Lecture 12

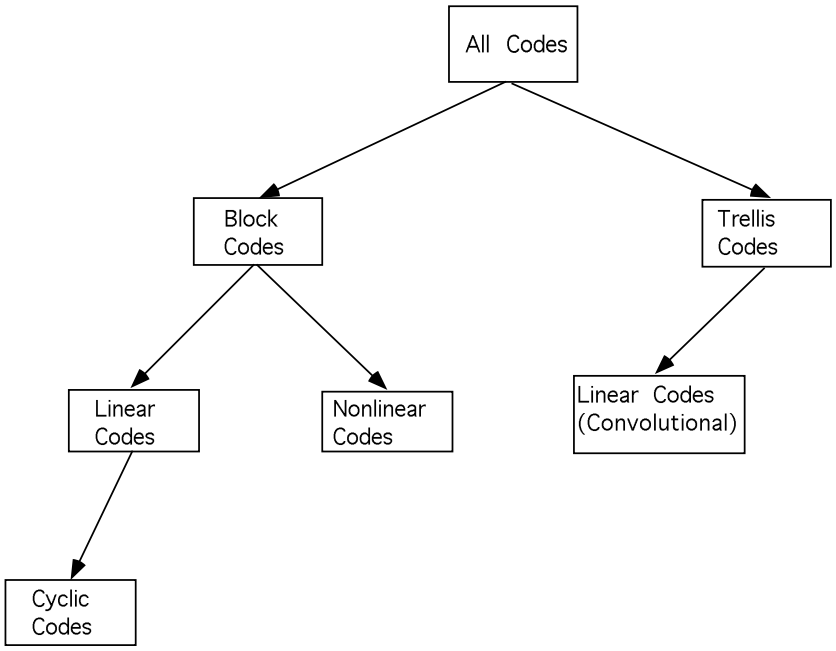
Goals

- Be able to encode using a linear block code
- Be able to decode a linear block code received over a binary symmetric channel or an additive white Gaussian channel

Block Diagram

Purpose: To use large dimensional signal space to improve performance of a communication system. Usually this mean trading increased complexity and increased bandwidth for significantly lower error probability.

Taxonomy of Codes



Definitions

Definition: A block code C is a set of M vectors (of possible channel inputs) of length n.

Definition: The rate of a code is $r = \frac{\log_2 M}{n}$ (measured in information bits/channel bit).

Definition: The minimum Hamming distance of a code is the smallest number of positions that any two (distinct) codewords differ by.

Definition: The minimum squared Euclidean distance of a code is the sum of the squares of the difference between code symbols from distinct codewords.

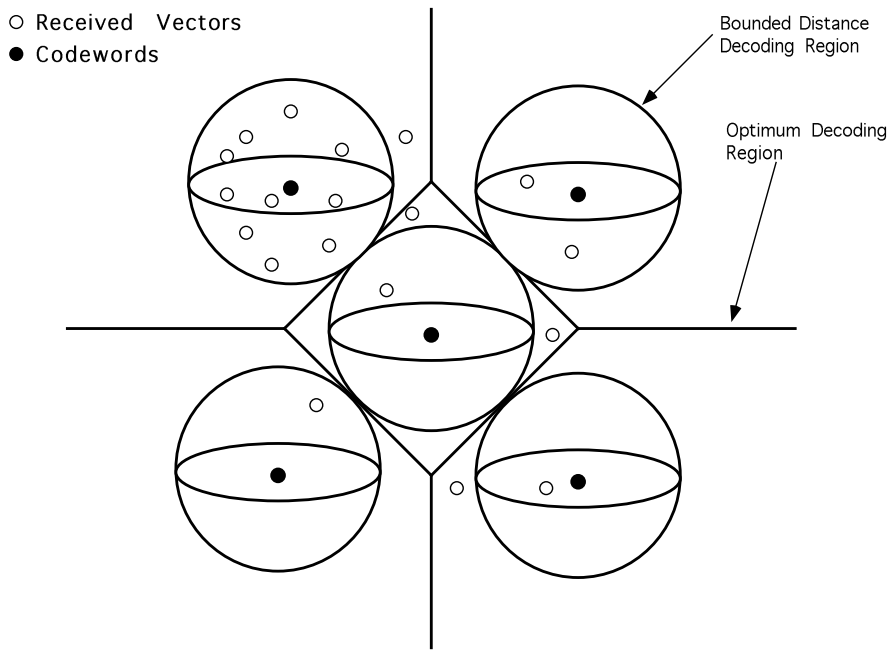


Figure 90: Decoding Region for Codes

There are usually two different representations of a code. In one representation the symbols are 0 and 1 while in the other representation the symbols are 1 and -1. The first representation is used when taking about adding two codewords (linearity) of a code. The later is used when talking about the Euclidean distance of a code. The mapping

$$\begin{aligned} 0 &\leftrightarrow 1 \\ 1 &\leftrightarrow -1 \end{aligned}$$

is used here.

Mathematically this is the mapping

$$\tilde{b} = (-1)^b$$

Examples

Code 1 (Repetition Code). $M = 2$, $n = 5$.

$$\begin{aligned} C &= \{(0,0,0,0,0),(1,1,1,1,1)\}. \\ \hat{C} &= \{(1,1,1,1,1),(-1,-1,-1,-1,-1)\} \end{aligned}$$

This is called the repetition code. The rate is $1/n$=$1/5$. The minimum Hamming distance is $d_{H,\min} = 5$. The minimum squared Euclidean distance is

$$d^2_{E,\min} = \sum_{k=1}^{5} (1-(-1))^2 = 5 \times 4 = 20.$$

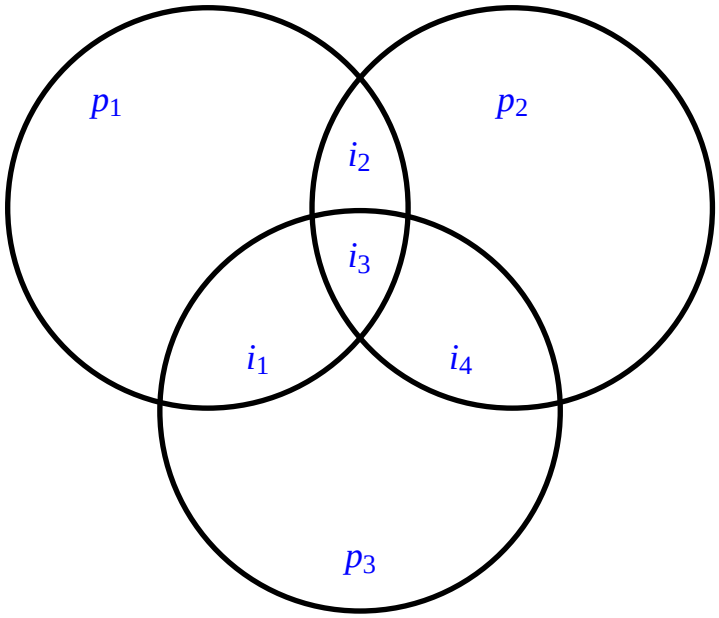
Code 2. $M = 4$, $n = 4$.

$$\begin{aligned} C &= \{(0,0,0,0),(0,0,1,1),(1,1,0,0),(1,1,1,1)\}. \\ \hat{C} &= \{(1,1,1,1),(1,1,-1,-1),(-1,-1,1,1),(-1,-1,-1,-1)\} \end{aligned}$$

The rate of this code is $1/2$. The minimum Hamming distance of this code is 2. The minimum squared Euclidean distance is $2 \times 4 = 8$.

Code 3: (Hamming Code) $M = 16$, $n = 7$.

$$C = \{(0,0,0,0,0,0,0),(0,0,0,1,0,1,1),...\}$$



Parity checks equations for Hamming code

$$\begin{aligned} p_1 &= i_1 + i_2 + i_3 (mod(2)) \\ p_2 &= i_2 + i_3 + i_4 (mod(2)) \\ p_3 &= i_1 + i_3 + i_4 (mod(2)) \end{aligned}$$

$$\begin{aligned} i_1 + i_2 + i_3 + p_1 &= 0 (mod(2)) \\ i_2 + i_3 + i_4 + p_2 &= 0 (mod(2)) \\ i_1 + i_3 + i_4 + p_3 &= 0 (mod(2)) \end{aligned}$$

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Let $c = (i_1, i_2, i_3, i_4, p_1, p_2, p_3)$ and

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

Then we obtain the relation $Hc^T = 0^T$.

The way the Hamming code works is as follows. There are four information bits from which three additional parity bits are determined (as described below). These 7 bits are transmitted over the channel. The channel (modulator, waveform channel, demodulator) may make errors.

Encoding

There are seven regions in the circles. There are four regions in two or more circles. In these four regions the information bits are written ($i_1,\ i_2,\ i_3\ i_4$). The three parity bits ($p_1,\ p_2,\ p_3$) are determined by making sure that the number of ones in each circle is even. The seven bits are transmitted over the channel. For example, if $i_1 = 0,\ i_2 = 1,\ i_3 = 1,\ i_4 = 1$ then the number of ones in the left circle (without including p_1 is 2 which is even so $p_1 = 0$. Similarly $p_2 = 1$ in order to make the number of ones in the right circle even. Finally $p_3 = 0$. So the codeword transmitted is (0111010).

The channel may make an error in one of the bits. For example the previous codeword could be received as (0101010).

Decoding

The decoder does not know which bit (if any) was changed by the channel. The decoder recomputes the parity and finds out which of the three circles has the correct parity. The error (if there is one or less) occurred in the circles where the parity did not check but not in the circles where the parity did check.

For the received vector being (0101010) we recompute the parity based on the information bits being (0101) and the parity bits being (010). In this case the parity of the left circle is odd. The parity of the right circle is odd and the parity of the bottom circle is odd. Thus we conclude that a single error must have occurred in all three circles and thus the bit representing i_3 was in error. The decoder would put out the decoded information (0111).

This decoder can correct an single error. It can also correct two erasures.

	Rate	d_E^2	d_H
Example 1	1/5	20	5
Example 2	1/2	8	2
Example 3	4/7	12	3

For binary coding and antipodal modulation (e.g. BPSK) the relation between Euclidean distance and Hamming distance is

$$d_{E,\min}^2 = 4E d_{H,\min}$$

where the output values take values $\{\pm\sqrt{E}\}$.

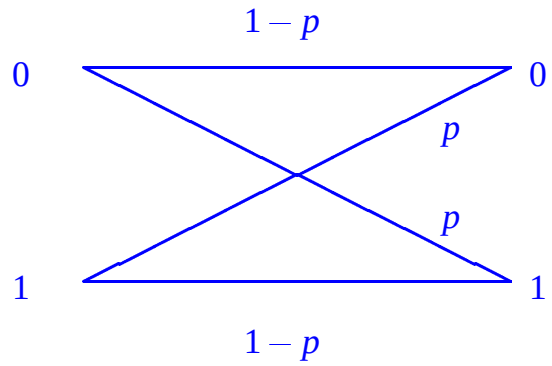
The goal in designing a code is to have as many codewords as possible (large M) but also as far apart as possible. There is a tradeoff between the code rate and the Euclidean distance of a code. The larger the code rate the smaller the Euclidean distance and the smaller the code rate the larger the Euclidean distance of a code.

The encoder maps information symbols into codewords. For Code 2 the encoder has the following mapping.

$$
\begin{aligned}
00 &\rightarrow \quad (\quad 1 \quad ,1 \quad ,1 \quad ,1 \quad) \\
01 &\rightarrow \quad (\quad 1 \quad ,1 \quad ,-1 \quad ,-1 \quad) \\
10 &\rightarrow \quad (\quad -1 \quad ,-1 \quad ,1 \quad ,1 \quad) \\
11 &\rightarrow \quad (\quad -1 \quad ,-1 \quad ,-1 \quad ,-1 \quad)
\end{aligned}
$$

The decoder must map the received symbols from the channel (which may be distorted) back to information symbols. It should do this to minimize the probability of making an error.

Example 1: Code 1 (Repetition Code) and Binary Symmetric Channel



Binary Symmetric Channel

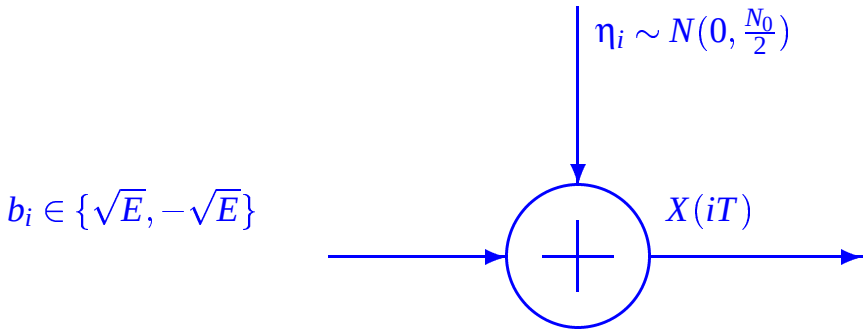
Data	Coded Data	Rcvd Data	Decoded Data
0	00000	01010	0
1	11111	11100	1

(Codewords of length L).

Decoding Rule: Choose codeword "closest" to demodulated data.

Conclusion: Decoder can correct up to $\lfloor (L-1)/2 \rfloor = 2$ errors with very low complexity.

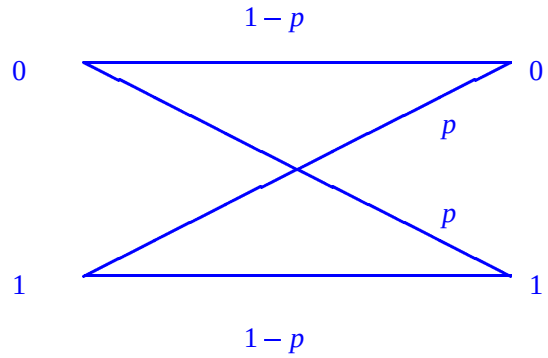
Example 2. Code 1 (Repetition Code) and an AWGN channel



The decoder finds the transmitted codeword by finding the coded data that is closest in Euclidean distance to the received data.

Data	Coded Data	Rcvd Data	Decoded Data
0	$(+\sqrt{E}, +\sqrt{E}, +\sqrt{E}, +\sqrt{E}, +\sqrt{E})$	1.2 0.5 0.8 -0.1 0.9	0
1	$(-\sqrt{E}, -\sqrt{E}, -\sqrt{E}, -\sqrt{E}, -\sqrt{E})$	0.1 0.2 0.3 -0.9 -1.1	1

Example 3: Code 2 and Binary Symmetric Channel

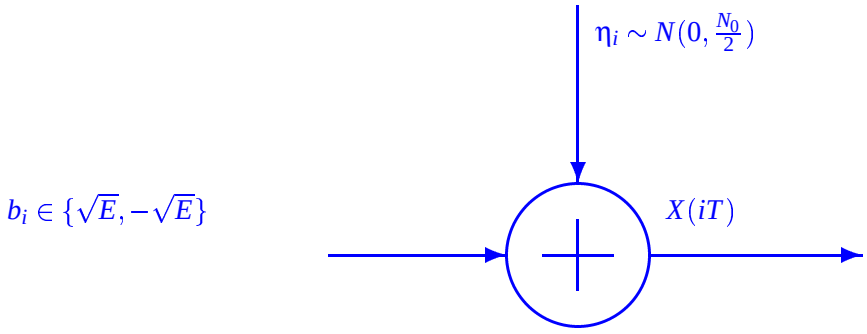


Binary Symmetric Channel

Data	Coded Data	Rcvd Data	Decoded Data
00	0000	0000	00
01	0011	0111	11

In the second case the received vector is equally close to two different codewords and chooses the wrong codeword.

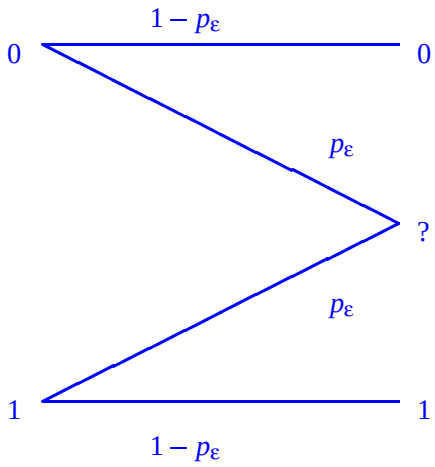
Example 4. Code 2 and an AWGN Channel



The decoder finds the transmitted codeword by finding the coded data that is closest in Euclidean distance to the received data.

Data	Coded Data	Rcvd Data	Decoded Data
00	$(+\sqrt{E}, +\sqrt{E}, +\sqrt{E}, +\sqrt{E})$	1.2 0.5 -0.1 0.9	00
01	$(+\sqrt{E}, +\sqrt{E}, -\sqrt{E}, -\sqrt{E})$	0.1 0.2 -0.9 -1.1	01

Example 5: Code 1 (Repetition Code) and Binary Erasure Channel

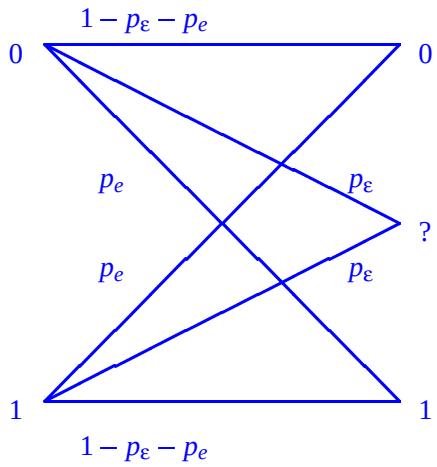


Data	Coded Data	Rcvd Data	Decoded Data
0	00000	0?0?0	0
1	11111	110??	Not possible

(Codewords of length L).

Decoder can correct up to $L-1$ erasures.

Example 6: Code 1 (Repetition Code) and Binary Errors and Erasures Channel



Data	Coded Data	Rcvd Data	Decoded Data
0	00000	0?0?1	0
1	11111	110??	1

(Codewords of length L).

Decoder ignores erased position and chooses closest "decimated" codeword.

Decoder can correct e errors and τ erasures if

$$2e+\tau \le L-1.$$

For other codes the decoder can correct e errors and τ erasures provided $2e+\tau \le d_{min}-1$ where d_{min} is the minimum Hamming distance between codewords.

Thus a code can correct (fi ll) twice as many erasures as it can correct errors. (With an erasure the location of the disturbance is known, whereas with an error the decoder does not know where it occurred).

Minimum Error Probability and Maximum Likelihood Decoding

Given a channel transition rule $p(\mathbf{z}|\mathbf{u})$ between the input $\mathbf{u} = (u_1, u_2, ..., u_N)$ to a (discrete) channel and the output $\mathbf{z} = (z_1, z_2, ..., z_N)$ and a set of possible transmitted (input) vectors (a code) $C = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_M\}$ with $\mathbf{c}_i = (c_{i,1}, c_{i,2}, ..., c_{i,N})$ we would like to find the optimal rule for determining, based on observing $\mathbf{z}$, which of the M vectors from C was transmitted. The goal is to minimize the overall probability of making an error in our decision.

Let H_m denote the event that the input to the channel (the codeword selected) is $\mathbf{c}_m$. Let π_m be the probability of this event; that is $\pi_m = P\{H_m\}$. The decoder observes $\mathbf{z}$ and makes a decision on which of the events

H_m, $m = 1, 2, ..., M$ occurred. The probability of error is given by

$$\begin{aligned} P_e &= \sum_{m=1}^{M} \sum_{k=1,\ k\neq m}^{M} P\{\text{decoder decides } H_k | H_m\} P\{H_m\} \\ &= \sum_{m=1}^{M} \left[\sum_{k=1,\ k\neq m}^{M} P\{\text{decide } H_k | H_m\} \right] \pi_m \\ &= \sum_{m=1}^{M} [1 - P\{\text{decide } H_m | H_m\}] \pi_m \end{aligned}$$

Consider the quantity $P\{\text{decide } H_m | H_m\}$. Let R_m be the set of all (channel) output vectors for which the decoder decides event (hypothesis) H_m occurred. Event H_m occurring is equivalent to the event that $\mathbf{c}_m$ was the input to the channel. Then

$$P\{\text{decide } H_m | H_m\} = \int_{R_m} p(\mathbf{z}|H_m) d\mathbf{z}$$

where the above integral is interpreted in one of two ways. If the channel is a continuous output channel then $p(\mathbf{z}|H_m)$ is a conditional density function and

the integral is a multidimensional integral over the region R_m. If the channel is a finite output channel then $p(\mathbf{z}|H_m)$ is a conditional probability mass function and the integral is really a sum over the region R_m. This will be made clear in the examples given below. Since $p(\mathbf{z}|H_m) = p(\mathbf{z}|\mathbf{c}_m)$ the above expression for probability of error becomes

$$\begin{aligned} P_e &= \sum_{m=1}^{M} \pi_m - \sum_{m=1}^{M} \int_{R_m} p(\mathbf{z}|\mathbf{c}_m)\pi_m d\mathbf{z} \\ &= 1 - \sum_{m=1}^{M} \int_{R_m} p(\mathbf{z}|\mathbf{c}_m)\pi_m d\mathbf{z}. \end{aligned}$$

To minimize the average error probability we would like to choose the regions R_m to maximize the second term. If for a given channel output, $p(\mathbf{z}|\mathbf{c}_5)\pi_5$ is larger than $p(\mathbf{z}|\mathbf{c}_k)\pi_k$ for $k \neq 5$ then choosing $\mathbf{z}$ to be in R_5 would make the last term largest. Thus the decision rule that minimizes average error probability is

$$\mathbf{z} \in R_m \text{ if } p(\mathbf{z}|\mathbf{c}_m)\pi_m = \max_{1 \leq k \leq M} p(\mathbf{z}|\mathbf{c}_k)\pi_k.$$

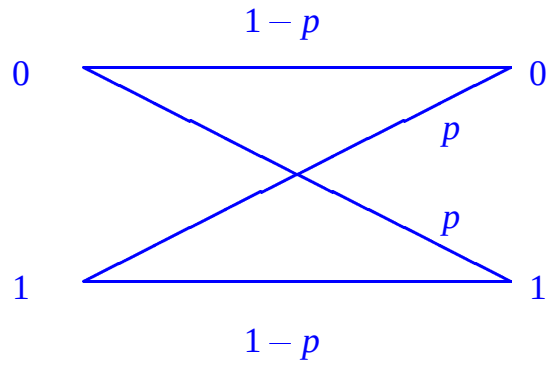
This is called the maximum a posteriori probability (MAP) decision rule. It is the rule that minimizes the average error probability. If all the prior probabilities are identical $\pi_m = 1/M$ then the optimum decision rule then reduces to

$$\mathbf{z} \in R_m \text{ if } p(\mathbf{z}|\mathbf{c}_m) = \max_{1 \leq k \leq M} p(\mathbf{z}|\mathbf{c}_k).$$

This is called the maximum likelihood decoding rule. It is useful if the prior probabilities are unknown to the system designer. In a digital communications context, if proper source coding has been done then the distribution on the input symbols should be uniform so maximum likelihood decoding optimum.

Example 1

Consider a binary symmetric channel with crossover probability $p < 1/2$. The input and output alphabet is just $\{0,1\}$.



The transition probability between an input vector and an output vector is

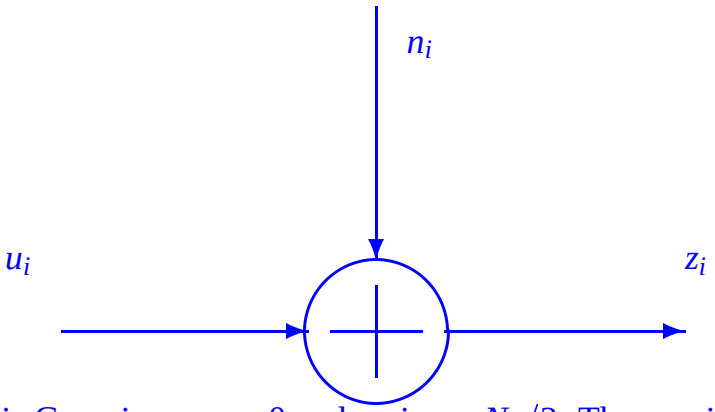
$$p(\mathbf{z}|\mathbf{u}) = p^{d_H(\mathbf{z},\mathbf{u})}(1-p)^{N-d_H(\mathbf{z},\mathbf{u})} = [\frac{p}{1-p}]^{d_H(\mathbf{z},\mathbf{u})}(1-p)^N$$

where $d_H(\mathbf{z},\mathbf{u})$ is the Hamming distance between the input vector $\mathbf{u}$ and the

output vector $\mathbf{z}$. If $p < 1/2$ then $\frac{p}{1-p} < 1$. So maximizing $p(\mathbf{z}|\mathbf{u})$ is equivalent to minimizing $d_H(\mathbf{z},\mathbf{u})$. So for a binary symmetric channel the optimum decision rule (assuming equal a prior probabilities) is to choose H_m (or equivalently $\mathbf{c}_m$) if $d_H(\mathbf{z},\mathbf{c}_m)$ is smallest. That is, choose the codeword closest in Hamming distance to the received vector (channel output).

Example 2

Consider an additive white Gaussian noise channel.



The noise n_i is Gaussian, mean 0 and variance $N_0/2$. The possible inputs to the channel are a finite set of real numbers (e.g $u_i \in \{+\sqrt{E}, -\sqrt{E}\}$) and the transition probability is

$$p(\mathbf{z}|\mathbf{u}) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\{-\frac{1}{2\sigma^2}(z_i - u_i)^2\} = [\frac{1}{\sqrt{2\pi}\sigma}]^N \exp\{-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(z_i - u_i)^2\}$$

$$= [\frac{1}{\sqrt{2\pi}\sigma}]^N \exp\{-\frac{1}{2\sigma^2} d_E^2(\mathbf{z}, \mathbf{u})\}.$$

where $d_E^2(\mathbf{z}, \mathbf{u}) = \sum_{i=1}^{N}(z_i - u_i)^2$ is the squared Euclidean distance between the channel input and output. Thus the optimal decoder finds $\mathbf{u}$ to minimize $d_E^2(\mathbf{z}, \mathbf{u})$. If the transmitted signals or codewords have equal energy then finding the codeword with the smallest Euclidean distance is the same as finding the codeword with the largest correlation ($\sum_{i=1}^{N} z_i u_i$).

Bounds on Performance

The performance of the maximum likelihood decoder is usually very difficult to evaluate exactly. Usually upper bounds on the decoding error probability are employed (for maximum likelihood decoding). These bounds are given by

$$P_e \leq \sum_{d=d_{min}}^{n} A_d P_2(d)$$

where $P_2(d)$ is the error probability between two codewords of a repetition code of length d and A_d is the number of codewords of weight d. The sum extends from $d = d_{min}$ the minimum nonzero weight, to n the length of the code. The bound is called the union bound. Notice that the effect on performance of the code is completely determined by A_d while the effect of the channel on the performance is determined by $P_2(d)$.

Pairwise Error Probability

For an additive white Gaussian noise channel with BPSK modulation the pairwise error probability between two codewords that differ in d positions is

$$P_2(d) = Q\left(\sqrt{\frac{2Ed}{N_0}}\right).$$

Note that E is the energy per code symbol. The energy per bit is

$$E_b = nE/k$$

Thus

$$P_2(d) = Q\left(\sqrt{\frac{2E_b rd}{N_0}}\right).$$

where $r = k/n$ is the rate of the code. Thus a key parameter of a code is the product of the rate and distance.

Pairwise Error Probability

For a binary symmetric channel the pairwise error probability between two codewords that differ in d positions is

$$P_2(d) = \begin{cases} \sum_{l=t}^{d} \binom{d}{l} p^l (1-p)^{d-l} & d \text{ odd} \\ \sum_{l=t}^{d} \binom{d}{l} p^l (1-p)^{d-l} + \frac{1}{2}\binom{d}{d/2} p^{d/2}(1-p)^{d/2} & d \text{ even} \end{cases}$$

where $t = \lfloor (d+1)/2 \rfloor$. The d even expression accounts for the possibilities of ties.

Union-Bhattacharyya Bound

This bound can be simplified greatly (and loosened) for binary codes by employing the Bhattacharyya bound. The result is

$$P_e \leq \sum_{d=d_{min}}^{n} A_d D^d$$

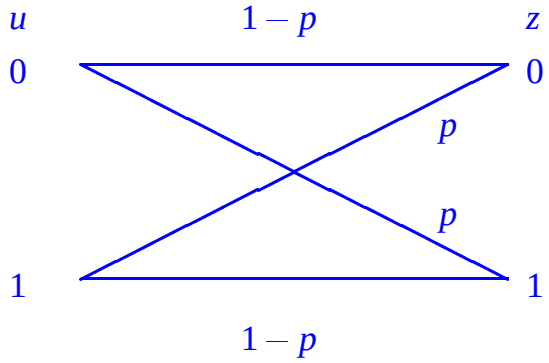
where

$$D = \sum_z \sqrt{p(z|0)\,p(z|1)}$$

Again the effect on performance of the code is through the weights A_d while the effect of the channel is through the parameter D.

Union-Bhattacharyya Bound

For a binary symmetric channel



$$\begin{aligned} D &= \sqrt{p(1|0)(p(0|0)} + \sqrt{p(0|1)p(1|1)} \\ &= \sqrt{p(1-p)} + \sqrt{(1-p)p} \\ &= 2\sqrt{p(1-p)} \end{aligned}$$

Union-Bhattacharyya Bound

For an additive white Gaussian noise channel

$$\begin{aligned} D &= \int_z \sqrt{p(z|0)p(z|1)}dz \\ &= \int_z \left[\frac{1}{\sqrt{2\pi}\sigma} e^{-(z-\sqrt{E})^2/(2\sigma^2)} \frac{1}{\sqrt{2\pi}\sigma} e^{-(z+\sqrt{E})^2/(2\sigma^2)} \right]^{1/2} dz \\ &= e^{-E/N_0} \end{aligned}$$

Repetition Code on Binary Symmetric Channel.

d	A_d
0	1
1	0
2	0
3	0
4	0
5	1

$$D = 2\sqrt{p(1-p)}$$

$$P_e \leq 1D^5 = (2\sqrt{p(1-p)})^5$$

Repetition Code on AWGN Channel.

$$D = e^{-E/N_0}$$

$$P_e \leq D^5 = e^{-E5/N_0} = e^{-E_b/N_0}$$

Code 2 on an AWGN Channel

d	A_d
0	1
1	0
2	2
3	0
4	1

For the additive white Gaussian noise (BPSK with coherent demodulation)

$$D = e^{-E/N_0}$$

$$P_e \leq 2D^2 + D^4$$

Hamming Code on an AWGN Channel

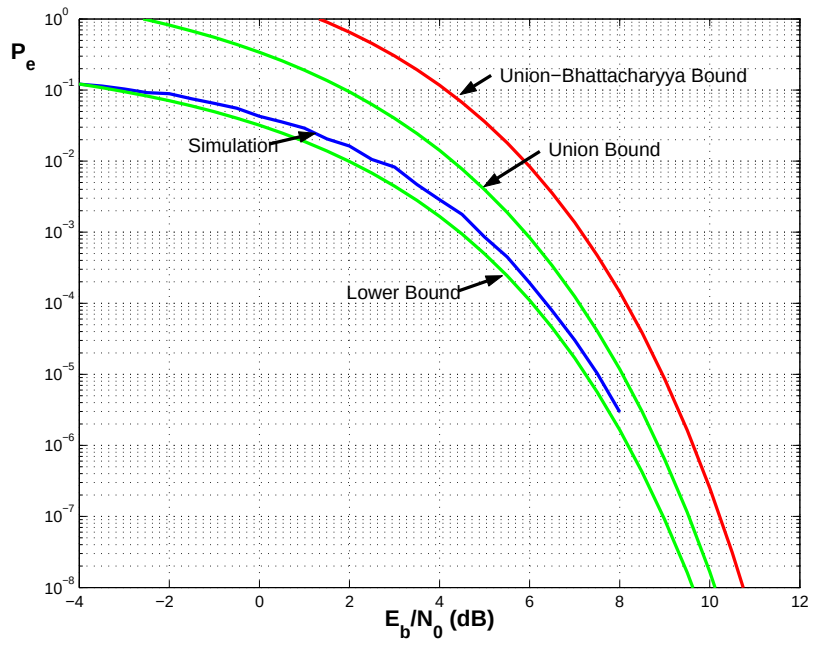
- Union Bound

$$P_e \leq 7Q\left(\sqrt{\frac{2E_b(4/7)3}{N_0}}\right) + 7Q\left(\sqrt{\frac{2E_b(4/7)4}{N_0}}\right) + 1Q\left(\sqrt{\frac{2E_b(4/7)7}{N_0}}\right)$$

- Union-Bhattacharyya Bound

$$\begin{aligned} P_e &\leq 7D^3 + 7D^4 + D^7 \\ &= 7e^{-[E_b(4/7)3/N_0]} + 7e^{-[E_b(4/7)4/N_0]} + e^{-[E_b(4/7)7/N_0]} \end{aligned}$$

Codeword Error Probability for Hamming Code on an AWGN Channel



Complexity of Decoding

For an arbitrary code the complexity of maximum likelihood decoding is large if M is large. Typical values of M are 2.7×10^{67} for one of the short Reed-Solomon codes used on a CD. The NASA standard (255,223) code has $M = 10^{537}$. Clearly this does not give a practical implementation. Thus we are forced to impose some structure on the code to reduce the complexity of decoding. The structure imposed is linearity and cyclicity.

Linear Block Codes

Definition: A code is said to be linear if the sum of any two codewords is also a codeword.

The minimum distance of a linear code is much easier to compute than the minimum distance of an arbitrary code.

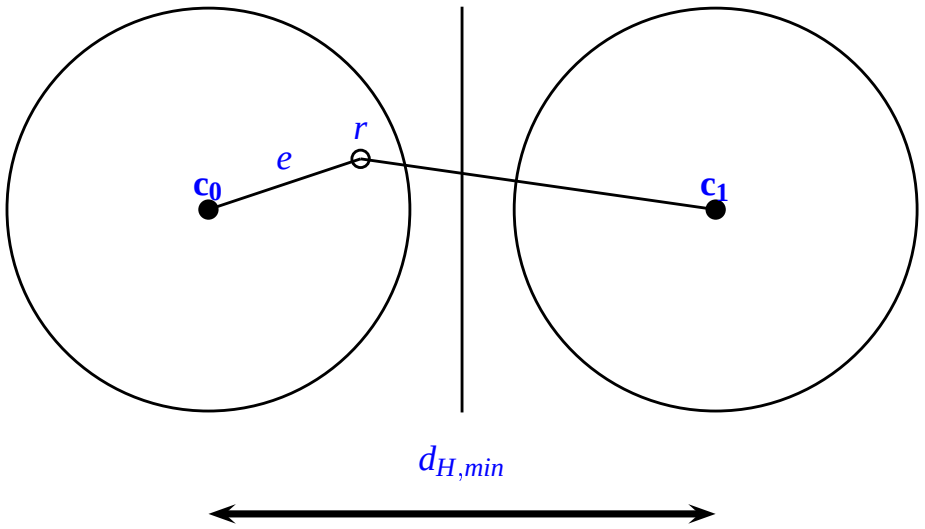
Proposition The minimum distance of a linear code is the minimum weight of any nonzero codeword.

$$\begin{aligned} d_{\min} &= \min_{\mathbf{c}_1 \neq \mathbf{c}_2} d_H(\mathbf{c}_1, \mathbf{c}_2) \\ &= \min_{\mathbf{c}_1 \neq \mathbf{c}_2} d_H(\mathbf{c}_1 - \mathbf{c}_2) \\ &= \min_{\mathbf{c} \neq 0} d_H(\mathbf{c}) \end{aligned}$$

Proposition: A block code can correct e errors provided $2e+1 \leq d_{H,min}$.

Proof: The decoder chooses the codeword that is closest in Hamming distance to the received vector. If e errors occurred then the received vector is distance e from the transmitted codeword. If the errors are in the right positions then the received vector could be distance $d_{H,min} - e$ from a codeword at the minimum distance. So correct decoding will occur if

$$\begin{aligned} e &< d_{H,min} - e \\ 2e &< d_{H,min} \\ 2e+1 &\leq d_{H,min} \end{aligned}$$



Linear code as subspace

The set of binary vectors of length n with components 0 and 1 forms a vector space. Addition is component wise mod 2 addition. Scalar multiplication is multiplication by 0 or 1.

$$\begin{array}{r} (01011001) \\ +(10010011) \\ \hline (11001010) \end{array}$$

$$0 \times (10110011) = (00000000), \quad 1 \times (10110011) = (10110011)$$

A vector space is a set of vectors and two operations (addition and multiplication) which is closed under addition, each vector has an additive inverse and satisfies the associative, commutative and distributed properties.

Linear code as subspace

A subspace of a vector space is a subset of the vector space that also satisfies the properties of a vector space. In particular it is closed under addition (the addition of any two elements in the subspace is also an element in the subspace).

A subspace C of a vector space V can be "generated" from a set of basis vectors by forming all linear combinations of the basis vectors.

Linear code as subspace

A linear code can be generated by a linear combination of $k = \log_2 M$ basis codewords. A generator matrix contains these k basis codewords. The particular linear combination used depends on the information bits to be transmitted. Because there are k basis vectors there are also k information bits. An (n,k) linear code is a mapping from k information bits to one of 2^k codewords of length n.

Example: (6,3) Code

$M = 8$, $n = 6$, $k = 3$.

$$\begin{aligned} \mathcal{C} &= \{(000000), (100101), (010111), (001011), \\ &\quad (110010), (101110), (011100), (111001)\} \end{aligned}$$

Generator Matrix

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

Here is how a codeword is generated. Assume the message is $m = 101$. Then

$$c = mG = [101] \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} = [101110]$$

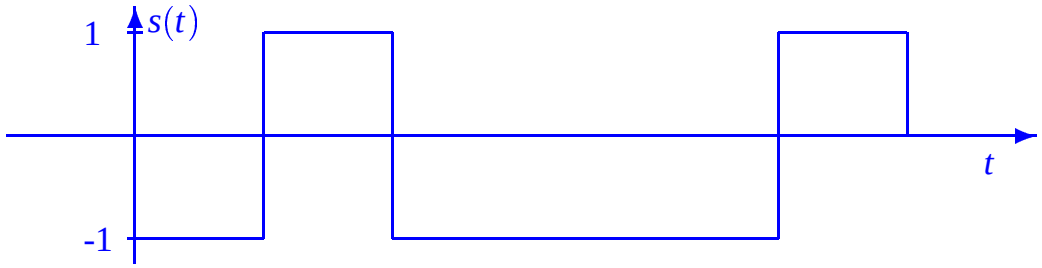
Thus the encoding is a simple matrix multiplication.

Coding and Modulation

If this codeword was to be transmitted we would convert the 0/1 vector into a +1/-1 vector as follows

$$(1,0,1,1,1,0) \rightarrow (-1,+1,-1,-1,-1,+1)$$

which for (baseband) signalling would be a waveform



Parity Checks

For a linear code there is a matrix called the parity check matrix H. All codewords $\mathbf{c}$ must satisfy the parity check equations

$$H\mathbf{c}^T = 0$$

This can be obtained from the generator matrix. From the generator matrix a set of equations relating the code symbols to the information symbols is straight forward.

For the example above we can write the equations

$$\begin{aligned} c_1 &= m_1 \\ c_2 &= m_2 \\ c_3 &= m_3 \end{aligned}$$

$$
\begin{aligned}
c_4 &= m_1 + m_2 = c_1 + c_2 \\
c_5 &= m_2 + m_3 = c_2 + c_3 \\
c_6 &= m_1 + m_2 + m_3 = c_1 + c_2 + c_3
\end{aligned}
$$

Rewriting the last three equations we obtain

$$
\begin{array}{cccccccc}
c_1 & +c_2 & & +c_4 & & & = 0 \\
 & c_2 & +c_3 & & +c_5 & & = 0 \\
c_1 & +c_2 & +c_3 & & & +c_6 & = 0
\end{array}
$$

These can be written in matrix form as

$$Hc^T = 0$$

The matrix H is known as the parity check matrix. For the above code the

parity check matrix is

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

For example the codeword (100101) satisfies this.

Channel

If the codeword is transmitted over a binary symmetric channel the received vector can be represented as

$$\mathbf{r} = \mathbf{c} + \mathbf{e}$$

where **e** is an error vector and the addition is done mod two. For example suppose the codeword (1,0,1,1,1,0) was transmitted over the channel and the noise in the channel causes the third bit to be in error. Then $\mathbf{e} = (0,0,1,0,0,0)$.

Decoding

Decoding is done by noticing that

$$\mathbf{s}^T = H\mathbf{r}^T = H(\mathbf{c}+\mathbf{e})^T = H\mathbf{c}^T + H\mathbf{e}^T = H\mathbf{e}^T.$$

Because there are more unknowns in the above equation (n unknowns and $n-k$ equations) we can not uniquely determine **e** from this. In fact there are 2^k different vectors **e** which satisfy the above equation. We would like to find the most likely such vector.

Example of Decoding

Let $\mathbf{r} = (011001)$. Compute $\mathbf{s}^T = H\mathbf{r}^T$

$$\mathbf{s}^T = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

The error patterns that satisfy this equation are

$(100000), (000101), (101011), (010010), (001110), (011001)(110111), (111100)$

The most likely error pattern is the one with the fewest errors (assuming a BSC with crossover probability less than 1/2). Thus the most likely error

pattern is $\mathbf{e} = (100000)$. The vector $\mathbf{s}$ is called the syndrome. The most likely error vector that yields a given syndrome is called the coset leader.

Thus an error is most likely to have occurred in the first position. Thus the decoder will decide that the transmitted codeword is (111001) and the channel made one error.

Standard Array for Decoding (6,3) code

Syndrome	Coset Leader							
000	000000	100101	010111	110010	001011	101110	011100	111001
101	100000	000101	110111	010010	101011	001110	111100	011001
111	010000	110101	000111	100010	011011	111110	001100	101001
011	001000	101101	011111	111010	000011	100110	010100	110001
100	000100	100001	010011	110110	001111	101010	011000	111101
010	000010	100111	010101	110000	001001	101100	011110	111011
001	000001	100100	010110	110011	001010	101111	011101	111000
110	101000	001101	111111	011010	100011	000110	110100	010001

Table of Codes

n	k	d_{min}	t	r
7	4	3	1	0.571
8	4	4	1	0.500
15	11	3	1	0.733
15	7	5	2	0.467
15	5	7	3	0.333
15	1	15	7	0.067
23	11	7	3	0.478
24	12	8	3	0.500

Table of Codes (cont.)

n	k	d_{min}	t	r
31	26	3	1	0.839
31	21	5	2	0.677
31	16	7	3	0.516
31	11	11	5	0.355
31	6	15	7	0.194
31	1	31	15	0.032
63	57	3	1	0.905
63	51	5	2	0.810
63	45	7	3	0.714

The Hamming Code

The Hamming code has the following parity check matrix

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

and generator matrix

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

Let

$$\begin{aligned}
\mathbf{v_1} &= (1,0,0,0,1,0,1) \\
\mathbf{v_2} &= (0,1,0,0,1,1,1) \\
\mathbf{v_3} &= (0,0,1,0,1,1,0) \\
\mathbf{v_4} &= (0,0,0,1,0,1,1)
\end{aligned}$$

These are a set of linear independent vectors that generate the code.

An alternative set of linear independent vectors that generate the code is

$$\begin{aligned}
\mathbf{v_1'} &= (1,0,1,1,0,0,0) \\
\mathbf{v_2'} &= (0,1,0,1,1,0,0) \\
\mathbf{v_3'} &= (0,0,1,0,1,1,0) \\
\mathbf{v_4'} &= (0,0,0,1,0,1,1)
\end{aligned}$$

To show that these are basis vectors for the code we need to show that they can be generated by the previous basis vectors and are linearly independent. It is easy to see that

$$\begin{aligned}
\mathbf{v_1'} &= \mathbf{v_1} + \mathbf{v_3} + \mathbf{v_4} \\
\mathbf{v_2'} &= \mathbf{v_2} + \mathbf{v_4} \\
\mathbf{v_3'} &= \mathbf{v_3} \\
\mathbf{v_4'} &= \mathbf{v_4}
\end{aligned}$$

To show that they are linearly independent consider

$$a_1\mathbf{v_1}\prime + a_2\mathbf{v_2}\prime + a_3\mathbf{v_3}\prime + a_4\mathbf{v_1}\prime, \qquad a_i \in \{0,1\}$$

The only way to get 0 in the first component from this linear combination is if $a_1 = 0$. The only way to get zero in the second component from this linear combination is if $a_2 = 0$. The only way to get zero in the last component from this linear combination is if $a_4 = 0$. Finally with $a_1 = a_2 = a_4 = 0$ the only way for the result to be 0 is if a_3 is also 0. Thus these vectors are linearly independent.

These basis vectors are cyclic shifts of each other. Also a cyclic shift of $\mathbf{v_4}\prime$ is $\mathbf{v_1}\prime + \mathbf{v_3}\prime + \mathbf{v_4}\prime$. Since the codewords are linear combinations of these basis vectors, a cyclic shift of a codeword is also a linear combination of these basis vectors and thus also a codeword.

The codewords that this code generate are

$$(0,0,0,0,0,0,0) \qquad (1,1,1,1,1,1,1)$$

$$(1,0,0,0,1,0,1) \qquad (0,1,0,0,1,1,1)$$
$$(1,1,0,0,0,1,0) \qquad (1,0,1,0,0,1,1)$$
$$(0,1,1,0,0,0,1) \qquad (1,1,0,1,0,0,1)$$
$$(1,0,1,1,0,0,0) \qquad (1,1,1,0,1,0,0)$$
$$(0,1,0,1,1,0,0) \qquad (0,1,1,1,0,1,0)$$
$$(0,0,1,0,1,1,0) \qquad (0,0,1,1,1,0,1)$$
$$(0,0,0,1,0,1,1) \qquad (1,0,0,1,1,1,0)$$

This code is called a cyclic code because every cyclic shift of a codeword is also a codeword. The minimum distance of this code is 3 (the minimum weight of any nonzero codeword). Thus this code is capable of correcting 1 error. If we consider for each codeword the number of received vectors that are decoded into it, then because it corrects any single error, there are 7 received vectors that differ from the codeword in one position. In addition, if the received vector is the codeword itself, then it will be decoded into codeword. Thus there are 8 received vectors that are decoded into each codeword. There are 16 codewords. This then accounts for $8 \times 16 = 128$ received vectors. But there are only $128 = 2^7$ possible received vectors. Thus there are no vectors outside of the single error correction decoding region.

Because this is a single error correcting code, the coset leaders must be all error patterns of weight 1. The coset leaders are then very easy to identify.

They are

Syndrome	Coset Leader
$(0,0,0)$	$(0,0,0,0,0,0,0)$
$(0,0,1)$	$(0,0,0,0,0,0,1)$
$(0,1,0)$	$(0,0,0,0,0,1,0)$
$(0,1,1)$	$(0,0,0,1,0,0,0)$
$(1,0,0)$	$(0,0,0,0,1,0,0)$
$(1,0,1)$	$(1,0,0,0,0,0,0)$
$(1,1,0)$	$(0,0,1,0,0,0,0)$
$(1,1,1)$	$(0,1,0,0,0,0,0)$

Thus to correct a error for this code, compute the syndrome and then identify which column of the matrix H is that syndrome (H contains all possible nonzero columns of length 3). The column that is the syndrome is the place a single error occurred. A double error will never be corrected for this code.

The bit error probability of the Hamming code can be calculated (for hard decision decoding) as

$$P_b = 9p^2(1-p)^5 + 19p^3(1-p)^4 + 16p^4(1-p)^3 + 12p^5(1-p)^2 + 7p^6(1-p) + p^7$$

The bound on the codeword error probability for soft decision decoding is

$$P_b \leq 7Q(\sqrt{\frac{6E}{N_0}}) + 7Q(\sqrt{\frac{8E}{N_0}}) + Q(\sqrt{\frac{14E}{N_0}}) \leq 7D^3 + 7D^4 + D^7$$

where $D = e^{-E/N_0}$ and $E = 4E_b/7$.

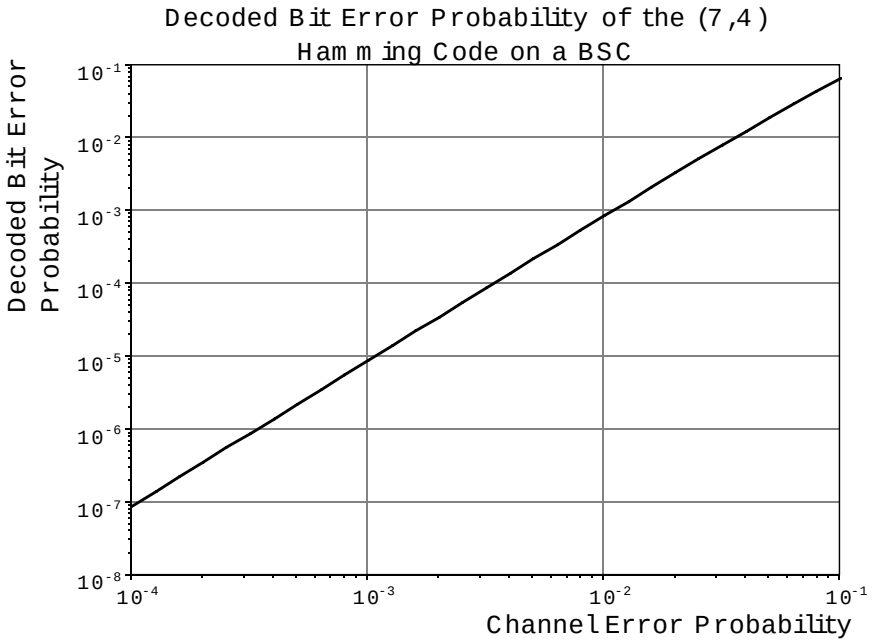


Figure 91: Bit error probability of Hamming code (as a function of the channel crossover probability).

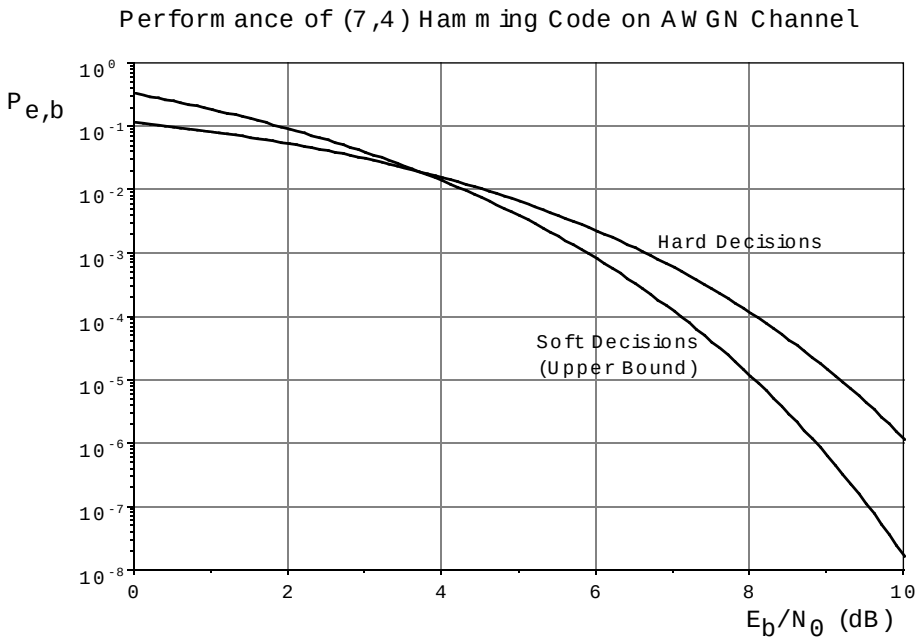


Figure 92: Bit error probability of Hamming code (as a function of E_b/N_0). (Hard decisions and soft decisions)

Cyclic Codes

Definition: A linear cyclic code is a linear block code such that every cyclic shift of a codeword is also a codeword

Notation: It is convenient to represent all codewords as polynomials

Example: If $c = (c_0, c_1, \ldots, c_{n-1})$ is a codeword of length n with $c_i \in \{0,1\}$ we will write this as

$$c(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}$$

$$xc(x) = c_0 x + c_1 x^2 + \cdots + c_{n-2} x^{n-1} + c_{n-1} x^n$$

Claim:

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

Let $c_1(x) = c_{n-1} + c_0 x + \cdots + c_{n-2} x^{n-1} = xc(x) + c_{n-1}(x^n - 1)$. Then $c_1(x)$ is

the remainder when $c(x)x$ is divided by $x^n - 1$

$$\Longrightarrow [xc(x) \equiv c_1(x)] \bmod (x^n - 1)$$

i.e. $xc(x) - c_1(x)$ is a multiple of $x^n - 1$

All polynomials have binary coefficients. Multiplication of polynomials is done as usual with mod two addition of coefficients. Consider a polynomial $g(x)$ of degree $n-k$ and a polynomial $m(x)$ of degree $k-1$

$$\begin{aligned} g(x) &= g_0 + g_1 x + \cdots + g_{n-k} x^{n-k} \\ m(x) &= m_0 + m_1 x + \cdots + m_{k-1} x^{k-1} \\ & \quad m_i \in \{0,1\}, \quad g_i \in \{0,1\} \end{aligned}$$

Consider all polynomials

$$c(x) = m(x)g(x)$$

for some $m(x), g(x)$ is fixed. There are 2^k such polynomials because there are 2^k such $m(x)$. ($c_1 = c_2 \Longrightarrow m_1 = m_2$)

Claim: The set of polynomials generated from $m(x)g(x)$ is a cyclic code if $x^n - 1$ is a multiple of $g(x)$ ($g(x)$ is a divisor of $x^n - 1$)

Proof: Consider any codeword $c(x) = m(x)g(x)$ A cyclic shift of $c(x)$ produces

$$c_1(x) = xc(x) - (x^n - 1)c_{n-1}$$

Since $c(x)$ is a multiple of $g(x)$ and $x^n - 1$ is a multiple of $g(x)$ so is $c_1(x)$. Thus a cyclic shift of any codeword is a codeword.

Consider $n = 7$. The factorization of $x^7 - 1$ over polynomials with binary coefficients is

$$x^7 - 1 = (x-1)(x^3 + x^2 + 1)(x^3 + x + 1)$$

We now consider three examples of cyclic codes of length 7. The first one is the (7,4) Hamming code, the second one is the simple single parity check code and the last one is the repetition code.

Example 1: Let

$$\begin{aligned}
g(x) &= (1+x^2+x^3) \\
m(x) &= 1 \Longrightarrow 1+x^2+x^3 \text{ is a codeword} \\
(1011000) &= c_4 \\
m(x) &= 0 \Longrightarrow 0 \text{ is a codeword} \\
(0000000) & \\
m(x) &= 1+x \\
\Longrightarrow c(x) &= (1+x^2+x^3)(1+x) \\
&= 1+x^2+x^3+x+x^3+x^4 \\
&= 1+x+x^2+x^4 = c_{11} \\
m(x) &= 1+x+x^3 \\
c(x) &= (1+x^2+x^3)(1+x+x^3) \\
&= 1+x+x^2+x^3+x^4+x^5+x^6 = c_{15}
\end{aligned}$$

So this is the (7,4) Hamming code.

Example 2: Let

$$g(x) = (1+x+x^3)$$

This is also a (7,4) Hamming code. In fact it can be found by taking the reverse of every codeword in the previous example.

Example 3: Let

$$g(x) = (1+x)$$

Codewords are of the form

$$c(x) = m(x)(x+1).$$

Notice that $c(1) = m(1)0$ implies that

$$c(1) = 0$$

so that

$$c_0 + c_1 + \cdots + c_{n-1} = 0$$

or in other words the number of ones in any codeword is even. Clearly any cyclic shift of a codeword is another codeword. This is the $(n, n-1)$ single (even) parity check code.

Example 4: Let

$$g(x) = (x^3 + x^2 + 1)(x^3 + x + 1) = (1 + x + x^2 + ... + x^6)$$

The two codewords are

$$c(x) = 0$$

and

$$c(x) = g(x) = 1 + x + x^2 + ... + x^{n-1}$$

This is the $(n, 1)$ repetition code. It is clear that every codeword is a cyclic shift of another codeword.

Introduction to Finite Fields

Everyone knows how to add, subtract, multiply and divide real numbers (assuming we are not dividing by 0). If for a set of numbers we can add, subtract, multiply and divide (except by 0) then that set of numbers is called a field. Note this implicitly requires that there be a number 0 that is the difference of two identical elements and a number 1 that is the the ratio of a number with itself. So each number has an additive inverse and each number (except 0) has a multiplicative inverse. The additive inverse of a is $-a$. The multiplicative inverse of a is a number b such that $ab = 1$.

Electrical engineers also know how to add, subtract, multiply, and divide vectors of length 2 with the components being real numbers.

$$\begin{aligned}
(a,b)+(c,d) &= (a+b,c+d)\\
(a,b)-(c,d) &= (a-b,c-d)\\
(a,b)\times(c,d) &= (ac-bd,ad+bc)\\
(a,b)/(c,d) &= (\frac{ac+bd}{c^2+d^2},\frac{bc-ad}{c^2+d^2})
\end{aligned}$$

But where did the last two equations come from? They came from actually representing the numbers as complex numbers (so the first part is the real part and the second part is the complex part). So

$$\begin{aligned}
(a+jb)(c+jd) &= ac+j(ad+bc)+j^2bd\\
&= ac-bd+j(ad+bc)
\end{aligned}$$

because $j^2=-1$. That is, j is the solution of the equation $x^2=-1$. Similarly

$$\begin{aligned}
(a+jb)/(c+jd) &= \frac{(a+jb)(c-jd)}{(c+jd)(c-jd)}\\
&= \frac{ac+j(bc-ad)-j^2bd}{c^2+d^2}\\
&= \frac{ac+bd}{c^2+d^2}+j\frac{bc-ad}{c^2+d^2}\\
&= (\frac{ac+bd}{c^2+d^2},\frac{bc-ad}{c^2+d^2})
\end{aligned}$$

Note that it is much easier to multiply and divide in the "log" domain. That is if we represent numbers in polar coordinated $(a+jb) = r_1 e^{j\theta_1}$ and $(c+jd) = r_2 e^{j\theta_2}$ then $(a+jb)(c+jd) = r_1 r_2 e^{j(\theta_1+\theta_2)}$ and $(a+jb)/(c+jd) = (r_1/r_2)e^{j(\theta_1-\theta_2)}$

Introduction to Finite Fields

Now let us repeat this for the case where instead of the components of the vector being real numbers the components are just 0 and 1. The underlying addition is done mod 2. That is $0+0=0,\ \ 0+1=1,\ \ 1+1=0$. Note that the equation $x^2+x+1=0$ does not have a solution from the set $\{0,1\}$. So, just like the complex number case let's make up a solution and call the solution α. That is $\alpha^2+\alpha+1=0$ or $\alpha^2=\alpha+1$. Now, how do we add, subtract, multiply and divide vectors of length two? First, note that the only possible vectors are $\{(0,0),(1,0),(0,1),(1,1)\}$. So there are a finite number of elements in this field. For addition and subtraction the process is to do component-wise addition and subtraction (note that subtraction is the same as addition).

For multiplication we repeat the same process used for complex numbers.

$$\begin{aligned}(a,b)\times(c,d) &= (a+\alpha b)(c+\alpha d)\\ &= ac+\alpha(ad+bc)+\alpha^2 bd\\ &= ac+\alpha(ad+bc)+(\alpha+1)bd\\ &= ac+bd+\alpha(ad+bc+bd)\\ &= (ac+bd,ad+bc+bd)\end{aligned}$$

Now consider division. Note that we can not divide by $(0,0)$. If we divide by $(1,0)$ we get the same vector. So what if we divide by $(c,1)$ where c is 0 or 1?

$$\begin{aligned}(a+\alpha b)/(c+\alpha) &= \frac{(a+\alpha b)(1+c+\alpha)}{(c+\alpha)(1+c+\alpha)}\\ &= \frac{(a+ac+\alpha(a+b+bc)+\alpha^2 b}{c(1+c)+\alpha(1+c+c)+\alpha^2}\\ &= \frac{(a+ac+\alpha(a+b+bc)+(\alpha+1)b}{\alpha+\alpha+1}\\ &= (a+b+ac)+\alpha(a+bc)\\ &= (a+b+ac,a+bc)\end{aligned}$$

Introduction to Finite Fields

We can build a table for addition and multiplication in this example.

$+$	$0=(0,0)$	$1=(1,0)$	$\alpha=(0,1)$	$\alpha+1=(1,1)$
$0=(0,0)$	0	1	α	$\alpha+1$
$1=(1,0)$	1	0	$\alpha+1$	α
$\alpha=(0,1)$	α	$\alpha+1$	0	1
$\alpha+1=(1,1)$	$\alpha+1$	α	1	0

$\times$	$0=(0,0)$	$1=(1,0)$	$\alpha=(0,1)$	$\alpha+1=(1,1)$
$0=(0,0)$	0	0	0	0
$1=(1,0)$	0	1	α	$\alpha+1$
$\alpha=(0,1)$	0	α	$\alpha+1$	1
$\alpha+1=(1,1)$	0	$\alpha+1$	1	α

Introduction to Finite Fields

We can represent the field by powers of α and 0.

00	0	$\alpha^{-\infty}$
10	1	α^0
01	α^1	α^1
11	$\alpha+1$	α^2

Consider

$$\begin{aligned}\alpha^3 &= \alpha^2(\alpha)\\ &= (\alpha+1)(\alpha)\\ &= \alpha^2+\alpha\\ &= (\alpha+1)+\alpha\\ &= 1\end{aligned}$$

So $\alpha \times \alpha^2 = \alpha^3 = 1$. Similarly $\alpha^2 \times \alpha^2 = \alpha^4 = \alpha^3\alpha = \alpha$.

Finite Fields

We can also do arithmetic on vectors of length 3 with component being 0 or 1. Consider the equation $x^3 + x + 1 = 0$. This equation does not have a solution in the binary field. So let us call α the solution. So a binary vector of length 3 corresponds to a polynomial in α of degree 2. In this case we let the first component of the vector represent the coefficient of α^2, the second component as the coefficient of α and the third component as the constant.

$$\begin{aligned} (1,1,0) &= 1\alpha^2 + 1\alpha^1 + 0 \\ (1,0,1) &= 1\alpha^2 + 0\alpha^1 + 1 \end{aligned}$$

000	0	$\alpha^{-\infty}$
001	1	α^0
010	α^1	α^1
011	$\alpha + 1$	α^3
100	α^2	α^2
101	$\alpha^2 + 1$	α^6
110	$\alpha^2 + \alpha$	α^4
111	$\alpha^2 + \alpha + 1$	α^5

Arithmetic

$$\begin{aligned}\alpha^7 &= \alpha^6\alpha \\ &= (\alpha^2+1)\alpha \\ &= \alpha^3+\alpha \\ &= \alpha+1+\alpha \\ &= 1\end{aligned}$$

Arithmetic

$$\begin{aligned}(\alpha^2+1)(\alpha+1) &= \alpha^6\alpha^3 \\ &= \alpha^9 \\ &= \alpha^7\alpha^2 \\ &= \alpha^2\end{aligned}$$

$$\begin{aligned}(\alpha^2+1)(\alpha^2+1) &= \alpha^6\alpha^6 \\ &= \alpha^{12} \\ &= \alpha^7\alpha^5 \\ &= \alpha^5 \\ &= \alpha^2+\alpha+1\end{aligned}$$

Reed-Solomon Codes

Reed-Solomon codes are widely used in digital communication systems. For example, the compact disc player uses a two Reed-Solomon codes. These codes are nonbinary. That is the code symbols are from an alphabet of size larger than two. For example, the (255,223) NASA standard code contains 223 information symbols that are 8 bit bytes. So one codeword contains 223 times 8 or 1784 information bits. These 223 information bytes or symbols are mapped by the encoder into 255 code symbols which are again eight bit bytes.

Reed-Solomon codes are linear codes so the parity check equations that apply to linear codes also apply to Reed-Solomon codes.

$$d_{\min} = N - K + 1 \quad (\therefore \text{ MDS code})$$

These codes are used in compact digital discs, spread-spectrum systems, computer memories.

For example the (7,4) Reed-Solomon code has symbols that are three bit bytes. Let us represent these symbols in the following way. This representation is called a finite field. (In this case the equation $x^3 + x + 1 = 0$ does not have a solution in the set $\{0, 1\}$). We let α be that solution.

000	0	$\alpha^{-\infty}$
001	1	α^0
010	α^1	α^1
011	$\alpha + 1$	α^3
100	α^2	α^2
101	$\alpha^2 + 1$	α^6
110	$\alpha^2 + \alpha$	α^4
111	$\alpha^2 + \alpha + 1$	α^5

Notice that $\alpha^7 = 1$ so when multiplying two elements of this field we can add the exponents of α modulo 7.

So when doing arithmetic with 3 bit bytes, adding two elements corresponds to bit-wise modulo two addition of the components and multiplication is best done by representing the two symbols as a power of α and then adding the exponents modulo 7.

The encoding is done as follows. There are four information symbols $i_1,\ i_2,\ i_3,\ i_4$. There are three parity symbols $p_1,\ p_2,\ p_3$. Each of these is a three bit byte.

The parity symbols are determined from the following equation.

$$\begin{aligned} p_1 &= \alpha^6 i_1 + \alpha^6 i_2 + \alpha^3 i_3 + \alpha i_4 \\ p_2 &= \alpha i_1 + \alpha^2 i_2 + \alpha^4 i_3 + i_4 \\ p_3 &= \alpha^6 i_1 + \alpha^5 i_2 + \alpha^5 i_3 + \alpha^2 i_4 \end{aligned}$$

For example, the four information symbols $i_1 = \alpha^5, i_2 = \alpha^3, i_3 = \alpha^2, i_4 = \alpha^0$ are mapped into the parity bits $p_1 = \alpha^5, p_2 = \alpha^4, p_3 = 1$. Thus the information bits (111 011 100 001) are mapped into the channel bits (111 011 100 001 111 110 001) for transmission.

Example: Consider the (4,2) Reed Solomon code over GF(5)=$\{0,1,2,3,4\}$. Addition and multiplication are done mod 5 so that $2\times 3 = 6 = 1 \bmod 5$ and thus $2^{(-1)} = 3 \bmod 5$. The generator polynomial is

$$g(x) = (x-1)(x-2) = x^2 - 3x + 2 = x^2 + 2x + 2$$

The codewords are

$$\begin{aligned}
(0+0x)g(x) &= 0x^3 + 0x^2 + 0x + 0 \\
(1+0x)g(x) &= 0x^3 + 1x^2 + 2x + 2 \\
(2+0x)g(x) &= 0x^3 + 2x^2 + 4x + 4 \\
(3+0x)g(x) &= 0x^3 + 3x^2 + 1x + 1 \\
(4+0x)g(x) &= 0x^3 + 4x^2 + 3x + 3 \\
(0+1x)g(x) &= 1x^3 + 2x^2 + 2x + 0 \\
(1+1x)g(x) &= 1x^3 + 3x^2 + 4x + 2 \\
(2+1x)g(x) &= 1x^3 + 4x^2 + 1x + 4
\end{aligned}$$

$$\begin{aligned}
(3+1x)g(x) &= 1x^3 + 0x^2 + 3x + 1 \\
(4+0x)g(x) &= 1x^3 + 1x^2 + 0x + 3 \\
(0+2x)g(x) &= 2x^3 + 4x^2 + 4x + 0 \\
(1+2x)g(x) &= 2x^3 + 0x^2 + 1x + 2 \\
(2+2x)g(x) &= 2x^3 + 1x^2 + 3x + 4 \\
(3+2x)g(x) &= 2x^3 + 2x^2 + 0x + 1 \\
(4+2x)g(x) &= 2x^3 + 3x^2 + 2x + 3 \\
(0+3x)g(x) &= 3x^3 + 1x^2 + 1x + 0 \\
(1+3x)g(x) &= 3x^3 + 2x^2 + 3x + 2 \\
(2+3x)g(x) &= 3x^3 + 3x^2 + 0x + 4 \\
(3+3x)g(x) &= 3x^3 + 4x^2 + 2x + 1 \\
(4+3x)g(x) &= 3x^3 + 0x^2 + 4x + 3 \\
(0+4x)g(x) &= 4x^3 + 3x^2 + 3x + 0
\end{aligned}$$

$$(1+4x)g(x) = 4x^3+4x^2+0x+2$$
$$(2+4x)g(x) = 4x^3+0x^2+2x+4$$
$$(3+4x)g(x) = 4x^3+1x^2+4x+1$$
$$(4+4x)g(x) = 4x^3+2x^2+1x+3$$

The codewords are easily seen to be cyclic shifts of the following vectors

0 0 0 0

0 1 2 2

0 2 4 4

0 3 1 1

0 4 3 3

1 3 4 2

1 4 1 4

2 3 2 3

Note that because this is a linear code the minimum distance is the minimum weight which is 3. Thus this code can correct one error. Suppose the code is used on a channel and suppose one error is made. Then

$$r(x) = c(x)+e(x)$$

where

$$e(x) = e_j x^j$$

This implies the error has magnitude e_j and occurred in the j-th position. Because $c(x) = g(x)i(x)$ for some polynomial $i(x)$ it is clear that

$$\begin{aligned} r(1) &= g(1)i(1)+e(1) = e(1) = e_j \\ r(2) &= g(2)i(2)+e(2) = e(2) = 2^j e_j \end{aligned}$$

Thus the magnitude of the error is just $r(1)$ and the location of the error can be determined from

$$\log_2(\frac{r(2)}{r(1)})$$

For example suppose we receive

$$r(x) = 4x^3 + 3x^2 + 4x^1 + 1$$

Then $r(1) = 4+3+4+1 = 2$ and $r(2) = 2+2+3+1 = 3$. Thus the error magnitude is 2. The error location is determined from

$$r(2)/r(1) = 3/2 = 3 \cdot 2^{-1} = 3 \cdot 3 = 4 = 2^2$$

Thus the error polynomial is $e(x) = 2x^2$. Thus

$$c(x) = r(x) - e(x) = 4x^3 + 3x^2 + 4x^1 + 1 - 2x^2 = 4x^3 + 1x^2 + 4x^1 + 1$$

which is indeed a codeword.

Here is $GF(2^4)$ using primitive polynomial $x^4 + x + 1$.

$\alpha^{-\infty}$	0
α^0	1
α^1	α
α^2	α^2
α^3	α^3
α^4	$\alpha+1$
α^5	$\alpha^2+\alpha$
α^6	$\alpha^3+\alpha^2$
α^7	$\alpha^3+\alpha+1$
α^8	α^2+1
α^9	$\alpha^3+\alpha$
α^{10}	$\alpha^2+\alpha+1$
α^{11}	$\alpha^3++\alpha$
α^{12}	$\alpha^3+\alpha^2+\alpha+1$
α^{13}	$\alpha^3+\alpha^2+1$
α^{14}	α^3+1

Bounded Distance Decoding

A code with minimum distance $d_{\min}$ can correct any error pattern of weight $\lfloor \frac{d\min-1}{2} \rfloor$ or less. If more than $\lfloor \frac{d\min-1}{2} \rfloor$ errors occur a decoder may or may not be able (in theory) to correct it. A bounded distance decoder is a decoder that corrects any error pattern of weight ℓ or less $\left(\ell \leq \lfloor \frac{d\min-1}{2} \rfloor\right)$ and either fails or errors otherwise. The decoder fails if no codeword is within distance ℓ or less of the received vector. The decoder makes an error if one codeword is distance ℓ or less from the received vector. Most cyclic codes can be decoded efficiently only by a bounded distance decoder. Note that a bounded distance decoder is not a MAP or maximum likelihood decoder since some received vectors are not decoded into the closest codeword. In fact there are received vectors that are not mapped (decoded) into any codeword. These cause the decoder to fail. The decoding complexity of a bounded distance decoder of a cyclic code is roughly $(n-k)^2$, which is much less than the complexity of syndrome decoding or brute force decoding for reasonable values of n and k.

For a bounded distance decoder with error correction capability ℓ the probability of correct decoding is

$$\begin{aligned} P_{CD} &= P\{\ell \text{ or fewer errors}\} \\ &= \sum_{i=1}^{\ell} \binom{n}{i} p^i (1-p)^{n-i} \end{aligned}$$

The probability of incorrect decoding (decoding error) can be upper bounded by assuming any error pattern of weight $d-\ell$ or larger is within distance ℓ of some codeword

$$P_{ICD} \leq \sum_{i=d-\ell}^{n} \binom{n}{i} p^i (1-p)^{n-i}$$

The probability of decoder failure or error detection (no codeword within distance ℓ of received vector) can be lower bounded as follows

$$\begin{aligned} P_{CD} &= 1 - P_{ED} - P_{ICD} \\ P_{ED} &= 1 - P_{ICD} - P_{CD} \end{aligned}$$

$$\geq \quad 1 - \sum_{i=d-\ell}^{n} \binom{n}{i} p^i (1-p)^{n-i} - \sum_{i=0}^{\ell} \binom{n}{i} p^i (1-p)^{n-i}$$

$$= \quad \sum_{i=\ell+1}^{d-\ell-1} \binom{n}{i} p^i (1-p)^{n-i}$$

The last equality follows since

$$\sum_{i=0}^{n} \binom{n}{i} p^i (1-p)^{n-i} = 1 \, .$$

The bit error probability can be upper bounded by assuming that all incorrect decoding events caused by i errors cause the decoder to insert at most $i+\ell$ errors.

$$P_b \leq \sum_{i=d-\ell}^{n} \frac{i+\ell}{n} \binom{n}{i} p^i (1-p)^{n-i}.$$

Note: As the rate of the code decreases the error probability decreases.

Fact: For binary symmetric channel for any $\varepsilon > 0$ there exist codes with error

probability $< \varepsilon$ provided the code rate is less than the capacity,

$$r < C = 1 - H_2(p)$$

$$H_2(p) = -p \log_2 p - (1-p) \log_2 (1-p)$$

Up to now we have only considered binary codes transmitted over a binary symmetric channel. However there are codes with symbol alphabet size larger than two. The most important such code is the Reed-Solomon code. let N and K be the length and dimension (number of information symbols of a Reed-Solomon code). The alphabet size is 2^m where m is an integer. The choice of m determines the possible lengths of the RS code. For example if $m = 5$ then N can have length $2^5 + 1$ or less. Typically it is either 31 or 32. The code can correct a number of symbol errors determined by the minimum distance. An (N,K) Reed-Solomon code has minimum Hamming distance $d_{\min} = N - K + 1$ and can correct $\lfloor (d_{\min} - 1)/2 \rfloor$ symbol errors. The Reed-Solomon code is also a cyclic code and can be (bounded distance) decoded with an efficient algorithm (for a hard decision channel). The symbol error

probability is closely approximated by the error probability of a bounded distance decoder.

This code has many applications. One example is the Compact Disc system. In this case actually two slightly different Reed-Solomon codes are used. The compact disc stores binary data. Eight bits of data are group together to form a symbol. The alphabet size is thus 256. If one of the bits in a symbol is in error then the symbol is in error. However, more than one bit error within a symbol still just cause a single symbol to be in error. In this sense, RS codes are burst error correcting codes. The CD system uses a (32,28) code with symbols of size 2^8 and a (28,24) code with the same size symbols. NASA has a standard for a (255,223) Reed-Solomon code that can correct up to 16 symbol errors (symbol size =256).

Let us compare a (7,4) Binary Hamming code with a (7,4) Reed- Solomon code with symbols size 8. The Hamming code has 16 codewords out of a total of 128 possible vectors (12.5 percent of the vectors are codewords). The Reed-Solomon code has $8^4 = 4096$ codewords and the total number of vectors

is $8^7 = 2097152$ so 0.195 percent of vectors are codewords. The (7,4) Hamming code can correct one error ($d_{\min} = 3$) and each decoding region contains 8 vectors. The decoding regions surrounding the 16 codewords account for $8 \times 16 = 128$ vectors; 100 percent of the vectors in the space. The (7,4) Reed-Solomon code has a minimum distance 4 and also can correct one symbol error. The decoding region of a bounded distance decoder contains $1 + 7 \times 7 = 50$ vectors. The total number of vectors in a decoding region is thus $50 \times 4096 = 204800$ of the total of 2097152 vectors or 9.76 percent of the total number of vectors. Thus given that the number of errors is greater than the correcting capability for a Reed-Solomon code, the received vector is likely not to be in the decoding region of any codeword.

Soft and Hard Decision Decoding of Block Codes for an Additive White Gaussian Noise Channel

So far we have only consider decoding algorithms for block codes when the channel is the binary symmetric channel. These algorithms can be used when the channel is the additive Gaussian channel. Basically, the idea is to convert the additive Gaussian channel into a binary symmetric channel. To do we restrict the input to the channel to be from the set $\{+\sqrt{E}, -\sqrt{E}\}$and use the following mapping from the set $\{0,1\}$ into the set $\{+\sqrt{E}, -\sqrt{E}\}$

$$0 \to \sqrt{E}$$

and

$$1 \to -\sqrt{E}.$$

If the allowable inputs to the additive Gaussian channel are $\{+\sqrt{E}, -\sqrt{E}\}$

then we can use a binary block code to transmit information over a additive Gaussian channel. Thus each codeword is transmitted as a sequence of $\pm\sqrt{E}$'s. The output of the channel is the input plus a Gaussian noise variable.

$$r_i = c_i + n_i$$

where c_i is the ith component of the codeword (after the mapping) and n_i is Gaussian noise.

The optimal decision rule is to decide codeword $\mathbf{c_j}$ was transmitted if it is closest in Euclidean distance to what was received. This can be very difficult for block codes. We can use decoding algorithms for a binary symmetric channel by making a hard decision of each of the symbols. That is, we decide that the i-th bit was $+\sqrt{E}$ if $r_i > 0$ and the i-th bit was $-\sqrt{E}$ if $r_i < 0$. We can then do the inverse map

$$1 \to \sqrt{E}$$

and

$$-1 \to -\sqrt{E}.$$

The probability that a channel output is received as +1 given the input was $-\sqrt{E}$ is the same as the probability that a channel output is received as $-\sqrt{E}$ given the input was $+\sqrt{E}$ (due to the symmetry of the noise distribution). We can then do the inverse map

$$\sqrt{E} \to 0$$

and

$$-\sqrt{E} \to 1.$$

and get a binary symmetric channel for which known algorithms can be used for decoding. This is called hard decision decoding since on every symbol of the received vector we make a decision as to what the transmitted symbol was.

The optimal decoding algorithm for the additive Gaussian channel (also called soft decision decoding) compares the received vector with every possible transmitted codeword and chooses the codeword that is closest (in Euclidean distance) to what was received. For example, if the repetition code of length 3 is used and the vector

$$(1.5, -.3, -.4)$$

is received the optimal (soft decision) algorithm would choose the codeword $(+\sqrt{E}, +\sqrt{E}, +\sqrt{E})$ as the transmitted codeword. (Note that it is easy to show that the closest codeword to the received vector is also the codeword with largest correlation with the received vector).

A hard decision algorithm would first make a decision on each symbol and produce the vector $(+\sqrt{E}, -\sqrt{E}, -\sqrt{E})$ then find the codeword closest in Hamming distance
$(-\sqrt{E}, -\sqrt{E}, -\sqrt{E})$. For this received vector the two decoding algorithms differ in their output.

The binary symmetric channel created by the hard decision device has crossover probability that depends on the energy E and the noise variance $N_0/2$ as

$$p = Q(\sqrt{\frac{2E}{N_0}}).$$

If E_b is the energy transmitted per information bit then the k information bits of a codeword use nE joules to transmit them. Thus the energy per

information bit is related to the energy per channel bit by

$$E_b = nE/k = E/r$$

where $r = k/n$ is the rate of the code. Thus if we make comparison between an uncoded system and a coded system with the same energy per information bit then each symbol of the coded system must have smaller energy so that p will be larger for a coded system than an uncoded system. Coding will be useful only if the capability to correct errors overcomes this increase in error probability caused by reducing the energy of each transmitted signal. Indeed, coding can provide significant gains over uncoded systems, even with the penalty of reduced energy per channel bit. The penalty of coding discussed so far is 1) Coding requires either higher bandwidth compared to an uncoded system of the same data rate or lower data rate compared to an uncoded system of the same bandwidth and 2) increased complexity.

Block Coding Summary

First, we defined a code as a set of M vectors of length n to be used over a channel. Two important examples of channels were the additive white Gaussian noise channel and the binary symmetric channel.

Second, we derived the receiver that minimizes the probability that a wrong codeword is chosen. This is called the maximum a posteriori probability rule. If all the codewords are equally likely then it is equivalent to the maximum likelihood rule. For a binary symmetric channel this means finding the codeword closest to the received vector in Hamming distance. For an additive Gaussian channel the optimum receiver is to find the codeword closest in Euclidean distance to the received vector.

These algorithms, in general require, M comparisons of the received vector with codewords. For practical purposes this is too complex. To overcome this complexity we added some structure to the code. We first imposed linearity on the code. A linear code has $M = 2^k$ codewords of length n such that the sum of two codewords is also a codeword. For the BSC we derived the syndrome decoding algorithm for linear codes. The complexity of this algorithm was $2^{(n-k)}$. This can be a significant reduction if $n-k$ is small but k and n are large. However, for k large and $n-k$ large the complexity is still large. We imposed further structure on the code by requiring the code to be cyclic, that is require a cyclic shift of a codeword to be a codeword. While we did not derive a simpler algorithm for decoding cyclic codes, algorithms that require on the order of $(n-k)^2$ complexity (for the BSC) exist.

For the additive Gaussian channel, cyclic linear codes do not simplify the decoding algorithms. We can turn the additive Gaussian channel into a BSC by making hard decisions on each code symbol. This degrades the performance of the system relative to a system that does not make hard decisions but allows the use of simple algorithms. For an additive Gaussian channel with an energy constraint of E_b per information bit, we must use part of this energy for the redundant bits transmitted. That is the energy per channel symbol must be reduced by the rate of the code, i.e. $E_b = E/r$. For the hard decision channel this increases the crossover probability of the resulting binary symmetric channel. Coding will only be useful if the error correction capability is sufficient to correct the additional errors caused by the increased crossover probability of the channel relative to an uncoded system.

We examined the error correcting capability in terms of the minimum distance of a binary code. We also derived expressions for the codeword error probability and the bit error probability.

Note that the repetition codes of length n has minimum distance $d_{min} = n$ and rate $1/n$. Thus as n becomes large the error correcting capability becomes large at the expense of very low rate. For a fixed channel error probability p of the binary symmetric channel, the bit error probability of the repetition code (at the decoder output) goes to 0 as n becomes larger but at the expense of a lower rate (and thus larger bandwidth). The Hamming codes [(7,4),(15,11),...] have $d_{min} = 3$ but the rate becomes close to one. For these codes the error probability does not go to zero as the code length becomes larger. The question of the existence of codes for which the error probability goes to zero but the rate does not was answered by Shannon in 1948. He showed that such codes exist if the rate was less than the capacity of the channel.

The improvement in performance for the additive Gaussian channel when hard decision are not made motivates the consideration of a different type of code, namely convolutional codes, which can be decoded without much complexity.