)^{\frac{r}{2}} r^{1/2}e^{1/6}.$$

The desired bound (7) follows straightforwardly from Lemma 6.

3.2.3 Proof of Theorem 4, Section (C)

We can now substitute (5), (6) and (7) back into the original bound given by (4). This, gives, for $T < \sqrt{n}$,

$$|A(z) - \Phi(z)| \leq \frac{168}{25\pi n} + \frac{4T^r e^{1/6}}{\pi r!\sqrt{r}}\left(\frac{r}{e}\right)^{\frac{r}{2}} + \frac{24}{T\pi\sqrt{2\pi}}.$$

We then demonstrate that this inequality can be bounded by a constant multiple of $r^{-1/2}$. To begin, since $n \geq r > \sqrt{r}$,

$$\frac{168}{25\pi}\left(\frac{1}{n}\right) < \frac{168}{25\pi}\left(\frac{1}{\sqrt{r}}\right). \tag{18}$$

Now, Stirling's equation states that

$$\lim_{r\to\infty} r! = \left(\frac{r}{e}\right)^r \sqrt{2\pi r},$$

or that, for any $n > 0$,

$$r! \geq \left(\frac{r}{e}\right)^r \sqrt{2\pi r}.$$

Thus,

$$\frac{4T^r e^{1/6}}{\pi r!\sqrt{r}} \left(\frac{r}{e}\right)^{\frac{r}{2}} \leq \frac{4T^r e^{1/6}}{\pi\sqrt{r}} \left(\frac{r}{e}\right)^{\frac{r}{2}} \left(\frac{e}{r}\right)^r \frac{1}{\sqrt{2\pi r}} = \frac{4e^{1/6}}{r\pi\sqrt{2\pi}} \left(\frac{e}{r}\right)^{\frac{r}{2}} T^r. \tag{19}$$

We would like to choose T such that T^r simplifies (19) as much as possible. Thus, take

$$T = \left(\frac{r}{e}\right)^{\frac{1}{2}} \left(\frac{6}{e^{1/6}}\right)^{\frac{1}{r}}.$$

This choice[2] is valid because $n \geq r + 1$ for any code, so, for any $n > 3$,

$$T \leq \left(\frac{r}{e}\right)^{\frac{1}{2}} \left(\frac{6}{e^{1/6}}\right) < \sqrt{n}.$$

Substituting into (19) gives

$$\frac{4T^r e^{1/6}}{\pi r!\sqrt{r}} \left(\frac{r}{e}\right)^{\frac{r}{2}} \leq \frac{24}{\pi\sqrt{2\pi}} \left(\frac{1}{r}\right) < \frac{24}{\pi\sqrt{2\pi}} \left(\frac{1}{\sqrt{r}}\right).$$

Given this choice of T,

$$\frac{24}{T\pi\sqrt{2\pi}} = \frac{24}{\pi\sqrt{2\pi}} \left(\frac{e}{r}\right)^{\frac{1}{2}} \left(\frac{e^{1/6}}{6}\right)^{\frac{1}{r}} \leq \frac{24e^{1/2}}{\pi\sqrt{2\pi}} \left(\frac{1}{\sqrt{r}}\right),$$

since $\frac{e^{1/6}}{6} < 1$. Substituting back into (8) gives

$$|A(z) - \Phi(z)| \leq \left(\frac{168}{25\pi} + \frac{24}{\pi\sqrt{2\pi}} + \frac{24\sqrt{e}}{\pi\sqrt{2\pi}}\right) \left(\frac{1}{\sqrt{r}}\right) = \frac{24}{\pi} \left(\frac{7}{25} + \frac{1+\sqrt{e}}{\sqrt{2\pi}}\right) \left(\frac{1}{\sqrt{r}}\right).$$

Now, $\frac{24}{\pi} < 10$ and $\frac{1+\sqrt{e}}{\sqrt{2\pi}} < \sqrt{2}$. By definition, either $d' = r + 1$ or $d' = r + 2$, so for $d' \geq 3$, $\frac{10\sqrt{2}}{\sqrt{r}} \leq \frac{20}{\sqrt{d'}}$. Thus, this statement leads to the desired relation

$$|A(z) - \Phi(z)| \leq \frac{20}{\sqrt{d'}}.$$

[2]The constant 6 is selected so that this value will be close to the optimal value for T, which can be computed by finding an appropriate zero of the derivative of the righthand side of (8).

3.3 Application of Theorem 4 to BCH Codes

As stated, Theorem 4 is useful only for values of d' greater than 400. Of course, in order to use this theorem, we need to know restrictions on d' based on the parameters of the original code $\mathcal{C}$.

One simple example of this is the even-weight code with parameters $[n, n-1, 2]$. The dual code of this code is the repetition code with parameters $[n, 1, n]$. Thus, the bound imposed by Theorem 4 is useful for $n > 400$. This example highlights the fact that the cumulative distribution function $A(z)$ of a code may coincide with $\Phi(z)$ even if the code does not follow the normal distribution at every point. For any even-weight code, $a_j = 0$ for every odd value of j, even at points near the mean weight where $A(z)$ is growing very fast overall.

For BCH codes, the following theorem, also by Sikel'nikov, gives a lower bound on d'. I will state it without proof.

Theorem 5 *If $\mathcal{C}$ is a binary BCH code of length $n = 2^r - 1$ and designated distance $2t + 1$, then $\mathcal{C}^\perp$ has minimum distance $d \geq 2^{r-1-\lceil \log_2(2t-1) \rceil}$.*

The application of this theorem is clear. Suppose $\mathcal{C}$ is a Hamming code ($t = 1$). Then, by Theorem 5, the minimum distance of the dual code is $d' \geq 2^{r-1}$. In fact, the minimum distance is exactly 2^{r-1}, since the dual of the Hamming code $\mathcal{H}_r$ is an $[n, r, 2^{r-1}]$ code whose codewords are 0 plus $2^r - 1$ codewords of weight 2^{r-1}. So Theorem 4 is useful for Hamming codes $\mathcal{H}_{10}$ or larger, and, for code $\mathcal{H}_{22}$, the bound is $|A(z) - \Phi(z)| < 0.01$. Hamming codes therefore become close to normally distributed fairly quickly.

In fact, even small Hamming codes resemble the binomial distribution strongly. Figure 1 shows Hamming codes next to the binomial distributions over the same weights, plus the cumulative distribution functions for the Hamming codes. The codes shown are $\mathcal{H}_3$ and $\mathcal{H}_8$. The code $\mathcal{H}_8$ appears normally distributed, and its cumulative distribution function has a clear sigmoid shape, matching the asymptotic shape of the binomial distribution.

For multiple-error correcting BCH codes, the bound is not useful until larger values. However, since the growth of $\lceil \log_2(2t-1) \rceil$ is slow, the size of BCH code needed in order to apply Theorem 4 does not grow very fast as a function of t. Thus, in conjunction with Theorem 5, Theorem 4 does give a useful restriction of BCH and Hamming codes to the normal distribution.

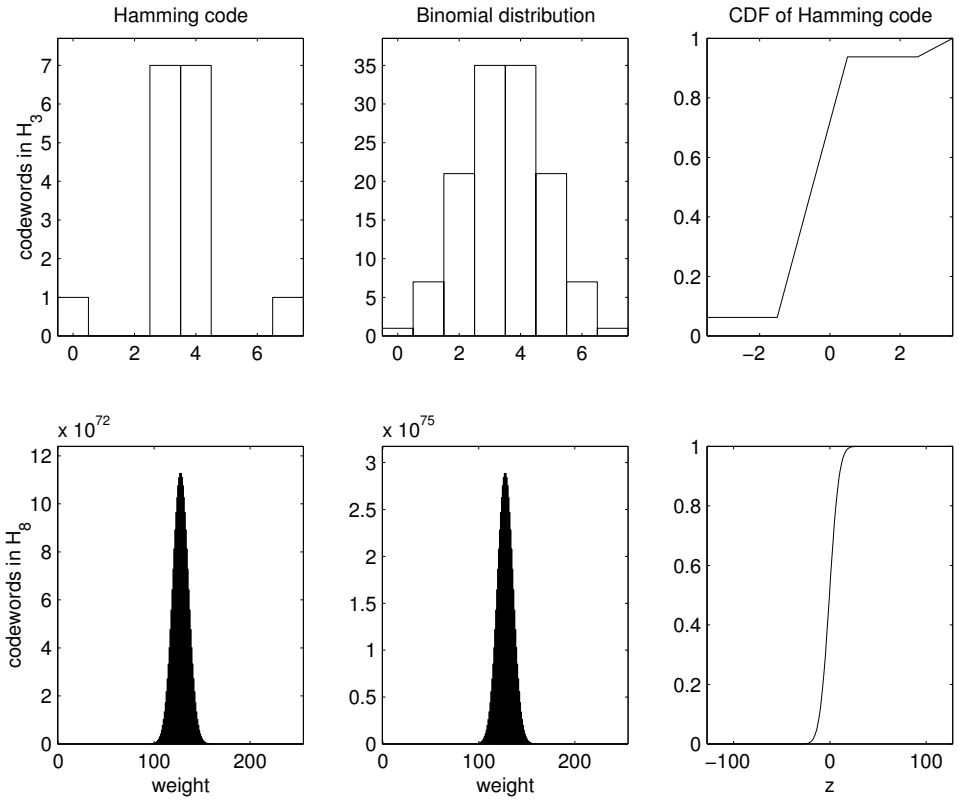

Figure 1: Weight distributions for small Hamming codes $\mathcal{H}_3$ and $\mathcal{H}_8$.