EEE8099 – Information Theory & Coding
EEE8104 – Digital Communications

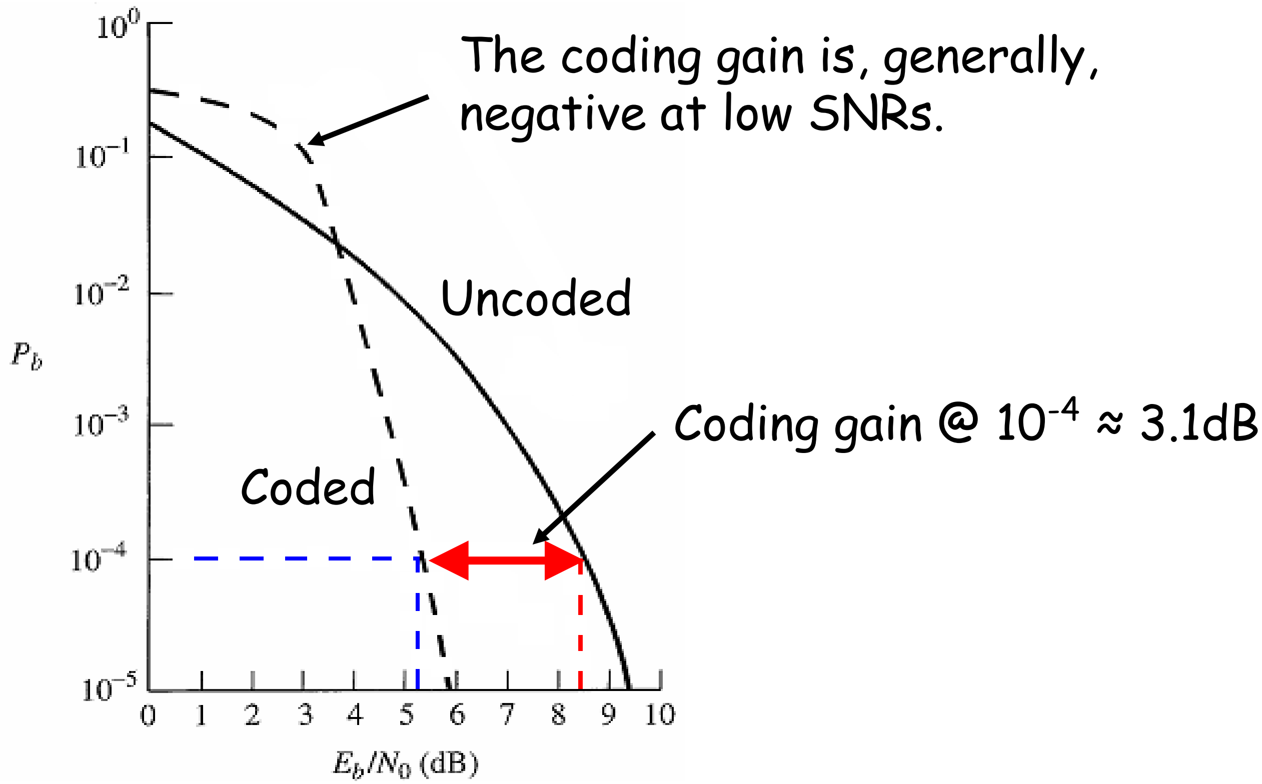
S. Le Goff

School of Engineering @ Newcastle University

Part 5
Error-Correcting Codes

Rationale of error-correcting coding

Goal: Achieve a target P_{eb} using a lower SNR.



Use E_b/N_0, not E_s/N_0 for the x axis

We need to perform a fair comparison between the uncoded and coded systems.

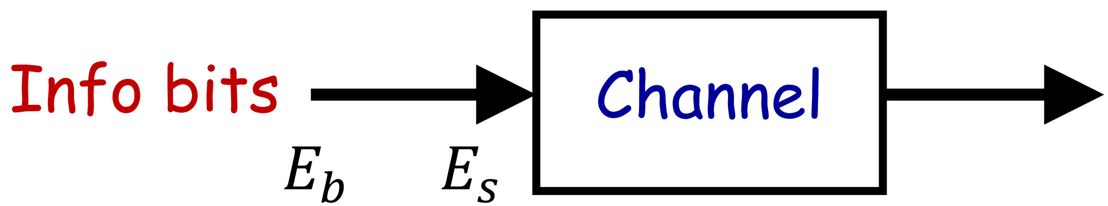
The bit error probabilities obtained at the receiver output for both systems must be computed/simulated under the same conditions.

We must assume that each info bit is given in both cases the same energy, i.e. the same SNR $\frac{E_b}{N_0}$.

Use E_b/N_0, not E_s/N_0 for the x axis

The bit error probabilities obtained with uncoded and coded systems must be compared at the same SNR per info bit, $\frac{E_b}{N_0}$.

Uncoded system



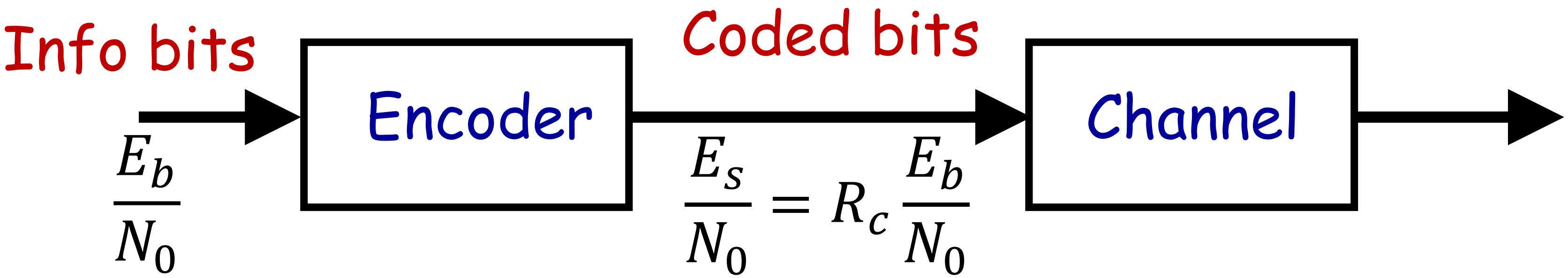
$$\frac{E_b}{N_0} = \frac{E_s}{N_0}$$

For a BPSK, AWGN channel, the noise variance is given by $\sigma^2 = \frac{1}{2\frac{E_s}{N_0}} = \frac{1}{2\frac{E_b}{N_0}}$.

Use E_b/N_0, not E_s/N_0 for the x axis

The bit error probabilities obtained with uncoded and coded systems must be compared at the same SNR per info bit, $\frac{E_b}{N_0}$.

Coded system



For a BPSK, AWGN channel, the noise variance is now given by $\sigma^2 = \frac{1}{2\frac{E_s}{N_0}} = \frac{1}{2R_c\frac{E_b}{N_0}}$.

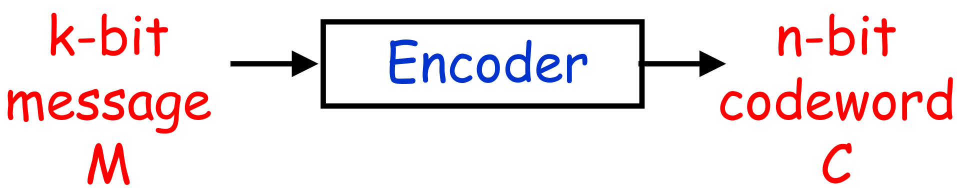
Use E_b/N_0, not E_s/N_0 for the x axis

A fair comparison between uncoded and coded systems implies that the SNR, $\frac{E_s}{N_0}$, per transmitted bit over the channel is higher for the uncoded system than for the coded system (because $R_c < 1$).

This puts the coded system at a disadvantage, but this is the correct way to compare error performances of uncoded and coded schemes.

Two types of codes

Block codes



No memory from one message to the other

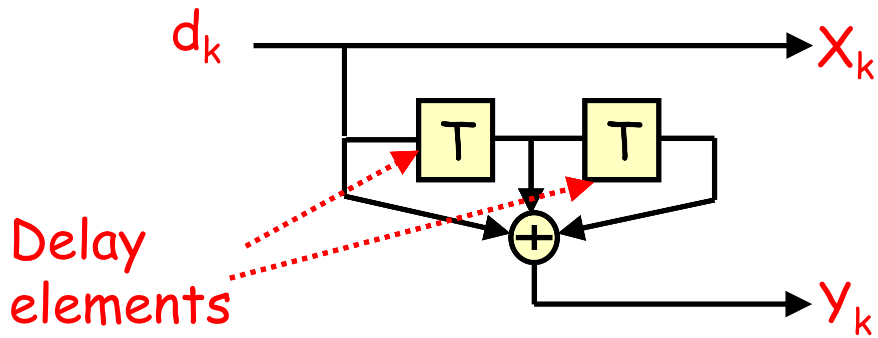
Message M

Codeword C

Coding rate $R_c = \frac{k}{n}$

$R_c < 1$

Convolutional codes



d_k: info bit at time kT

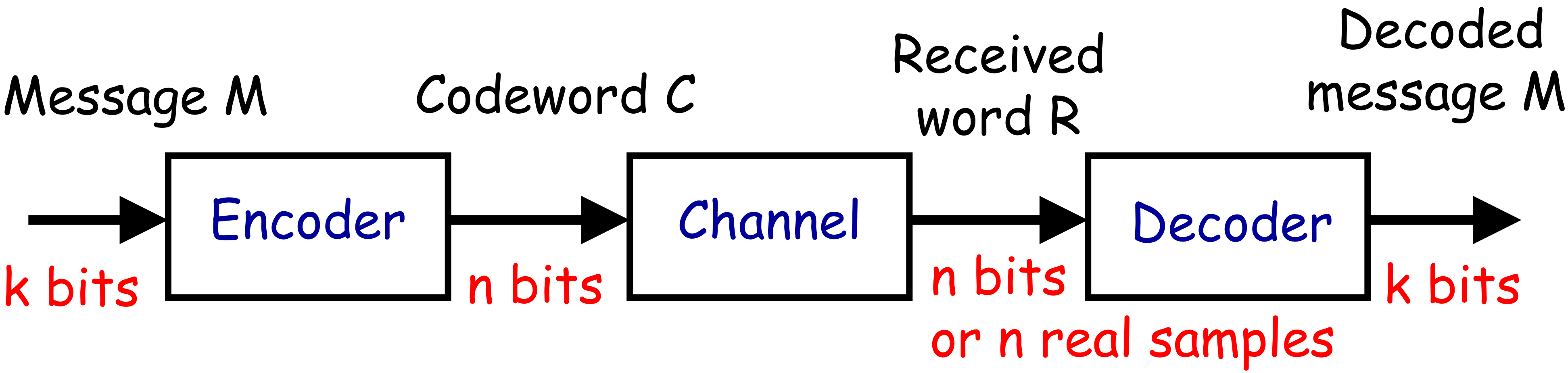
X_k, Y_k: coded bits at kT

Coding rate $R_c = \frac{1}{2}$

R_c can be increased using a technique called 'puncturing'

Block and convolutional codes are linear codes.

The big picture (block codes)



$$\left.\begin{aligned} M &= (m_0, m_1, m_2, \dots, m_{k-1}) \\ C &= (c_0, c_1, c_2, \dots, c_{n-1}) \\ M' &= (m_0', m_1', m_2', \dots, m_{k-1}') \end{aligned}\right\} \text{Binary vectors}$$

$$R = (r_0, r_1, r_2, \dots, r_{n-1}) \rightarrow$$ Binary vector (BSC) or vector of real samples (BPSK, AWGN channel).

The big picture (block codes)

Coding rate: $R_c = \frac{k}{n}$ $\quad$ ($R_c < 1$ since $k < n$)

The use of an error-correcting code increases the bandwidth by a factor $\frac{1}{R_c} = \frac{n}{k}$ for a given info bit rate OR decreases the info bit rate by a factor $\frac{1}{R_c} = \frac{n}{k}$ for a given bandwidth.

If bandwidth/info bit rate is a critical parameter, make sure to keep R_c as close to the unit as possible.

The big picture (block codes)

The Hamming distance between two binary words C_1 and C_2 is defined as the number of positions at which these two vectors are different.

Ex: $C_1 = (0\ 1\ 1\ 0\ 1\ ), C_2 = (1\ 1\ 1\ 1\ 0\ ) \rightarrow d_H(C_1, C_2\ ) = 3.$

Since there are 2^k possible messages M, we need to use a subset of 2^k codewords C among the 2^n (> 2^k) possible codewords C.

The big picture (block codes)

Our goal is <u>mainly</u> to choose the set of 2^k codewords so that the minimum Hamming distance between these codewords is maximal (to be demonstrated later).

Block codes are (almost) always linear codes.

A linear block code is completely defined using a $k \times n$ generator matrix G so that $C = M \cdot G$.

Each codeword C is obtained by multiplying a message M by the generator matrix G.

Linear block codes

Ex 1: (n = 7, k = 4) Hamming code

Modulo-2 addition:

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0

$$G = \begin{bmatrix} 1\,0\,0\,0\,1\,1\,0 \\ 0\,1\,0\,0\,0\,1\,1 \\ 0\,0\,1\,0\,1\,1\,1 \\ 0\,0\,0\,1\,1\,0\,1 \end{bmatrix} \rightarrow$$

$c_0 = m_0$

$c_1 = m_1$

$c_2 = m_2$

$c_3 = m_3$

$c_4 = m_0 + m_2 + m_3$

$c_5 = m_0 + m_1 + m_2$

$c_6 = m_1 + m_2 + m_3$

A linear block code is completely defined by a set of n linear equations.

Linear block codes

Ex 2: (n = 5, k = 2) code

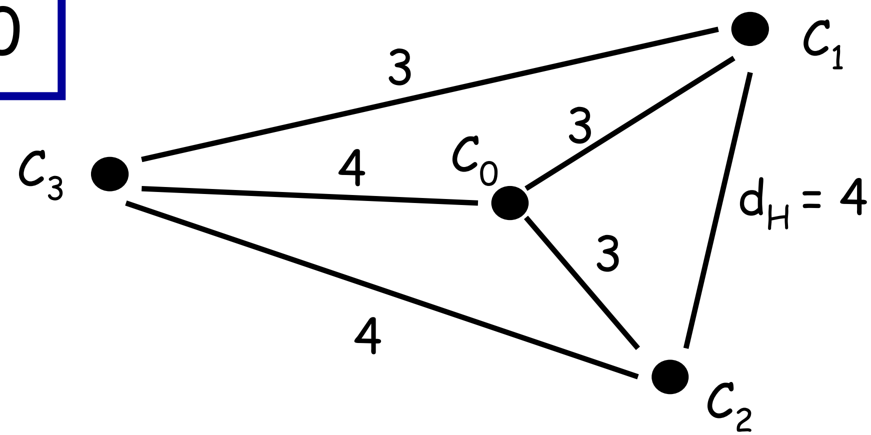
$$G = \begin{bmatrix} 1\,0\,1\,0\,1 \\ 0\,1\,0\,1\,1 \end{bmatrix} \rightarrow \begin{aligned} c_0 &= m_0 \\ c_1 &= m_1 \\ c_2 &= m_0 \\ c_3 &= m_1 \\ c_4 &= m_0 + m_1 \end{aligned}$$

Let us have a closer look at this code.

An example: (5, 2) code

m_0	m_1	c_0	c_1	c_2	c_3	c_4
0	0	0	0	0	0	0
0	1	0	1	0	1	1
1	0	1	0	1	0	1
1	1	1	1	1	1	0

$C_0 = (00000)$
$C_1 = (01011)$
$C_2 = (10101)$
$C_3 = (11110)$



Using the (5, 2) code over a BSC

Assume C_0 = (00000) is transmitted over a BSC, and the corresponding channel output is R = (00001), which indicates that there is a transmission error in the received binary word.

Can the decoder detect and even correct this error?

Over BSC, the decoder starts by computing the 4 Hamming distances between R and the 4 possible codewords (which are all known to the receiver):

Using the (5, 2) code over a BSC

The results are $d_H(R, C_0) = 1$, $d_H(R, C_1) = 2$, $d_H(R, C_2) = 2$, and $d_H(R, C_3) = 5$.

The decoder sees that R is not a valid codeword, which implies that there has been at least one transmission error.

The decoder must now take a decision regarding the transmitted codeword. To this end, it uses the maximum-likelihood (ML) decoding algorithm.

Using the (5, 2) code over a BSC

The decision rule over BSC consists of choosing the codeword which is at minimum Hamming distance from R
→ The decoder chooses C_0, and thus selects the right codeword despite the transmission error.

In this case, the decoder has been able, using ML decoding, to determine that the 5th bit in R was erroneous: it has corrected one error in the received 5-bit word.

We have M = (00), C = (00000), and M' = (00): Perfect!

Using the (5, 2) code over a BSC

Assume now C_0 = (00000) is transmitted over a BSC, and the corresponding channel output is R = (00011), which indicates that there are now two transmission errors in the received binary word.

Can the decoder detect and even correct both errors using the ML decoding algorithm?

Once again, the decoder starts by computing the 4 Hamming distances between R and the 4 possible codewords:

Using the (5, 2) code over a BSC

The results are $d_H(R, C_0) = 2$, $d_H(R, C_1) = 1$, $d_H(R, C_2) = 3$, and $d_H(R, C_3) = 4$.

The decoder sees that R is not a valid codeword, and thus concludes that there has been <u>at least one</u> transmission error.

The decoder chooses C_1, and thus select the wrong codeword this time
→ The decoder has NOT been able to correct both errors in the received 5-bit word.

Using the (5, 2) code over a BSC

So, we have M = (00), C = (00000), and M' = (01).

As M is not identical to M', we conclude that the decoder has failed to do its job properly.

We could show that, over the BSC, our (5, 2) code can correct <u>up to one error</u> in a received 5-bit word.

Error-correction power of a code

Consider a code for which the minimal Hamming distance between any pair of codewords is d_{min}.

$$C_i \xleftrightarrow{\quad d_{min} \quad} C_j \qquad i, j \in \{0, \dots, 2^k - 1\},\ i \neq j$$

Over a BSC, the maximum number of errors that can be corrected in a received n-bit word R is the error-correction power t computed as

$$t = Int\left(\frac{d_{min} - 1}{2}\right).$$

Error-correction power of a code

Ex: $d_{min} = 2 \ \rightarrow t = 0, d_{min} = 3 \ \rightarrow t = 1,$
$d_{min} = 4 \ \rightarrow t = 1, d_{min} = 5 \ \rightarrow t = 2,$
$d_{min} = 6 \ \rightarrow t = 2, d_{min} = 7 \ \rightarrow t = 3,$
$d_{min} = 8 \ \rightarrow t = 3, d_{min} = 9 \ \rightarrow t = 4 \ldots$

Clearly, over a BSC, increasing the minimum distance d_{min} between codewords, results in a better error correction capability.

Will this also be true if the code is used over a BPSK, AWGN channel?

Examples of linear block codes

Ex 1: Repetition code

n odd, $k = 1$, $G = [1 \quad \dots \quad 1]$

$C = (c_0, c_1, c_2, \dots, c_{n-1})$ with

$$c_0 = c_1 = c_2 = \cdots = c_{n-1} = m_0$$

$d_{min} = n \rightarrow t = \frac{n-1}{2}$

Low coding rate since $R_c = \frac{1}{n}$. ☹

Examples of linear block codes

(3, 1) repetition code
Two possible codewords $C_0 = (000)$ and $C_1 = (111)$, $R_c = \frac{1}{3}$, $d_{min} = 3 \rightarrow t = 1$.

(5, 1) repetition code
Two possible codewords $C_0 = (00000)$ and $C_1 = (11111)$, $R_c = \frac{1}{5}$, $d_{min} = 5 \rightarrow t = 2$.

.

(7, 1) repetition code
Two possible codewords $C_0 = (0000000)$ and $C_1 = (1111111)$, $R_c = \frac{1}{7}$, $d_{min} = 7 \rightarrow t = 3$.

.

Examples of linear block codes

Ex 2: Parity-check code

$$n = k + 1,\ G = \left[I_{k\times k} \begin{matrix} 1 \\ \vdots \\ 1 \end{matrix} \right]$$

The number of 1s in any codeword must be even.

$c_0 = m_0,\ c_1 = m_1,\ \ldots,\ c_{k-1} = m_{k-1},\ c_n = m_0 + m_1 + \cdots + m_{k-1}$

$d_{min} = 2\ \rightarrow t = 0$ → Not able to correct errors. Just able to detect one error in a word of n bits.

High coding rate since $R_c = \frac{k}{k+1}$. ☺

Examples of linear block codes

(3, 2) Parity-check code

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

$c_0 = m_0, c_1 = m_1, c_2 = m_0 + m_1$

Four possible codewords $C_0 = (000)$, $C_1 = (011)$, $C_2 = (101)$, $C_3 = (110)$, $R_c = \frac{2}{3} \sim 0.67$, $d_{min} = 2 \rightarrow t = 0$.

The number of 1s in any codeword must be even.

Examples of linear block codes

(4, 3) Parity-check code

$$G = \begin{bmatrix} 1001 \\ 0101 \\ 0011 \end{bmatrix}$$

$c_0 = m_0, c_1 = m_1, c_2 = m_2, c_3 = m_0 + m_1 + m_2$

Eight possible codewords $C_0 = (0000)$, $C_1 = (0011)$, $C_2 = (0101)$, $C_3 = (0110)$, $C_4 = (1001)$, $C_5 = (1010)$, $C_6 = (1100)$, $C_7 = (1111)$, $R_c = \frac{3}{4} = 0.75$, $d_{min} = 2 \rightarrow t = 0$.

The number of 1s in any codeword must be even.

Examples of linear block codes

Ex 3: (7, 4) Hamming code

M	C
0 0 0 0	0 0 0 0 0 0 0
0 0 0 1	0 0 0 1 1 0 1
0 0 1 0	0 0 1 0 1 1 1
0 0 1 1	0 0 1 1 0 1 0
0 1 0 0	0 1 0 0 0 1 1
0 1 0 1	0 1 0 1 1 1 0
0 1 1 0	0 1 1 0 1 0 0
0 1 1 1	0 1 1 1 0 0 1

M	C
1 0 0 0	1 0 0 0 1 1 0
1 0 0 1	1 0 0 1 0 1 1
1 0 1 0	1 0 1 0 0 0 1
1 0 1 1	1 0 1 1 1 0 0
1 1 0 0	1 1 0 0 1 0 1
1 1 0 1	1 1 0 1 0 0 0
1 1 1 0	1 1 1 0 0 1 0
1 1 1 1	1 1 1 1 1 1 1

$$R_c = \frac{4}{7} \sim 0.57,\ d_{min} = 3 \ \rightarrow t = 1$$

Minimum Hamming distance of a linear code

How to easily determine the minimum Hamming distance d_{min} of a linear block code?

Linear block codes have some interesting properties:

1. The all-zero binary word (C_0) is always a valid codeword.

Demonstration: The all-zero k-bit word M_0 is obviously a possible message for any code. By multiplying M_0 by any generator matrix G, we obtain the all-zero codeword: $C_0 = M_0 \cdot G$.

Minimum Hamming distance of a linear code

2. The sum of two codewords is another codeword.

Demonstration: By summing two arbitrary messages M_i and M_j, $i, j \in \{0, \dots, 2^k - 1\}$, we obviously obtain another possible message M_l, $l \in \{0, \dots, 2^k - 1\}$.

Therefore, we can write

$C_l = M_l \cdot G = (M_i + M_j) \cdot G = M_i \cdot G + M_j \cdot G = C_i + C_j$.

This property greatly simplifies the search for the value of d_{min} for any linear block code.

Minimum Hamming distance of a linear code

The Hamming weight of a binary vector is defined as the number of 1s in this vector.

We can show that the Hamming distance between two arbitrary codewords is equal to the Hamming weight of the sum of these codewords:

$$d_H(C_i, C_j) = w_H(C_i + C_j) = w_H(C_l),$$

where $w_H(\cdot)$ denotes the Hamming weight of "$\cdot$".

The sum of two codewords is another codeword

Minimum Hamming distance of a linear code

Thus finding the minimum distance d_{min} between codewords for a linear block code consists of scanning all possible codewords in search for the minimum Hamming weight:

$$d_{min} = Min\left(d_H(C_i, C_j)\right) = Min\big(w_H(C_l)\big),$$
$$l \in \left\{1, \dots, 2^k - 1\right\}.$$

The all-zero codeword C_0 must NOT be included in this search as C_0 is the sum of two identical codewords, and thus associated with the Hamming distance between two identical codewords.

Systematic block codes

A block code is said to be systematic if the info bits are explicitly included in each codeword.

In other words, an n-bit codeword is composed of the k info bits to which $(n-k)$ coded bits are appended:
$C = (c_0, c_1, c_2, \dots, c_{n-1}) = (m_0, m_1, \dots, m_{k-1}, c_k, \dots, c_{n-1})$.

In this case, the $(n-k)$ coded bits $c_k, \dots, c_{n-1}$ are often referred to as parity bits.

Systematic block codes

The generator matrix of a systematic code is a $k \times n$ matrix in the form $G = [I_k|P]$, where I is a $k \times k$ identity matrix and P is a $k \times (n-k)$ matrix.

Most linear codes used in practice are systematic.

Examples of systematic codes: Hamming codes, parity-check codes, LDPC codes, recursive and systematic convolutional codes, turbo codes, etc.

Decoding of error correcting codes

How to decode in an optimal fashion an error-correcting code?

We want to determine the decoding algorithm that maximises the probability to successfully recover the transmitted codeword.

This optimal decoding algorithm is known as <u>maximum-likelihood (ML) decoding</u>.

ML decoding over BSC (hard-decision decoding)

Let us start with ML decoding over BSC (also known as hard-decision decoding), by considering the following initial assumptions:

We have the choice between two different codewords $C_i = \left(c_{i,0}, c_{i,1}, c_{i,2}, \dots, c_{i,n-1}\right)$ and $C_j = \left(c_{j,0}, c_{j,1}, c_{j,2}, \dots, c_{j,n-1}\right)$, with $c_{i,l}, c_{j,l} \in \{0, 1\}$, $l \in \{0, 1, 2, \dots, n-1\}$.

The received word at the channel output is $R = (r_0, r_1, r_2, \dots, r_{n-1})$, with $r_l \in \{0, 1\}$, $l \in \{0, 1, 2, \dots, n-1\}$.

ML decoding over BSC (hard-decision decoding)

We start by considering the optimal decision rule.

Given the choice between the two codewords C_i and C_j, we must take our decision as follows:
If $\Pr\{R, C_i\} > \Pr\{R, C_j\}$, we select C_i;
If $\Pr\{R, C_i\} < \Pr\{R, C_j\}$, we select C_j.

$\Pr\{R, C_i\}$ is the probability that the received word is R and the transmitted codeword was C_i.

ML decoding over BSC (hard-decision decoding)

$\Pr\{R, C_j\}$ is the probability that the received word is R and the transmitted codeword was C_j.

Detection theory tells us that this the optimal decision rule, i.e. the decision rule that will maximise the probability to successfully recover the transmitted codeword.

To simplify the analysis, let us from now on focus on the inequality $\Pr\{R, C_i\} > \Pr\{R, C_j\}$.

ML decoding over BSC (hard-decision decoding)

Using Bayes' rule, the inequality

$$\Pr\{R, C_i\} > \Pr\{R, C_j\}$$

can be written as

$$\Pr\{R|C_i\} \cdot \Pr\{C_i\} > \Pr\left\{R \,\middle|\, C_j\right\} \cdot \Pr\{C_j\},$$

where $\Pr\{R|C_i\}$ and $\Pr\left\{R \,\middle|\, C_j\right\}$, denote the probabilities to receive R given the fact that C_i and C_j, respectively, were transmitted.

$\Pr\{C_i\}$ and $\Pr\{C_j\}$ denote the probabilities that C_i and C_j, respectively, were transmitted.

ML decoding over BSC (hard-decision decoding)

Since all codewords can be generated with equal probabilities, we can write

$$\Pr\{C_i\} = \Pr\{C_j\} = \frac{1}{2^k},$$

and the inequality

$$\Pr\{R|C_i\} \cdot \Pr\{C_i\} > \Pr\left\{R\,\middle|\,C_j\right\} \cdot \Pr\{C_j\}$$

thus becomes

$$\Pr\{R|C_i\} > \Pr\left\{R\,\middle|\,C_j\right\}.$$

Let us now focus on the term $\Pr\{R|C_i\}$.

ML decoding over BSC (hard-decision decoding)

It is easy to show that

$$\Pr\{R|C_i\} = \Pr\left\{\left(r_0 \left| c_{i,0}\right.\right) \cap \left(r_1 \left| c_{i,1}\right.\right) \cap \cdots \cap \left(r_{n-1} \left| c_{i,n-1}\right.\right)\right\},$$

where the event $\left(r_l \left| c_{i,l}\right.\right)$, $l \in \{0, 1, 2, \dots, n-1\}$, refers to the reception of a channel sample r_l given the transmission of a coded bit $c_{i,l}$.

Given the fact that all events $\left(r_l \left| c_{i,l}\right.\right)$, $l \in \{0, 1, 2, \dots, n - 1\}$, are independent, we can write

$$\Pr\{R|C_i\} = \Pr\left\{r_0 \left| c_{i,0}\right.\right\} \cdot \Pr\left\{r_1 \left| c_{i,1}\right.\right\} \dots \Pr\left\{r_{n-1} \left| c_{i,n-1}\right.\right\}$$

ML decoding over BSC (hard-decision decoding)

The terms $\Pr\left\{r_l \middle| c_{i,l}\right\}$, $l \in \{0, 1, 2, \ldots, n-1\}$, can be equal to $(1-p)$ or p, depending on whether r_l and $c_{i,l}$ are identical or not.

Recall that the parameter p denotes the bit error probability over the BSC.

Let d_i denote the Hamming distance between the binary words R and C_i.

ML decoding over BSC (hard-decision decoding)

In other words, these two words differ in d_i positions, which also means that they are equal in $(n - d_i)$ positions.

Therefore, it appears that

$$\Pr\{R|C_i\} = \Pr\{r_0 \mid c_{i,0}\} \cdot \Pr\{r_1 \mid c_{i,1}\} \ldots \Pr\{r_{n-1} \mid c_{i,n-1}\}$$

can be written as

$$\Pr\{R|C_i\} = p^{d_i}(1-p)^{n-d_i}.$$

ML decoding over BSC (hard-decision decoding)

The inequality

$$\Pr\{R \big| C_i\} > \Pr\left\{R \,\middle|\, C_j\right\}$$

can thus finally be written as

$$p^{d_i}(1-p)^{n-d_i} > p^{d_j}(1-p)^{n-d_j},$$

where d_j denote the Hamming distance between the binary words R and C_j.

ML decoding over BSC (hard-decision decoding)

We notice that $p^{d_i}(1-p)^{n-d_i} > p^{d_j}(1-p)^{n-d_j}$ is equivalent to

$$\frac{p^{d_i-d_j}}{(1-p)^{n-d_j-n+d_i}} > 1,$$

i.e.

$$\left(\frac{p}{1-p}\right)^{d_i-d_j} > 1.$$

The term $\frac{p}{1-p}$ is positive but less than the unit because we always have $0 < p < 0.5$ for a BSC.

ML decoding over BSC (hard-decision decoding)

This implies that the inequality

$$\left(\frac{p}{1-p}\right)^{d_i - d_j} > 1.$$

is strictly equivalent to $d_i - d_j < 0$, i.e. $d_i < d_j$.

We conclude that the ML decision rule over BSC is finally as follows:

If $d_i < d_j$, we select C_i;
If $d_i > d_j$, we select C_j.

By extending this rule to the whole set of possible codewords, we obtain the ML decoding procedure.

ML decoding over BSC (hard-decision decoding)

ML decoding over BSC simply consists of choosing, among all possible 2^k codewords, the one which is at minimum Hamming distance from the received word R.

Let us now turn our attention to the issue of maximum-likelihood decoding over a BPSK, AWGN channel.

ML decoding over BPSK, AWGN channel (soft-decision decoding)

Let us consider the following initial assumptions.

We have the choice between two different codewords $C_i = \left(c_{i,0}, c_{i,1}, c_{i,2}, \dots, c_{i,n-1}\right)$ and $C_j = \left(c_{j,0}, c_{j,1}, c_{j,2}, \dots, c_{j,n-1}\right)$, with $c_{i,l}, c_{j,l} \in \{-1, +1\}$, $l \in \{0, 1, 2, \dots, n-1\}$.

The received word at the channel output is $R = (r_0, r_1, r_2, \dots, r_{n-1})$. Each channel sample r_l, $l \in \{0, 1, 2, \dots, n-1\}$, has a Gaussian distribution.

ML decoding over BPSK, AWGN channel (soft-decision decoding)

We start by considering the optimal decision rule.

Given the choice between the two codewords C_i and C_j, we must take our decision as follows:
If $\Pr\{R, C_i\} > \Pr\{R, C_j\}$, we select C_i;
If $\Pr\{R, C_i\} < \Pr\{R, C_j\}$, we select C_j.

$\Pr\{R, C_i\}$ is the probability that the received word is R and the transmitted codeword was C_i.

ML decoding over BPSK, AWGN channel (soft-decision decoding)

$\Pr\{R, C_j\}$ is the probability that the received word is R and the transmitted codeword was C_j.

In fact, the start of the derivation for the BPSK, AWGN channel is identical to that already performed for the BSC.

The difference only begins once we reach the expression

$$\Pr\{R|C_i\} = \Pr\left\{r_0\left|c_{i,0}\right.\right\}\cdot\Pr\left\{r_1\left|c_{i,1}\right.\right\}\ldots\Pr\left\{r_{n-1}\left|c_{i,n-1}\right.\right\}.$$

ML decoding over BPSK, AWGN channel (soft-decision decoding)

A term $\Pr\left\{r_l \middle| c_{i,l}\right\}$, $l \in \{0, 1, 2, \dots, n-1\}$, can be derived by replacing it with the probability density function $P\left(r_l \middle| c_{i,l}\right)$ of a sample r_l, given the translission of a coded bit $c_{i,l}$:

$$\Pr\left\{r_l \middle| c_{i,l}\right\} = P\left(r_l \middle| c_{i,l}\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(r_l - c_{i,l}\right)^2}{2\sigma^2}\right).$$

We thus have

$$\Pr\{R|C_i\} = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n \prod_{l=0}^{n-1} \exp\left(-\frac{\left(r_l - c_{i,l}\right)^2}{2\sigma^2}\right)$$

ML decoding over BPSK, AWGN channel (soft-decision decoding)

This can be written as

$$\Pr\{R|C_i\} = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n \exp\left(-\frac{\sum_{l=0}^{n-1}\left(r_l - c_{i,l}\right)^2}{2\sigma^2}\right).$$

The inequality

$$\Pr\{R|C_i\} > \Pr\left\{R\,\middle|\,C_j\right\}$$

can be written as

$$\exp\left(-\frac{\sum_{l=0}^{n-1}\left(r_l - c_{i,l}\right)^2}{2\sigma^2}\right) > \exp\left(-\frac{\sum_{l=0}^{n-1}\left(r_l - c_{j,l}\right)^2}{2\sigma^2}\right).$$

ML decoding over BPSK, AWGN channel (soft-decision decoding)

This inequality is equivalent to

$$\sum_{l=0}^{n-1}\left(r_l - c_{i,l}\right)^2 < \sum_{l=0}^{n-1}\left(r_l - c_{j,l}\right)^2.$$

,

We recognise that the term $\sum_{l=0}^{n-1}\left(r_l - c_{i,l}\right)^2$, resp. $\sum_{l=0}^{n-1}\left(r_l - c_{j,l}\right)^2$, is actually the square of the Euclidean distance d_i, resp. d_j, between R and C_i, resp. C_j.

This is easy to understand once you remember your high school mathematics,

ML decoding over BPSK, AWGN channel (soft-decision decoding)

Consider two points A$\begin{pmatrix} a_0 \\ a_1 \end{pmatrix}$ and B $\begin{pmatrix} b_0 \\ b_1 \end{pmatrix}$ in the two-dimensional space.

The Euclidean distance d_{AB} between A and B is defined as

$$d_{AB} = \sqrt{(a_0 - b_0)^2 + (a_1 - b_1)^2}.$$

The square of d_{AB} can thus be written as

$$d_{AB}^2 = (a_0 - b_0)^2 + (a_1 - b_1)^2 = \sum_{l=0}^{1} (a_l - b_l)^2.$$

ML decoding over BPSK, AWGN channel (soft-decision decoding)

Let us extend this reasoning to an n-dimensional space, where A and B have n coordinates instead of two. It is then clear that the square of the Euclidean distance between A and B can be redefined as

$$d_{AB}^2 = \sum_{l=0}^{n-1} (a_l - b_l)^2.$$

By analogy, in coding theory, the term $\sum_{l=0}^{n-1} \left(r_l - c_{i,l}\right)^2$, resp. $\sum_{l=0}^{n-1} \left(r_l - c_{j,l}\right)^2$, can be seen as the square of the Euclidean distance d_i, resp. d_j, between R and C_i, resp. C_j.

ML decoding over BPSK, AWGN channel (soft-decision decoding)

The inequality $\sum_{l=0}^{n-1}\left(r_l - c_{i,l}\right)^2 < \sum_{l=0}^{n-1}\left(r_l - c_{j,l}\right)^2$ can thus be written as

$$d_i^2 < d_j^2 \text{ or, equivalently, } d_i < d_j \ .$$

We conclude that the ML decision rule over BPSK, AWGN channel is as follows:

If $d_i < d_j$, we select C_i;
If $d_i > d_j$, we select C_j.

By extending this rule to the whole set of possible codewords, we obtain the ML decoding procedure.

ML decoding over BPSK, AWGN channel (soft-decision decoding)

ML decoding over BPSK, AWGN channel simply consists of choosing, among all possible 2^k codewords, the one which is at minimum Euclidean distance from the received word R.

In practice, we do not have to compute any Euclidean distance to implement ML decoding.

ML decoding over BPSK, AWGN channel (soft-decision decoding)

The square of the Euclidean distance between a codeword $C_i = \left(c_{i,0}, c_{i,1}, c_{i,2}, \dots, c_{i,n-1}\right)$ and the received word $R = (r_0, r_1, r_2, \dots, r_{n-1})$ can be written as

$$d_i^2 = \sum_{l=0}^{n-1}\left(r_l - c_{i,l}\right)^2 = \sum_{l=0}^{n-1} r_l^2 + \sum_{l=0}^{n-1} c_{i,l}^2 - 2\sum_{l=0}^{n-1} r_l \cdot c_{i,l},$$

which is equivalent to

$$d_i^2 = \sum_{l=0}^{n-1} r_l^2 + n - 2\sum_{l=0}^{n-1} r_l \cdot c_{i,l}.$$

The first two terms in this expression are identical for all codewords, i.e. do not depend on the codeword under consideration.

ML decoding over BPSK, AWGN channel (soft-decision decoding)

The inequality $d_i^2 < d_j^2$ can therefore be written as

$\sum_{l=0}^{n-1} r_l^2 + n - 2\sum_{l=0}^{n-1} r_l \cdot c_{i,l} < \sum_{l=0}^{n-1} r_l^2 + n - 2\sum_{l=0}^{n-1} r_l \cdot c_{j,l}$,

which is equivalent to

$$\sum_{l=0}^{n-1} r_l \cdot c_{i,l} > \sum_{l=0}^{n-1} r_l \cdot c_{j,l}.$$

Therefore, finding the codeword at minimal distance from R is equivalent to finding the codeword for which the term $\gamma(R, C_i) = \sum_{l=0}^{n-1} r_l \cdot c_{i,l}$ is maximal.

The quantity $\gamma(R, C_i)$ actually measures the degree of correlation between the received word R and a codeword C_i.

ML decoding over BPSK, AWGN channel (soft-decision decoding)

This result thus means that finding the codeword at minimal distance from R is strictly equivalent to finding the codeword which is the most correlated with R.

To illustrate how ML decoding is implemented over a BPSK, AWGN channel, let us consider our (5, 2) code once again.

The four possible codewords are C_0 = (00000), C_1 = (01011), C_2 = (10101), and C_3 = (11110).

ML decoding over BPSK, AWGN channel (soft-decision decoding)

Since we assume transmission over a BPSK, AWGN channel, the codewords must be written using the (+1, -1) notation: C_0 = (-1 -1 -1 -1 -1), C_1 = (-1 +1 -1 +1 +1), C_2 = (+1 -1 +1 -1 +1), and C_3 = (+1 +1 +1 +1 -1).

Assume the codeword C_0 = (-1 -1 -1 -1 -1) is transmitted and the resulting channel output is R = (0.1 0.5 0.2 -1.5 -1.0). What is the decoded message?

Before moving on, we can notice that the channel "believes" that the first three coded bits were all equal to +1, whereas the last two bits were equal to -1.

ML decoding over BPSK, AWGN channel (soft-decision decoding)

The channel also indicates the level of confidence it has in these "beliefs".

For the 1st and 3rd coded bits, this level of confidence is clearly very low because both channel samples are very close to the border line between +1 and -1.

For the last two bits, the channel is much more confident about its belief as the last two channel samples are clearly in negative territory.

ML decoding over BPSK, AWGN channel (soft-decision decoding)

We could take a hard decision on R before decoding (by inserting a decision block between channel output and decoder input).

It would be a dumb thing to do because we would instantly lose the info regarding the reliability of the channel samples, i.e. the level of confidence that the channel has in its "beliefs".

Surely, the decoder would do a better job if being given all available info.

ML decoding over BPSK, AWGN channel (soft-decision decoding)

Anyway, if we still insisted to take a hard decision on R before decoding, we would feed the decoder with the binary vector R′ = (+1 +1 +1 -1 -1), i.e. R′ = (1 1 1 0 0).

Although the actual channel is a BPSK, AWGN channel, the decoder would "see" a BSC. It would thus be operating using hard decisions.

The computations of the Hamming distances between R and the four possible codewords would yield $d_H(R, C_0) = 3$, $d_H(R, C_1) = 4$, $d_H(R, C_2) = 2$, and $d_H(R, C_3) = 1$.

ML decoding over BPSK, AWGN channel (soft-decision decoding)

The decoder would select the codeword which is at minimum Hamming distance from R and thus decide that the codeword C_3 = (11110) was transmitted.

The decoded message would thus be M' = (11).

The decoding process had evidently failed as C_0 = (00000), associated with the message M_0 = (00), was actually transmitted.

ML decoding over BPSK, AWGN channel (soft-decision decoding)

Now, a better way consists of feeding our decoder directly with the channel samples R = (0.1 0.5 0.2 -1.5 -1.0). The decoder is then said to operate from soft decisions provided by the BPSK, AWGN channel.

The decoder computes the four correlation terms

$\gamma(R, C_i) = \sum_{l=0}^{n-1} r_l \cdot c_{i,l}$:

$\gamma(R, C_0) = -0.1 - 0.5 - 0.2 + 1.5 + 1.0 = +1.7,$

$\gamma(R, C_1) = -0.1 + 0.5 - 0.2 - 1.5 - 1.0 = -2.3,$

$\gamma(R, C_2) = +0.1 - 0.5 + 0.2 + 1.5 - 1.0 = +0.3,$

$\gamma(R, C_3) = +0.1 + 0.5 + 0.2 - 1.5 + 1.0 = +0.3.$

ML decoding over BPSK, AWGN channel (soft-decision decoding)

As a result, the decoder decides that the codeword C_0 = (00000) was transmitted and the decoded message is M' = (00).

The decoding has been successful as the decoder has been able to recover the right codeword and message.

Our advice: Never discard info. Provide your decision systems with as much of it as possible. They will thank you for that by doing a better job.

ML decoding procedure - Summary

a. Compute the distance (Hamming or Euclidean) between the received word and all codewords.

b. Choose the codeword corresponding to the minimum distance.
→ Can be impractical since there are 2^k distances to compute.

ML decoding provides optimal decoding performance, but is inherently too complex to implement for practical codes for which k is often large (long blocks of info bits).

ML decoding procedure - Summary

When ML decoding is not practical, we must use other (sub-optimal) decoding algorithms instead.

If the code possesses some structure, it is often possible to employ a decoding algorithm that displays near-ML performance, or even ML performance in some cases.

The best example of this are codes whose operation can be described using a trellis diagram (mainly convolutional codes).

ML decoding procedure - Summary

With these codes, we can exploit the trellis structure in order to devise an ML decoding algorithm which can actually be implemented in practice.

This well-known algorithm is called “Viterbi algorithm” (invented by A Viterbi in 1967), and has been used for decades in many applications beyond the field of channel coding.

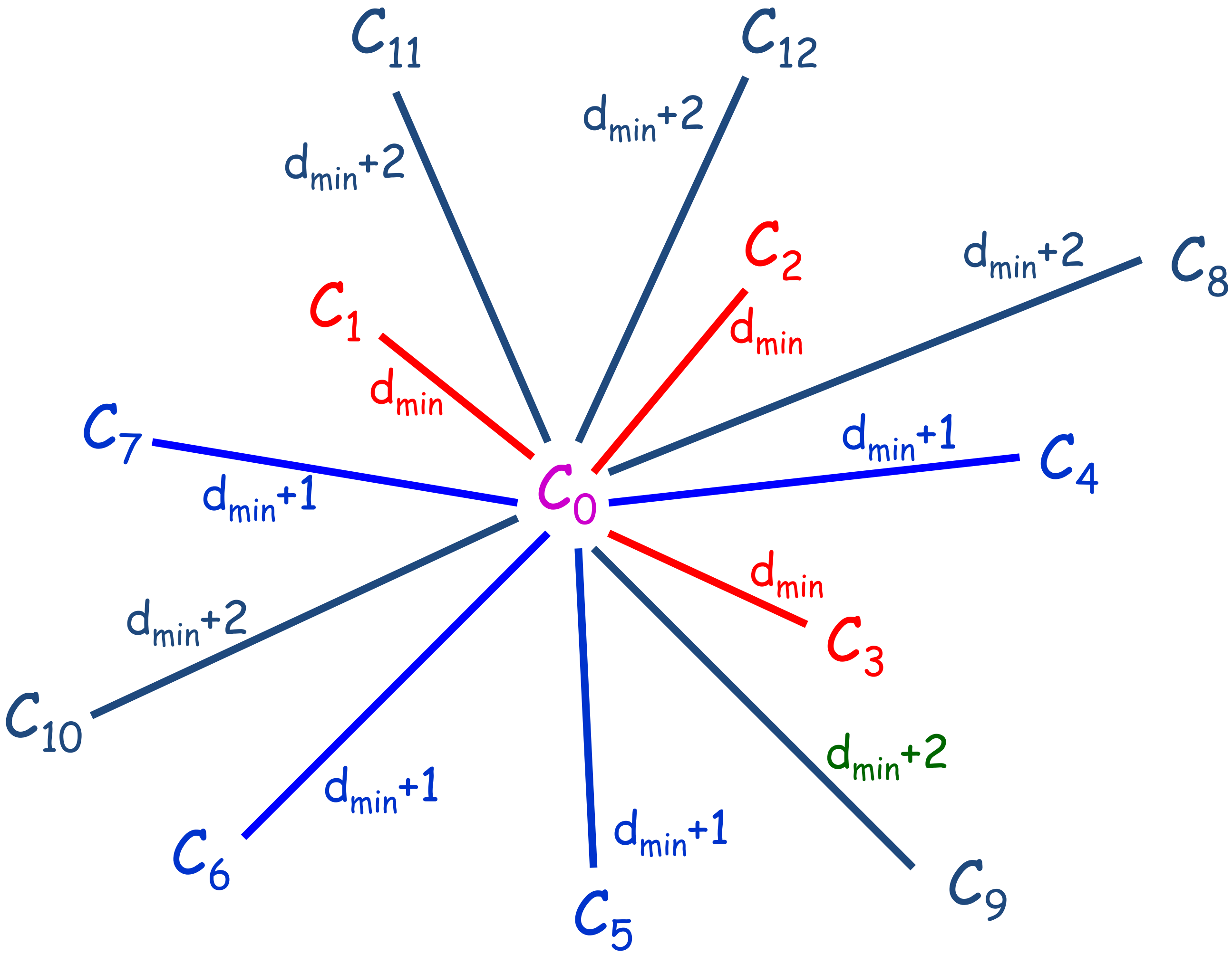
The Viterbi algorithm can operate both in hard and soft-decision modes. We will study it later.

ML decoding procedure - Summary

Some practical codes possess very little structure (e.g., turbo codes, LDPC codes). They are considered as pseudo-random codes. For these codes, we can employ an iterative decoding algorithm to achieve a decoding performance very close to that of the ML procedure.

Long codes (for which $k, n \rightarrow +\infty$) with very little structure are particularly important. These pseudo-random codes are precisely the types of codes that could be used to reach the channel capacity limit, as shown by Shannon in 1948.

Pictorial view of a code



Pictorial view of a code

There are
3 codewords at minimal distance d_{min} from C_0;
4 codewords at $(d_{min}+1)$ from C_0;
5 codewords at $(d_{min}+2)$ from C_0.

C_1, C_2 and C_3 are nearest neighbours of C_0.

We need to understand this pictorial view to determine the expression of the bit error probability at the decoder output, for a given block code.

Bit error probability of a coded system

Assume, without loss of generality, that the all-zero codeword C_0 is transmitted.

Bit error probability after decoding: $P_{eb} = \Pr\{m' \neq m\}$, where m is an info bit and m' is the corresponding decoded bit.

We can also write $P_{eb} = \Pr\left\{(m' \neq m \cap C_0 \rightarrow C_1) \cup \cdots \cup \left(m' \neq m \cap C_0 \rightarrow C_{2^k-1}\right)\right\}$, where "$C_0 \rightarrow C_i$", $i \in \{1, 2, \ldots, 2^k - 1\}$, refers to the event where the decoder selects the codeword C_i rather than the transmitted codeword C_0.

Bit error probability of a coded system

It is impossible to find an expression of this probability as the $(2^k - 1)$ events "$m' \neq m \cap C_0 \to C_i$", $i \in \{1, 2, \dots, 2^k - 1\}$, are NOT mutually exclusive.

We can only determine an upper bound for P_{eb} by using Boole's inequality also known as the <u>union bound</u>:

$$P_{eb} \leq \sum_{i=1}^{2^k-1} \Pr\{(m' \neq m \cap C_0 \to C_i)\}.$$

By using Bayes' rule, we obtain

$$P_{eb} \leq \sum_{i=1}^{2^k-1} \Pr\{m' \neq m | C_0 \to C_i\} \Pr\{C_0 \to C_i\}$$

Only depends on the Hamming distance between C_0 and C_i

Bit error probability of a coded system

Grouping together terms associated with codewords being at the same distance d from C_0, we obtain an alternative and more useful way of representing this sum:

$$\sum_{i=1}^{2^k-1} \Pr\{m' \neq m \middle| C_0 \to C_i\}.\Pr\{C_0 \to C_i\} = \sum_{d=d_{min}}^{n} \frac{w_d}{k}.P_d.$$

P_d: Probability of decoding a codeword which is at distance d from C_0;

w_d: Total Hamming weight of the messages associated with the codewords at distance d from C_0.

Bit error probability of a coded system

Let us consider an example to better understand this equation.

Consider our (5, 2) linear block code.

$C_0 = (00000)$
$C_1 = (01011)$
$C_2 = (10101)$
$C_3 = (11110)$

M		C				
m_0	m_1	c_0	c_1	c_2	c_3	c_4
0	0	0	0	0	0	0
0	1	0	1	0	1	1
1	0	1	0	1	0	1
1	1	1	1	1	1	0

Bit error probability of a coded system

For the (5, 2) code we can write

$$\sum_{i=1}^{2^k-1} \Pr\{m' \neq m | C_0 \to C_i\}. \Pr\{C_0 \to C_i\}$$
$$= \Pr\{m' \neq m | C_0 \to C_1\} \cdot \Pr\{C_0 \to C_1\}$$
$$+ \Pr\{m' \neq m | C_0 \to C_2\} \cdot \Pr\{C_0 \to C_2\}$$
$$+ \Pr\{m' \neq m | C_0 \to C_3\} \cdot \Pr\{C_0 \to C_3\}$$
$$= \left[\Pr\{m' \neq m | C_0 \to C_1\} + \Pr\{m' \neq m | C_0 \to C_2\}\right] \cdot P_{d=3}$$
$$+ \Pr\{m' \neq m | C_0 \to C_3\} \cdot P_{d=4}$$

with $P_{d=3} = \Pr\{C_0 \to C_1\} = \Pr\{C_0 \to C_2\}$ and $P_{d=4} = \Pr\{C_0 \to C_3\}$.

Bit error probability of a coded system

This expression can be written as

$$\sum_{i=1}^{3} \Pr\left\{m' \neq m \middle| C_0 \to C_i\right\}. \Pr\{C_0 \to C_i\}$$

$$= \left[\Pr\left\{m' \neq m \middle| C_0 \to C_1\right\} + \Pr\left\{m' \neq m \middle| C_0 \to C_2\right\}\right] \cdot P_{d=3}$$

$$+ \Pr\left\{m' \neq m \middle| C_0 \to C_3\right\} \cdot P_{d=4} = \left[\frac{1}{2} + \frac{1}{2}\right] \cdot P_{d=3} + \frac{2}{2} \cdot P_{d=4}$$

$$= \left[\frac{1+1}{2}\right] \cdot P_{d=3} + \frac{2}{2} \cdot P_{d=4} = \sum_{d=3}^{4} \frac{w_d}{k} . P_d$$

Bit error probability of a coded system

The expression of the union bound is thus given by

$$P_{eb} \leq \sum_{d=d_{min}}^{n} \frac{w_d}{k} .P_d .$$

From now on, we are only going to focus on the BPSK, AWGN channel, and forget about the BSC.

The BPSK, AWGN channel is the standard channel in info theory and digital communications.

We need to find an expression for the term P_d that can be defined as $P_d = \Pr\{C_0 \rightarrow C_i | d_H(C_0, C_i) = d\}$.

Bit error probability of a coded system

Assuming maximum-likelihood decoding, we have
$P_d = \Pr\{C_0 \to C_i | d_H(C_0, C_i) = d\} = \Pr\left\{\sum_{l=0}^{n-1}(r_l - c_{i,l})^2 < \sum_{l=0}^{n-1}(r_l - c_{0,l})^2 \ | d_H(C_0, C_i) = d\right\}.$
with $C_i = (c_{i,0}, c_{i,1}, c_{i,2}, \ldots, c_{i,n-1})$, and $C_0 = (c_{0,0}, c_{0,1}, c_{0,2}, \ldots, c_{0,n-1}) = (-1, -1, -1, \ldots, -1)$.

The vector $R = (r_0, r_1, r_2, \ldots, r_{n-1})$ denotes the vector of channel samples.

Since the all-zero codeword was transmitted, we have $r_l = c_{0,l} + n_l = -1 + n_l$, $l \in \{0, 1, 2, \ldots, n-1\}$.

Bit error probability of a coded system

We can write $\sum_{l=0}^{n-1}\left(r_l - c_{i,l}\right)^2 = \sum_{l=0}^{n-1} r_l^2 + \sum_{l=0}^{n-1} c_{i,l}^2 - 2\sum_{l=0}^{n-1} r_l \cdot c_{i,l}$, which is equivalent to

$$\sum_{l=0}^{n-1}\left(r_l - c_{i,l}\right)^2 = \sum_{l=0}^{n-1} r_l^2 + n - 2\sum_{l=0}^{n-1} r_l \cdot c_{i,l}.$$

In the same way, we can show that

$$\sum_{l=0}^{n-1}\left(r_l - c_{0,l}\right)^2 = \sum_{l=0}^{n-1} r_l^2 + n - 2\sum_{l=0}^{n-1} r_l \cdot c_{0,l}.$$

The inequality "$\sum_{l=0}^{n-1}\left(r_l - c_{i,l}\right)^2 < \sum_{l=0}^{n-1}\left(r_l - c_{0,l}\right)^2$" can be written as $\sum_{l=0}^{n-1} r_l \cdot c_{i,l} > \sum_{l=0}^{n-1} r_l \cdot c_{0,l}$, which is equivalent to $\sum_{l=0}^{n-1} r_l\left(c_{i,l} - c_{0,l}\right) > 0$.

Bit error probability of a coded system

We can now make use of the fact that $d_H(C_0, C_i) = d$ in order to proceed further.

This equation implies that, in codewords C_0 and C_i , there are d positions for which $c_{0,l} = -c_{i,l}$ and (n-d) positions for which $c_{0,l} = c_{i,l}$.

Therefore, the inequality "$\sum_{l=0}^{n-1} r_l \left(c_{i,l} - c_{0,l}\right) > 0$" can be written as $\sum_{l=1}^{d} r_l > 0$, which is equivalent to $\sum_{l=0}^{n-1} (-1 + n_l) > 0$ since $r_l = c_{0,l} + n_l = -1 + n_l$.

Bit error probability of a coded system

The inequality "$\sum_{l=0}^{n-1} (-1+n_l) > 0$" can be written as $\sum_{l=1}^{d} n_l > d$, which finally leads to the following expression of the term P_d:

$$P_d = \Pr\left\{\sum_{l=1}^{d} n_l > d\right\}.$$

The quantity $\sum_{l=1}^{d} n_l$ is the sum of d independent Gaussian noise samples. As a result, $\sum_{l=1}^{d} n_l$ also has a Gaussian distribution with:

- a mean m_d equal to the sum of the means of the d samples n_l.
- and a variance σ_d^2 equal to the sum of the variances of the d samples n_l.

Bit error probability of a coded system

The quantity $\sum_{l=1}^{d} n_l$ has a Gaussian distribution.

Its mean is given by $m_d = E\left\{\sum_{l=1}^{d} n_l\right\} = \sum_{l=1}^{d} E\{n_l\} = 0.$

Its variance is given by

$$\sigma_d^2 = \sum_{l=1}^{d} \sigma^2 = d\sigma^2 = \frac{d}{2\left(\frac{E_S}{N_0}\right)}$$

where σ^2 is the variance of a noise sample n_l and $\frac{E_S}{N_0}$ denotes the SNR per coded bit.

Bit error probability of a coded system

It is preferable to express σ_d^2 as a function of the SNR per info bit:

$$\sigma_d^2 = \frac{d}{2\left(\frac{E_s}{N_0}\right)} = \frac{d}{2R_c\left(\frac{E_b}{N_0}\right)}.$$

Therefore, using the knowledge of the probability density function for a Gaussian random variable, we obtain

$$P_d = \Pr\left\{\sum_{l=1}^{d} n_l > d\right\} = \int_d^{+\infty} \frac{1}{\sqrt{2\pi\sigma_d^2}} exp\left(-\frac{(x-m_d)^2}{2\sigma_d^2}\right) dx$$

$$P_d = \frac{1}{\sqrt{2\pi\sigma_d^2}} \int_d^{+\infty} exp\left(-\frac{x^2}{2\sigma_d^2}\right) dx.$$

Bit error probability of a coded system

We can perform the following change of variable to compute this integral more easily:

$$u = \frac{x}{\sqrt{2\sigma_d^2}} \text{ and thus } dx = du\sqrt{2\sigma_d^2}.$$

We then obtain

$$P_d = \frac{\sqrt{2\sigma_d^2}}{\sqrt{2\pi\sigma_d^2}} \int_{\frac{d}{\sqrt{2\sigma_d^2}}}^{+\infty} e^{-u^2} du = \frac{1}{\sqrt{\pi}} \int_{\frac{d}{\sqrt{2\sigma_d^2}}}^{+\infty} e^{-u^2} du.$$

This integral does not have a closed-form expression. So, to proceed further, we now need to introduce the well-known complementary error function erfc(x).

Bit error probability of a coded system

The complementary error function is defined as

$$\operatorname{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^{+\infty} e^{-u^2} du.$$

This function can be computed using tables, matlab, excel, etc.

Finally, we obtain $P_d = \frac{1}{2}\operatorname{erfc}\left(\frac{d}{\sqrt{2\sigma_d^2}}\right) = \frac{1}{2}\operatorname{erfc}\left(\sqrt{\frac{d^2}{2\sigma_d^2}}\right) =$

$$\frac{1}{2}\operatorname{erfc}\left(\sqrt{\frac{d^2}{2\frac{d}{2R_c\left(\frac{E_b}{N_0}\right)}}}\right) = \frac{1}{2}\operatorname{erfc}\left(\sqrt{dR_c\frac{E_b}{N_0}}\right).$$

Bit error probability of a coded system

We finally obtain the union bound expression:

$$P_{eb} \leq \sum_{d=d_{min}}^{+\infty} \frac{w_d}{2k} \cdot \text{erfc}\left(\sqrt{dR_c \frac{E_b}{N_0}}\right).$$

The terms $e(d) = \frac{W_d}{2k}$ are called the error coefficients.

At sufficiently high SNR ($\frac{E_b}{N_0} \rightarrow +\infty$), the dominant term in this sum is that corresponding to the smallest value of d for the code, i.e. the minimal Hamming distance d_{min}.

Bit error probability of a coded system

$$P_{eb} \leq e(d_{min}) \cdot \operatorname{erfc}\left(\sqrt{d_{min} R_c \frac{E_b}{N_0}}\right)$$

$$+ \; e(d_{min}+1) \cdot \operatorname{erfc}\left(\sqrt{(d_{min}+1) R_c \frac{E_b}{N_0}}\right)$$

$$+ \; e(d_{min}+2) \cdot \operatorname{erfc}\left(\sqrt{(d_{min}+2) R_c \frac{E_b}{N_0}}\right)$$

$$+ \; e(d_{min}+3) \cdot \operatorname{erfc}\left(\sqrt{(d_{min}+3) R_c \frac{E_b}{N_0}}\right) + \cdots$$

Bit error probability of a coded system

As $x \to +\infty$, the negative slope of the $\mathrm{erfc}(x)$ function becomes more and more pronounced.

At some point, we can write $\mathrm{erfc}(x) \gg \mathrm{erfc}(x + \Delta x)$, where $\Delta x > 0$ represents an arbitrarily small positive increase in x.

Thus, as $\frac{E_b}{N_0} \to +\infty$, we have

$$P_{eb} \leq e(d_{min}) \cdot \mathrm{erfc}\left(\sqrt{d_{min} R_c \frac{E_b}{N_0}}\right).$$

Bit error probability of a coded system

This upper bound on P_{eb} is actually so tight in practice (to be checked) that we can even write

$$P_{eb} \approx e(d_{min}) \cdot \mathrm{erfc}\left(\sqrt{d_{min} R_c \frac{E_b}{N_0}}\right) \text{ as } \frac{E_b}{N_0} \to +\infty.$$

At SNRs of practical interest (corresponding to $P_{eb} \sim 10^{-4} - 10^{-7}$), this simplification is not always possible, and the next few terms associated with $d_{min} + 1$, $d_{min} + 2$, $d_{min} + 3$, etc, may also have to be considered.

For now, however, we will assume that the first term in the union bound provides a tight upper bound on P_{eb}.

Bit error probability of a coded system

Coded system over BPSK, AWGN channel (high SNR):

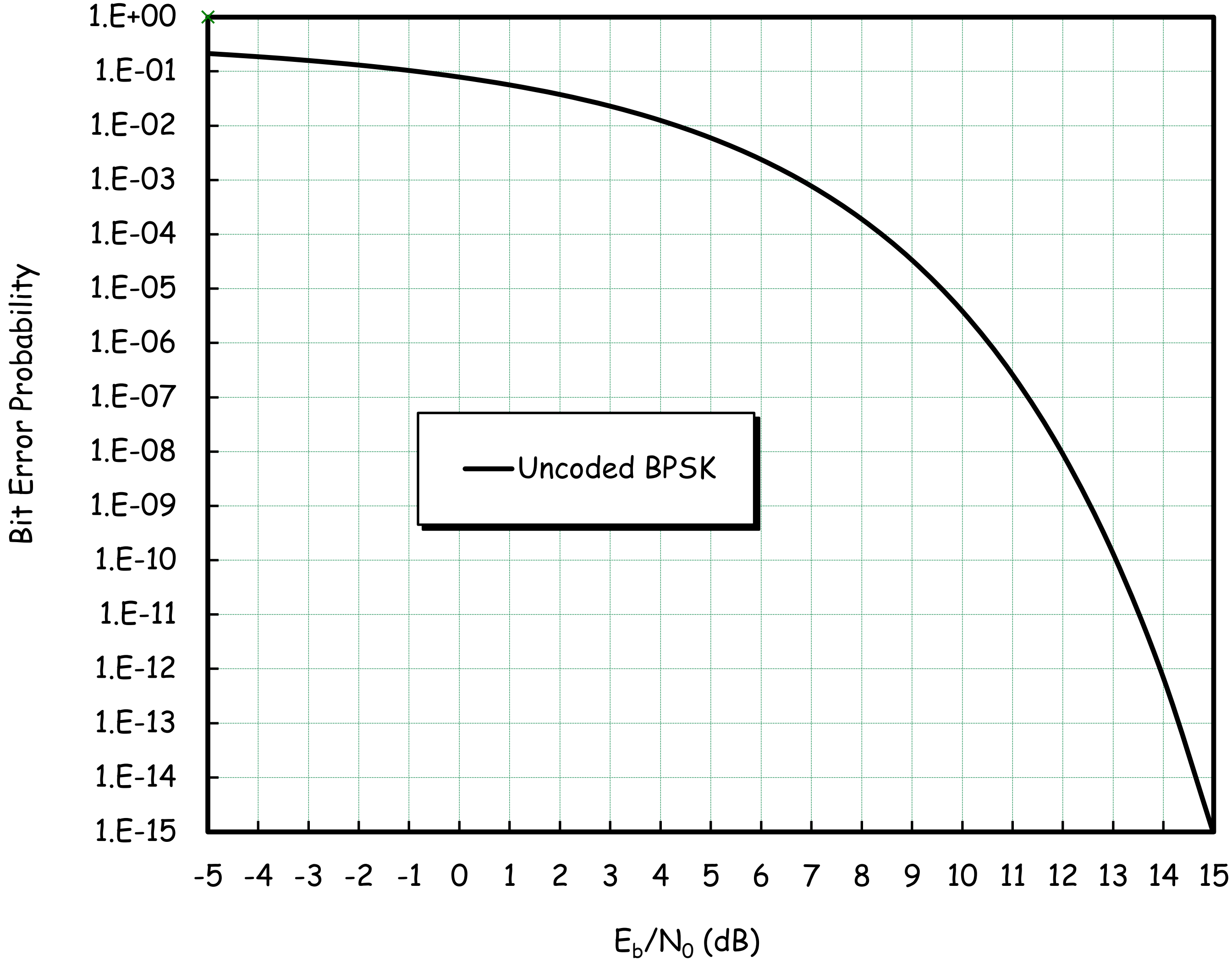
$$P_{eb} \approx e(d_{min}) \cdot \mathrm{erfc}\left(\sqrt{d_{min} R_c \frac{E_b}{N_0}}\right)$$

Uncoded BPSK: $P_{eb} = \frac{1}{2} \cdot \mathrm{erfc}\left(\sqrt{\frac{E_b}{N_0}}\right)$

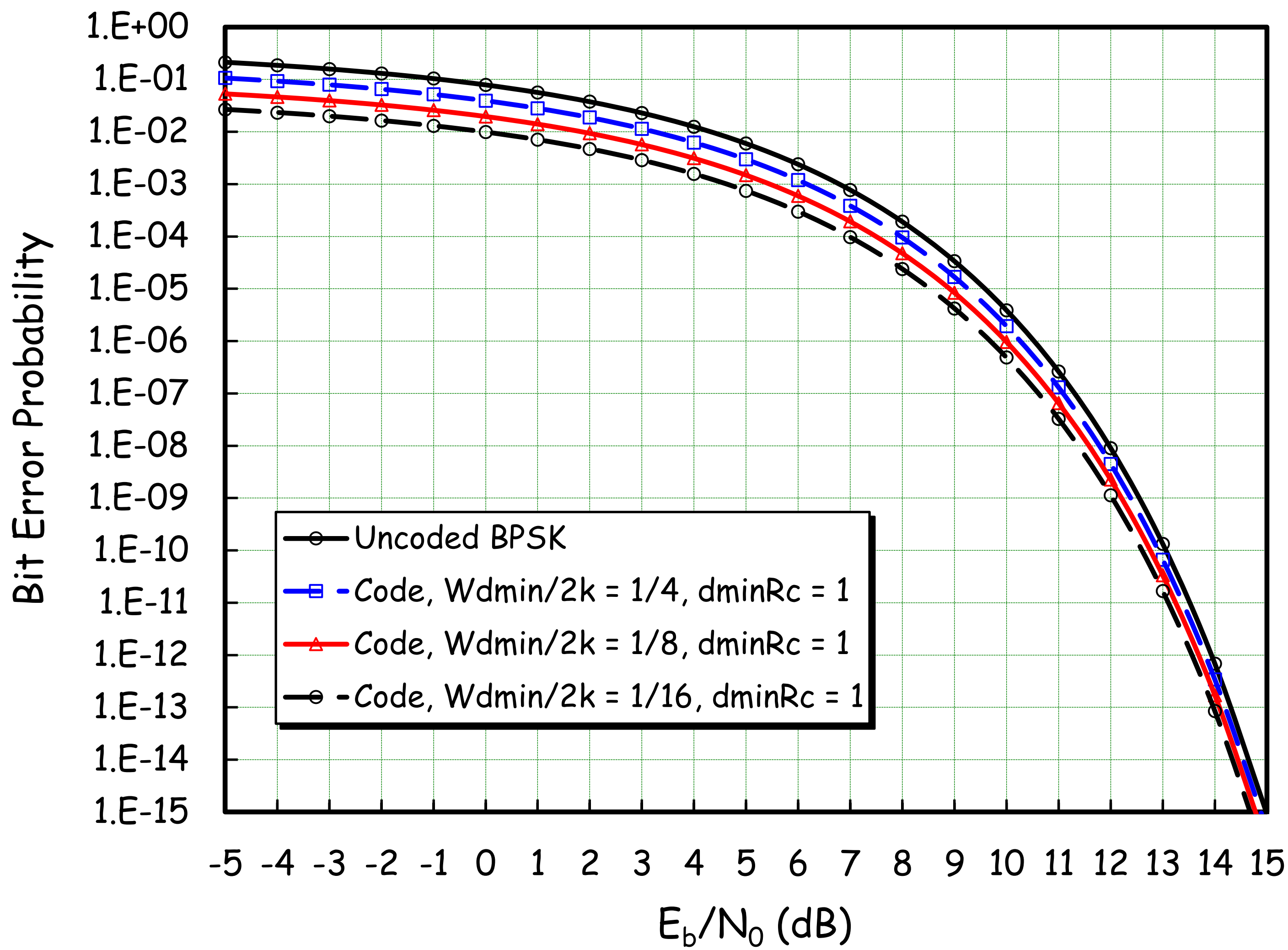
At high SNR, the bit error probability P_{eb} is mainly dependent on the argument inside the erfc(.) function.

At high SNR, the error coefficients $e(d_{min})$ and $\frac{1}{2}$ have a negligible effect on P_{eb}.

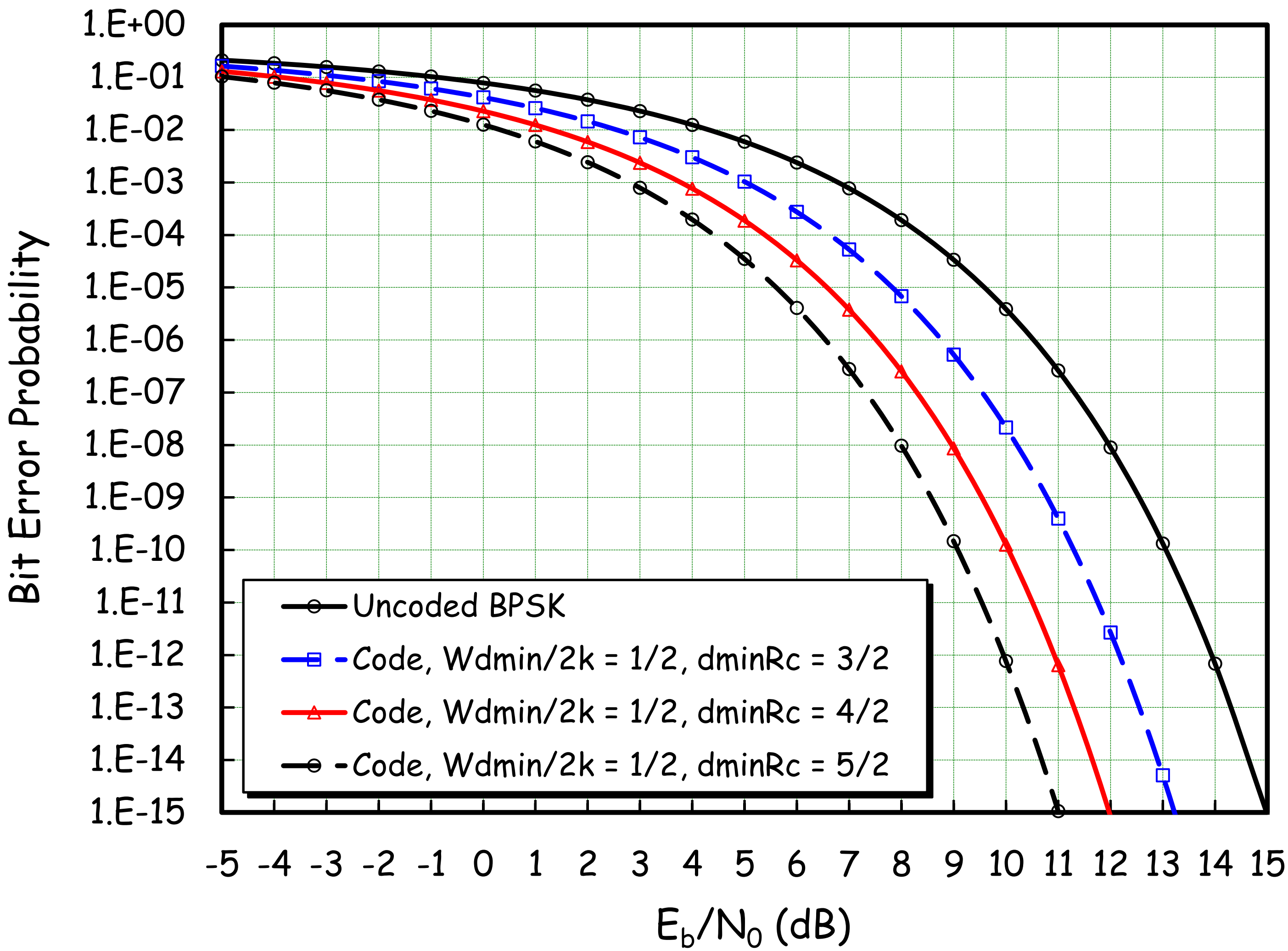
Error performance of uncoded BPSK over AWGN channel



The effect of the error coefficient(s) on the error performance can be considered as negligible at high SNRs.



The effect of the $d_{min}R_c$ product on the error performance is very significant at high SNRs



Bit error probability of a coded system

Comparison between coded and uncoded systems at high SNR.

If we want to achieve the same error probability with the coded and uncoded systems, we must have

$$P_{eb}(uncoded\ system) = P_{eb}(coded\ system),$$

which yields $\left(\frac{E_b}{N_0}\right)_{uncoded} \approx d_{min} R_c \left(\frac{E_b}{N_0}\right)_{coded}$.

By expressing both SNRs in decibels (dB), we obtain

$$\left(\frac{E_b}{N_0}\right)_{uncoded} - \left(\frac{E_b}{N_0}\right)_{coded} \approx 10 \cdot \log_{10}(d_{min} R_c) \text{ dB}.$$

Bit error probability of a coded system

At high SNR, both P_{eb} curves are actually "parallel" since the gap (called asymptotic coding gain) between them becomes constant as $\frac{E_b}{N_0} \to +\infty$.

The coded system provides the same P_{eb} as uncoded BPSK with a SNR which is $10 \cdot \log_{10}(d_{min} R_c)$ dB smaller.

The coding gain at high SNR (asymptotic coding gain) is thus given by $G \sim 10 \cdot \log_{10}(d_{min} R_c)$ dB.

Example 1: $(n = 7, k = 4)$ Hamming code $(d_{min} = 3)$

M				C						
0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	1	0	1
0	0	1	0	0	0	1	0	1	1	1
0	0	1	1	0	0	1	1	0	1	0
0	1	0	0	0	1	0	0	0	1	1
0	1	0	1	0	1	0	1	1	1	0
0	1	1	0	0	1	1	0	1	0	0
0	1	1	1	0	1	1	1	0	0	1

M				C						
1	0	0	0	1	0	0	0	1	1	0
1	0	0	1	1	0	0	1	0	1	1
1	0	1	0	1	0	1	0	0	0	1
1	0	1	1	1	0	1	1	1	0	0
1	1	0	0	1	1	0	0	1	0	1
1	1	0	1	1	1	0	1	0	0	0
1	1	1	0	1	1	1	0	0	1	0
1	1	1	1	1	1	1	1	1	1	1

7 codewords at distance $d_{min} = 3$ from C_0 → $w_{d_{min}} = 12$

Example 1: (n = 7, k = 4) Hamming code (d_{min} = 3)

$$P_{eb} \approx \frac{12}{2\times 4}\operatorname{erfc}\left[\sqrt{\frac{12}{7}\frac{E_b}{N_0}}\right] = \frac{3}{2}\operatorname{erfc}\left[\sqrt{\frac{12}{7}\frac{E_b}{N_0}}\right]$$

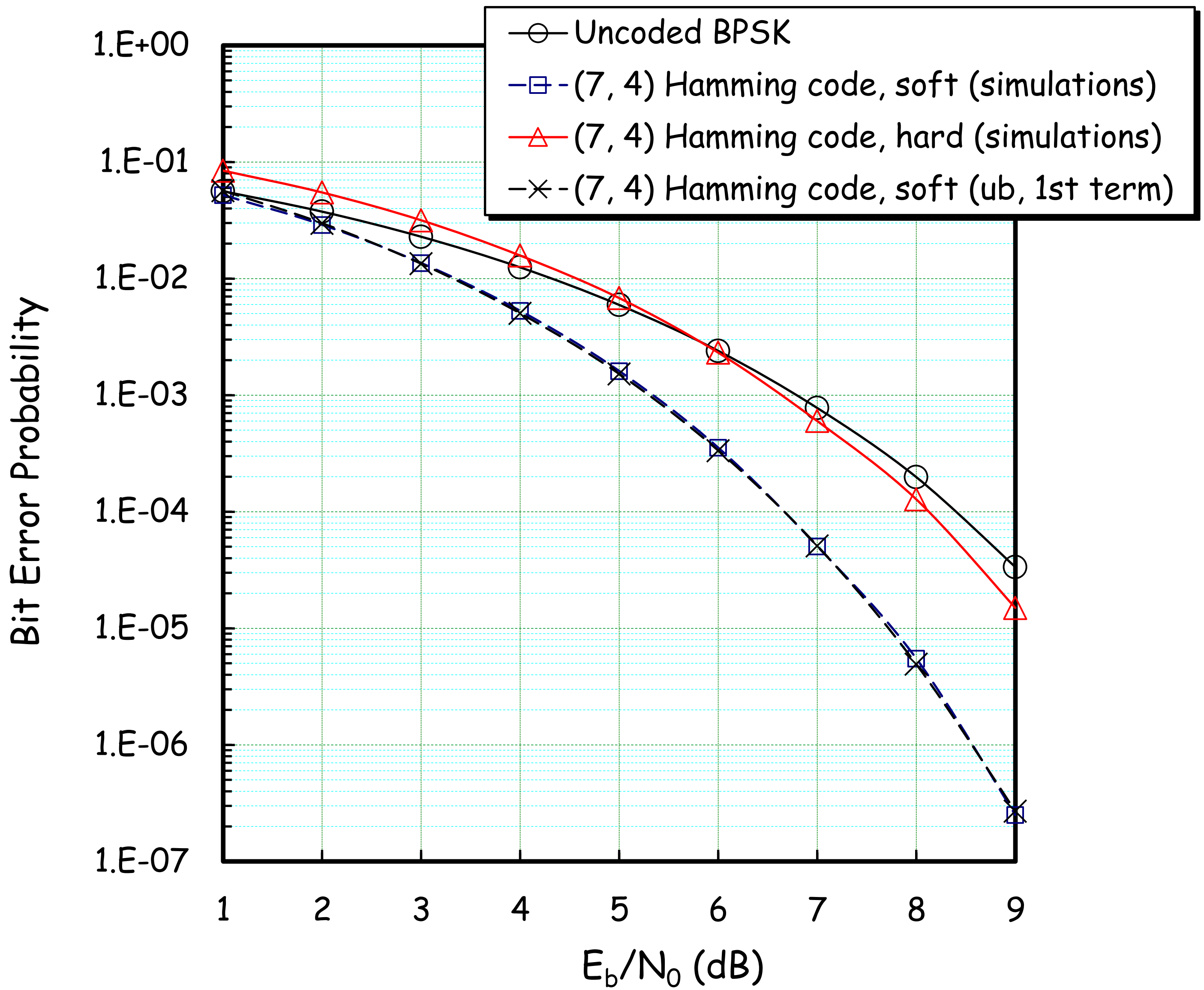
Asymptotic coding gain G = 2.34 dB.

On the following plots, the error performance (over AWGN channel) of the coded system is compared with that of uncoded BPSK.

Example 1: ($n = 7$, $k = 4$) Hamming code ($d_{min} = 3$)

The results shown for the coded system are obtained via computer simulations and also by using the theoretical equation provided by the 1st term of the union bound only.

In addition, we display the simulation results achieved when the decoder is fed with hard decisions rather than soft decisions. To generate hard decisions, we insert a decision block between the BPSK, AWGN channel output and the decoder input in order to convert the channel samples $r = s + n$ into bits prior to decoding.



Uncoded BPSK
(7, 4) Hamming code, soft (simulations)
(7, 4) Hamming code, hard (simulations)
(7, 4) Hamming code, soft (ub, 1st term)
Bit Error Probability
1.E+00
1.E-01
1.E-02
1.E-03
1.E-04
1.E-05
1.E-06
1.E-07
1
2
3
4
5
6
7
8
9
E_b/N_0 (dB)

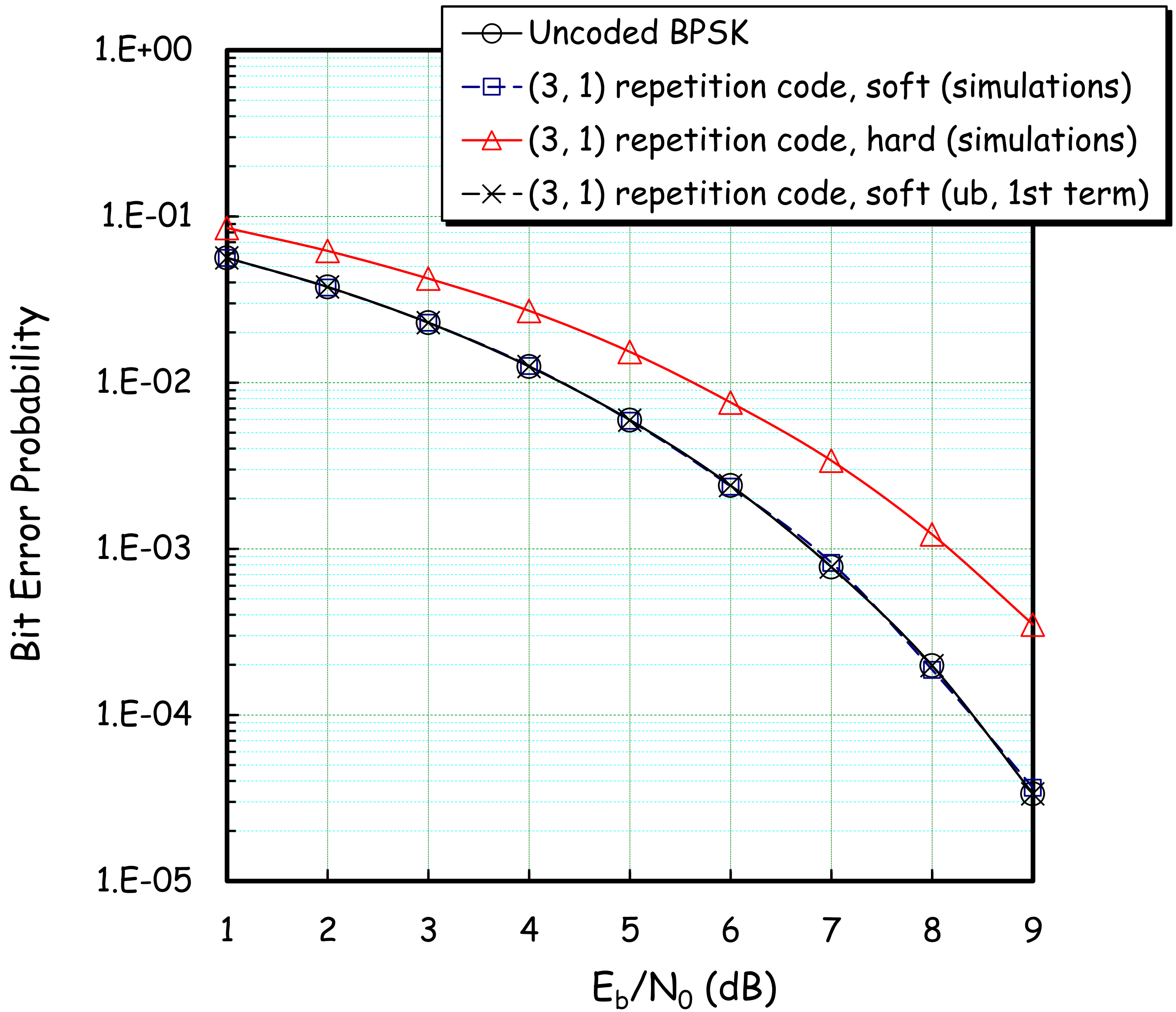
Example 2: (n = 3, k = 1) repetition code (d_{min} = 3)

M	C
0	000
1	111

1 codeword at distance $d_{min} = 3$ from C_0 → $w_{dmin} = 1$

$$P_{eb} \approx \frac{1}{2}\,\mathrm{erfc}\left[\sqrt{\frac{E_b}{N_0}}\right]$$

No coding gain over BPSK!


Uncoded BPSK
(3, 1) repetition code, soft (simulations)
(3, 1) repetition code, hard (simulations)
(3, 1) repetition code, soft (ub, 1st term)
1.E+00
1.E-01
1.E-02
1.E-03
1.E-04
1.E-05
Bit Error Probability
1
2
3
4
5
6
7
8
9
E_b/N_0 (dB)

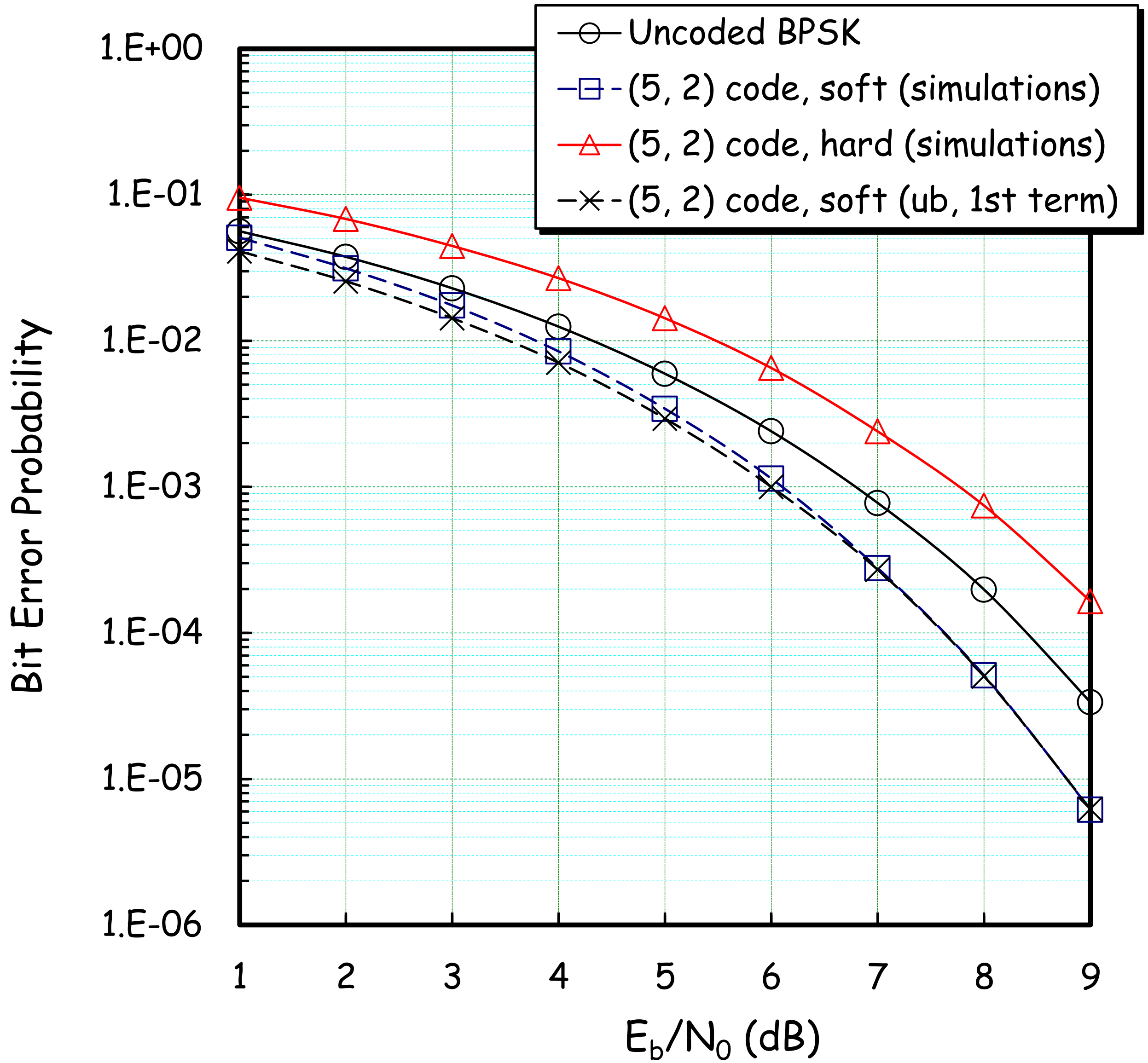
Example 3: Our (n = 5, k = 2) code ($d_{min} = 3$)

M	C
00	00000
01	01011
10	10101
11	11110

2 codewords at distance $d_{min} = 3$ from C_0
→ $w_{dmin} = 2$

$$P_{eb} \approx \frac{1}{2}\operatorname{erfc}\left[\sqrt{\frac{6}{5}\frac{E_b}{N_0}}\right]$$

Asymptotic coding gain G = 0.79 dB.



Uncoded BPSK
(5, 2) code, soft (simulations)
(5, 2) code, hard (simulations)
(5, 2) code, soft (ub, 1st term)
1.E+00
1.E-01
1.E-02
1.E-03
1.E-04
1.E-05
1.E-06
Bit Error Probability
1
2
3
4
5
6
7
8
9
E_b/N_0 (dB)

Example 4: ($n = 3$, $k = 2$) parity-check code ($d_{min} = 2$)

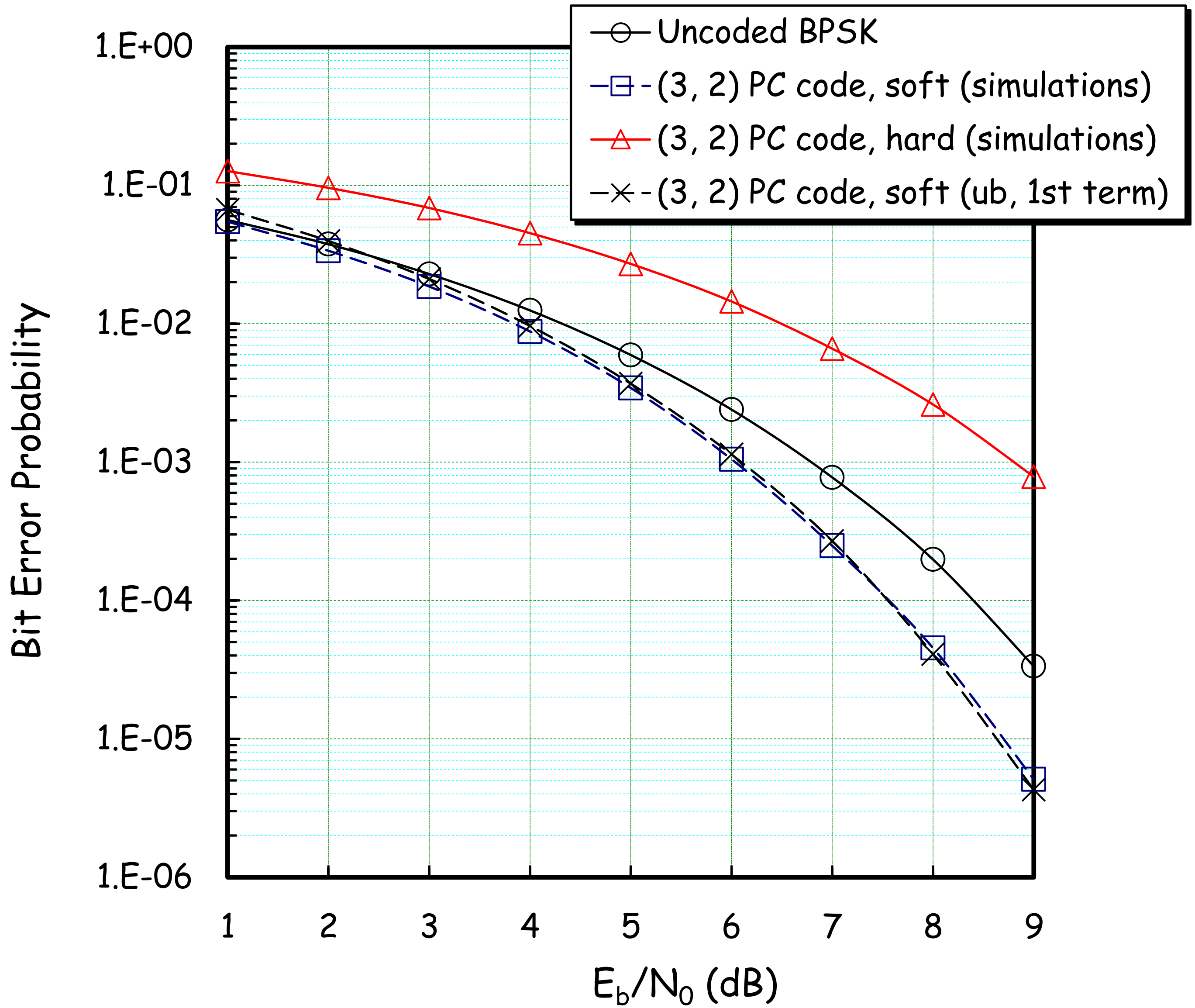
M	C
00	000
01	011
10	101
11	110

2 codewords at distance $d_{min} = 2$ from C_0
→ $w_{dmin} = 4$

$$\boxed{P_{eb} \approx \operatorname{erfc}\left[\sqrt{\frac{4}{3}\frac{E_b}{N_0}}\right]}$$

Asymptotic coding gain $G = 1.25$ dB.

Parity-check codes can correct transmission errors over BPSK, AWGN channels, which was not the conclusion we reached earlier when considering transmission over BSC.



Uncoded BPSK
(3, 2) PC code, soft (simulations)
(3, 2) PC code, hard (simulations)
(3, 2) PC code, soft (ub, 1st term)
1.E+00
1.E-01
1.E-02
1.E-03
1.E-04
1.E-05
1.E-06
Bit Error Probability
1
2
3
4
5
6
7
8
9
E_b/N_0 (dB)

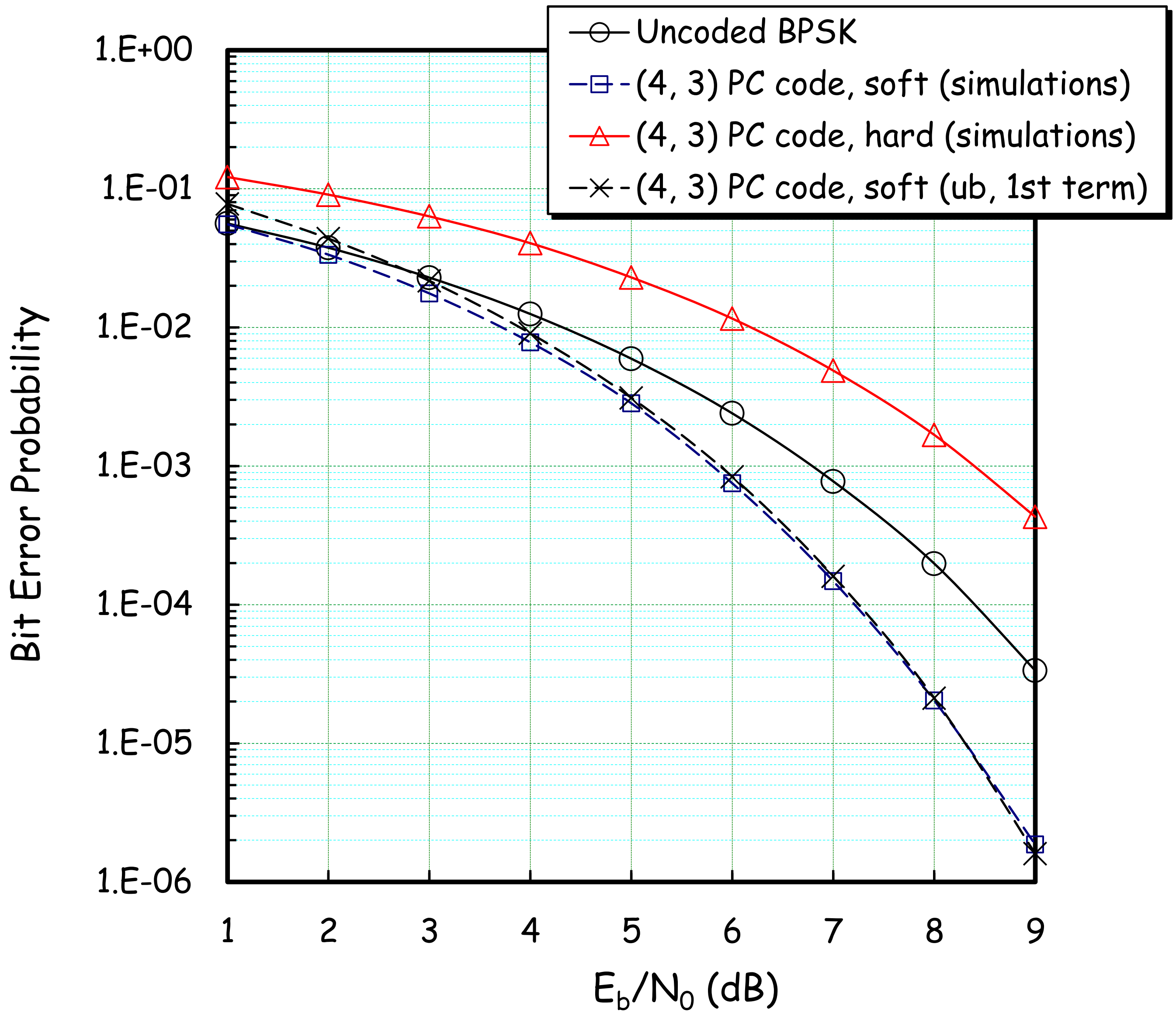
Example 5: ($n = 4$, $k = 3$) parity-check code ($d_{min} = 2$)

M	C
000	0000
001	0011
010	0101
011	0110
100	1001
101	1010
110	1100
111	1111

6 codewords at distance $d_{min} = 2$ from C_0
→ $w_{dmin} = 9$

$$\boxed{P_{eb} \approx \frac{3}{2}\,\mathrm{erfc}\left[\sqrt{\frac{3}{2}\frac{E_b}{N_0}}\right]}$$

Asymptotic coding gain $G = 1.76$ dB.



Uncoded BPSK
(4, 3) PC code, soft (simulations)
(4, 3) PC code, hard (simulations)
(4, 3) PC code, soft (ub, 1st term)
1.E+00
1.E-01
1.E-02
1.E-03
1.E-04
1.E-05
1.E-06
Bit Error Probability
1
2
3
4
5
6
7
8
9
E_b/N_0 (dB)

Example 6: ($n = 5$, $k = 4$) parity-check code ($d_{min} = 2$)

M				C				
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	0	1
0	0	1	1	0	0	1	1	0
0	1	0	0	0	1	0	0	1
0	1	0	1	0	1	0	1	0
0	1	1	0	0	1	1	0	0
0	1	1	1	0	1	1	1	1

M				C				
1	0	0	0	1	0	0	0	1
1	0	0	1	1	0	0	1	0
1	0	1	0	1	0	1	0	0
1	0	1	1	1	0	1	1	1
1	1	0	0	1	1	0	0	0
1	1	0	1	1	1	0	1	1
1	1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1	0

10 codewords at distance $d_{min} = 2$ from C_0 → $w_{dmin} = 16$

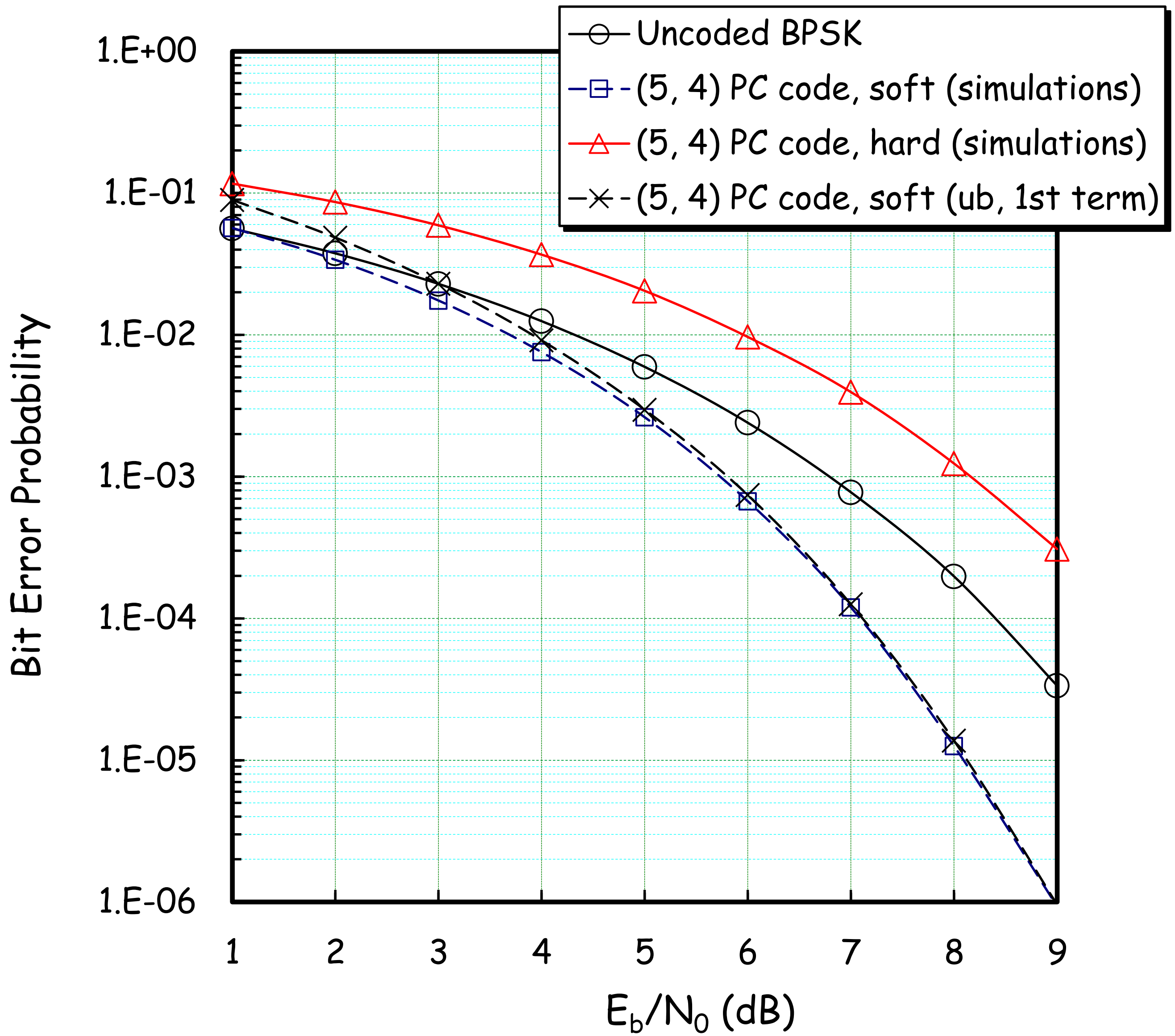
Example 6: (n = 5, k = 4) parity-check code (d_{min} = 2)

$$\boxed{P_{eb} \approx \frac{16}{2\times 4}\operatorname{erfc}\left[\sqrt{\frac{8}{5}\frac{E_b}{N_0}}\right] = 2\operatorname{erfc}\left[\sqrt{\frac{8}{5}\frac{E_b}{N_0}}\right]}$$

Asymptotic coding gain G = 2.04 dB.

As $k \to +\infty$, G → 10.log(2) = 3.01 dB.

However, the error coefficient values are increasing fast as $k \to +\infty$, which implies that this 3-dB asymptotic coding gain value is almost certainly not achieved at error probabilities of practical interest.


Uncoded BPSK
(5, 4) PC code, soft (simulations)
(5, 4) PC code, hard (simulations)
(5, 4) PC code, soft (ub, 1st term)
Bit Error Probability
1.E+00
1.E-01
1.E-02
1.E-03
1.E-04
1.E-05
1.E-06
1 2 3 4 5 6 7 8 9
E_b/N_0 (dB)

Conclusions

Soft-decision decoding outperforms hard-decision decoding by 2 - 3 dB: the more info we give to the decoder, the better it performs its task.

Excellent match between union bound and simulation results.

Very simple coding techniques can provide significant coding gains over uncoded BPSK.

However, we should be more ambitious by considering more complicated channel codes.