# DMcc - Channel Coding

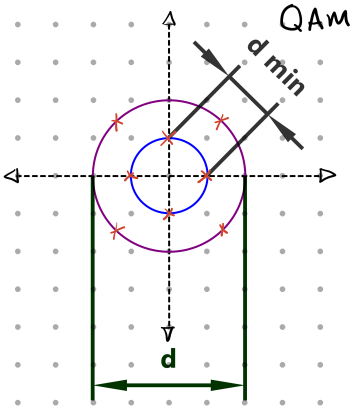


## Basic Ideas

- Channel Capacity relates to bit rate:
  - As long as Rate, R < Channel Capacity, C => Prob(error), $P(E) \simeq 0$
  - If $R > C$ then the performance is not acceptable
- To acheive performance as outlined above (good performance), we require:
  - Vector space dimensionality, $N \to \infty$
  - At recievers we must use real ($n \in \mathbb{R}$), not quantised/integer values, these are known as Soft decisions. → If we were working with hard values we would approximate the signals and remove information from them. → Shannon was, however, unable to achieve this result practically (despite trying).
- Turbo codes come very close to this so called "Shannon Capacity" and then a few years later LDPC codes were re-introduced with even better performance than turbo codes. Researchers today are still trying to achieve the "Shannon Capacity"

## Traditional Modulation and Error control codes

The main idea is to have the minimum possible distance between signals in a given Constellation:




- To increase the number of signals we would have to increase the distance, meaning an increase in power. (meaning introducing N.L's to our amplifiers)
- Instead, with coding we attempt to decrease the required separation.

### Classic Codes

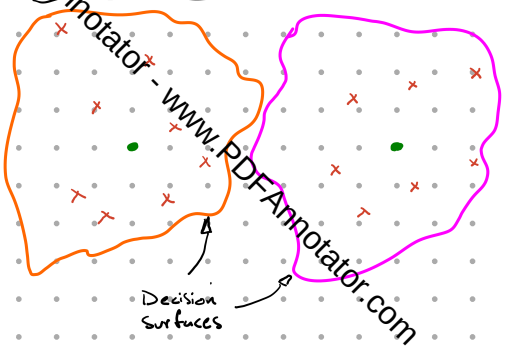
With classic codes we do not get anywhere close to the Shannon capacity as we need more bandwidth, more power, or both.
Additionally increasing N makes the system more and more complicated.
To solve these problems classic codes took a new approach to improve P(E):

- Time domain concatenation of signals → This creates symbols which are orthogonal
- Introduce a set of rules for sending a signal (encode the signal)

This still had one problem, the recieving systems could be impractically complex, with Complexity $\propto N$.
The codes eventually introduced were known as Forward Error Correction Codes, FEC codes:
- Linear Block codes / Parity check codes / Cyclic Redundancy Codes (CRC)...
- Convolutional Codes: Trellis codes / Tree codes...

The basic idea here was to discard some "possible signals" to increase $d_{min}$. It is clear from the start though that these will not get near shannon's Capacity, if the the signals are random and we have 1000 bits/word (signal), avoiding the mathematics we will have $10^{301}$ possible signals to decode.

# The Basic Idea of Channel Coding





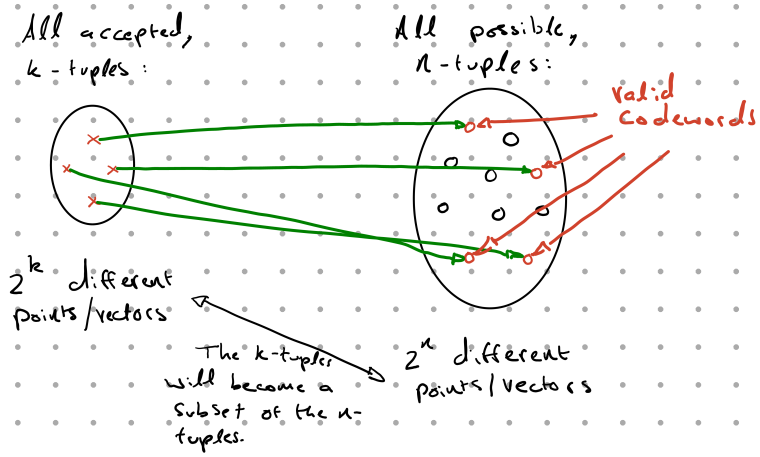

Legend:
- • = Defined Codeword (signal)
- x = Recieved signals/sequences (with errors)

The idea of channel coding is that we will define a decision region around a codeword and say that any signals in that region belong to the defined codeword.
OR we know that there has been an error and do not accept the transmission.

## Block Codes

We begin with a vector of index, k, and we map this vector onto all of our possible vectors.

$$\underbrace{(x_1, x_2, \ldots, x_k)}_{\text{Accepted codeword / signal vectors}} \overset{\text{(mapped onto)}}{\longmapsto} \underbrace{(y_1, y_2, \ldots, y_n)}_{\text{Possible codeword / signal vectors}}$$




This will allow us to increase $d_{min}$ and thus decrease $P(\epsilon)$

Some simple examples...

### Repetition Code

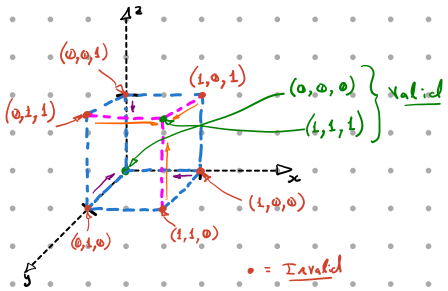
$N = 3 \quad k = 1 \quad \Rightarrow \quad m = \{0, 1\}$

$x = \{000, 001, \ldots, 111\}$

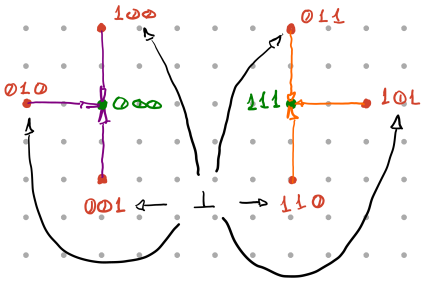
Valid code words:

$m = 0 \Rightarrow x = 000$

$m = 1 \Rightarrow x = 111$

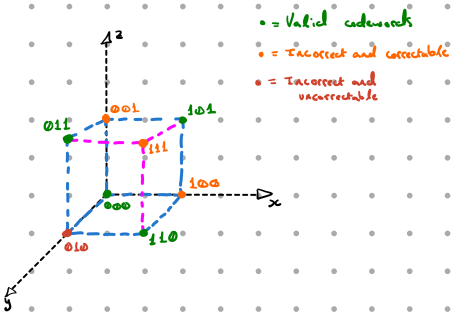



So the only acceptable codewords are as shown and any time we recieve an incorrect signal, we guess that there has been an error and correct it to the nearest signal




Note that each possible signal has a corresponding, fully orthogonal signal

### Parity check code (simple)



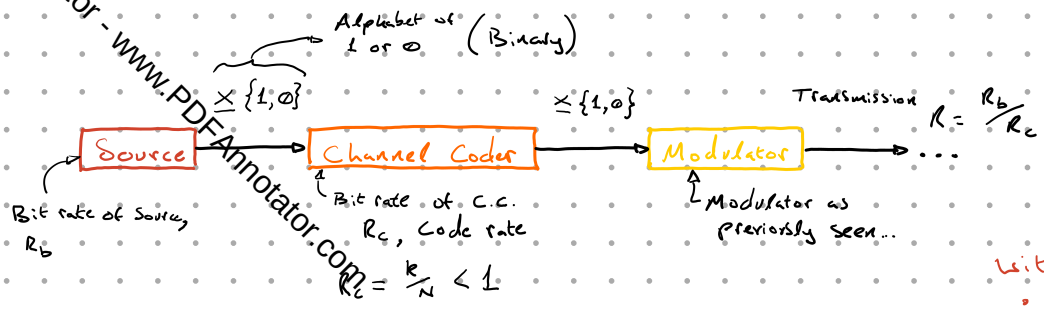

$k = 2 \quad N = 3$

Codewords all at a distance of two from eachother

Valid codewords:

$\{000, 011, 101, 110\}$

# Basic Coding Scheme

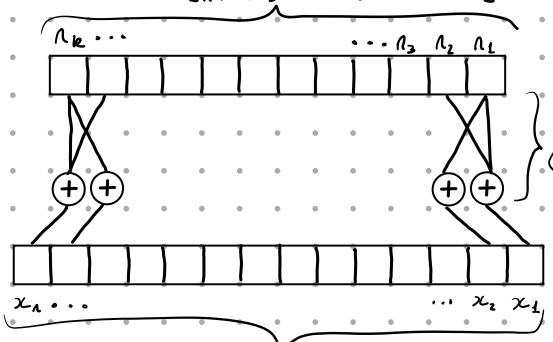



We shall differentiate between two general types of channel coder:

1. Error Detecting
2. Error Correcting

With a further two distinctions:
- Block coders
- Convolutional Coders

## Modern Block Codes - Linear Block Codes




$n > k$

These are called linear block codes as a linear operation is used to produce each codeword.

Of course the number of codewords is $2^k$

The generator matrix must have linearly independent vectors (rows).
i.e. given a vector, $g_i$, cannot be described by a linear combination of the other vectors (rows).

Each of these rows is called a basis vector (or just basis) of the vector space, C.
The dimensions of these vector spaces are the number of basis vectors we have, i.e. k
Finally each vector, $G_i \in C$, is a valid codeword

## An example, the Hamming Code (7, 4):

With information vector occupying the space: $U[k] = U(u_1, u_2, u_3, u_4)$ i.e. $(k=4) \preceq \{1,0\} \Rightarrow N = 2^4$

And a set of codeword vectors occupy: $x[n] = x(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ i.e. $n=7 \preceq \{1,0\}$

The relation between x and u which gives all valid codewords is:

$$x_i = \begin{cases} i = 1,2,3,4; & u_i \\ i = 5; & u_1 + u_2 + u_3 \\ i = 6; & u_2 + u_3 + u_4 \\ i = 7; & u_1 + u_2 + u_4 \end{cases}$$

Or in a table we define the vector space as:

| Data words | Code words |
|---|---|
| 0000 | 0000 000 |
| 0001 | 0001 011 |
| 0010 | 0010 110 |
| 0011 | 0011 101 |
| 0100 | 0100 111 |
| 0101 | 0101 100 |
| 0110 | 0110 001 |
| 0111 | 0111 010 |
| 1000 | 1000 101 |
| 1001 | 1001 110 |
| 1010 | 1010 011 |
| 1011 | 1011 000 |
| 1100 | 1100 010 |
| 1101 | 1101 001 |
| 1110 | 1110 100 |
| 1111 | 1111 111 |

# Important Definitions



Hamming weight of a codeword, $w(c_i)$ = number of non-zero elements in a codeword. ★ known as the hamming weight

Minimum weight of a code scheme, $w_{min}$ = the weight of the lightest codeword

For any linear code $d_{min} = w_{min}$ ← This is important as there is a theorem which states:

$i, v, \ldots$ = Information vector (bits), $1 \times k$ dimensional

$x, c, \ldots$ = codeword vector (bits), $1 \times n$ dimensional

$G \ldots$ = Codeword Generator matrix, $k \times n$ dimensional

$H \ldots$ = Parity check matrix, $(N-k) \times N$ dimensional

Codeword, $c = (\text{info}, i) \cdot (\text{Generator } G)$

$c = iG$

and

$(\text{Codeword}, c) \cdot (\text{Parity matrix}, H)^T = 0$

$c \cdot H^T = 0$

★★ PAY ATTENTION TO THE TRANSPOSE!!!

All codes can be defined with the forms:

$$G = [I_k \quad P] \Rightarrow H^T \begin{bmatrix} P \\ I_{n-k} \end{bmatrix}$$

with $c = \underbrace{c_1\ c_2\ c_3 \ldots c_k}_{i = i_1, i_2, i_3 \ldots i_k}\ c_{k+1} \ldots c_n$

Hamming distance between two codewords is the number of different bits between them

$d_H(a, b)$ where $a$ and $b$ are the codewords

★ RETURN TO THIS $t = ???$ ★

## Decoding Strategies - Maximum Likelihood (Hard Decision) or minimum distance.

Generally (as seen with the optimal reciever) we try to maximise the likelihood function and look for surfaces where we have equal probabilities between codewords. There is a number of techniques to achieve this, in this case of AWGN this is the same as minimising the distance between the recieved c.w. and the possible c.w's. With codewords we will work in the same way as it is most optimal. ★ ★

Mathematically this means:

$$\hat{e} = \underbrace{\operatorname{argmin} w_H(e)}_{\text{minimum weight}}, \text{ but; } w_H(e) = d_H(c, v)$$

$e : v - e \in C$

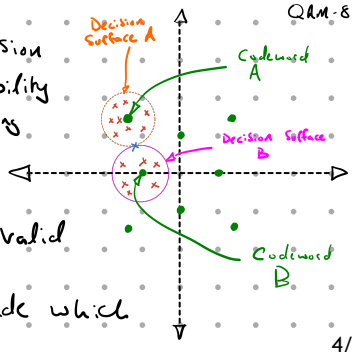
OR $\hat{e} = \hat{c}$

$$\Rightarrow \hat{c} = \underbrace{\operatorname{argmin} d_H(c, v)}_{\text{minimum distance}} \rightarrow \text{Minimum Distance criterion}$$

$c : c \in C$

The symbols × are codewords which are undefined, they are technically not valid, only • describes valid codewords.

But if we define a decision surface which has equal probability with neighbours and everything inside belongs to a given codeword, it is the same as finding what the nearest valid codeword is to a given ×




There is a point though where probabilities are equal and cannot decide which code word that signal belongs to, such as with ×.

# Decoding strategies - Syndrome

TODO MORE!!!



If we have errors during transmission

$\underline{X} \{1,0\}$

$\vec{x} \xrightarrow{\text{channel}(\dots)} \vec{y} = (y_1, y_2, y_3, \dots, y_n)$

From parity check:

$\vec{y} H^T = 0$

but if we have error:

$\vec{y} H^T \neq 0$

$\Rightarrow (\vec{x} + \vec{i}) H^T = \vec{i} H^T = \vec{S}$

Syndrome

The syndrome, $S$, is 0 if there is no errors or it a valid codeword was sent.

This means we can sometimes miss errors due to them still being valid codewords!!
We only detect errors that are not codewords.

# Cyclic Codes



These are a sub-set or sub-type of linear codes which are characterised by the fact that any bit shifts of the original codeword are also codewords.

★★ The bit shift must be circular

This does not only apply to the codewords but to the generator matrix also.

These codes have highly algebraic structure and are much easier to decode. Cyclic codes make heavy use of polynomials, eg:

Code/word of length $N$ = polynomial of degree $N-1$

$$\vec{a}(a_1, a_2, a_3 \ldots a_N) \longrightarrow a(D) = a_1 D^{N-1} + a_2 D^{N-2} + \ldots + a_{N-1} D + a_N$$

($D$ in this case likely means delay) we could use $z$ (z trans) or $x$ etc. etc.

Any operations on the polynomial only requires shift registers.

For any given cyclic code $(N,k)$ the codewords are generated using generator polynomials instead of matrices, eg: $g(D) = D^{N-k} \ldots + 1$ and all codewords are simply multiples of this polynomial.

Important, we can only say a polynomial, $g(D)$, is a generator polynomial of a cyclic codeword if:
- The generator polynomial is a factor of: $D^{N+1}$ ← {This is a complex algebraic property not proven here.
- Each factor of $D^{N+1}$ (of the degree $N-k$) generates a cyclic code $(N,k)$.

So if we consider a generator polynomial $g(D)$ with a degree, $r$. In order to create a codeword polynomial, $x(D)$ we "simply" multiply the generator polynomial by the information polynomial, $m(D)$:

$$x(D) = g(D) \cdot m(D) \qquad m(D) = m_1 D^{k-1} + \ldots m_k$$

$$x(D) = \{\text{degree/order}, N \leq k + r + 1\} \quad \text{and is linear}$$

Code will only be cyclic if $g(D)$ is a factor of $D^{N-1}$ (as stated before). However, it it is not systematic but we can make it systematic: (i.e. the input code is embedded in the codeword)

$$\left(m(D) D^{N-k}\right) + \text{remain.} \left\{\frac{m(D) D^{N-k}}{g(D)}\right\}$$

$$\therefore \quad x(D) = \underbrace{m_1 D^{N-1} + m_2 D^{N-2} \ldots + m_k D^{N-k}}_{\text{k-bits of information}} + \underbrace{r_1 D^{N-k-1} + \ldots + r_{N-k}}_{\substack{N-k \text{ bits of} \\ \text{remainder} \left\{\frac{m(D) D^{N-k}}{g(D)}\right\}}}$$

Inversly each codeword is divisible by $g(D)$ with no remainder and thus all errors can be noticeable and any remainders will be related to Syndrome.

## Important Cyclic Code example: Hamming

Very famous code scheme, it was the first "one error" correcting code. Part of a set or class of codes with:

$N = 2^{N-k} - 1$ ; eg's: (7, 4) (15, 11) (31, 26) (63, 57)...

$g(D) = D^3 + D + 1$ $\quad d_{min} = 3$ IN ALL CASES $\quad \therefore N \propto P(E)$ (P(E) quite high and increasing with dimensionality)

This coding scheme is the foundation for more sophisticated coding techniques.

$P(E) \cong 0 \Leftrightarrow N \simeq \infty$

For N small ($N \cong \infty$) transmission is not interesting anyway.

## Useful Cyclic Code Example: BCH (Bose, Ray-Chaudhuri, Hocquenghem) & Reed-Solomon Codes

This is a kind of extension of Hamming codes. It is a class of multiple-error-correcting code. For a message, m, with a number of errors, t:

$$n = 2^m - 1 \qquad k \geq n - mt$$

To correct a given number of errors the minimum hamming distance must be:

$$d_{H\,min} \geq 2t + 1$$

◊ BCH codes are very efficiently decoded and the selection of (n, k) is very flexible.

### Reed-Solomon codes

These are technically a subset of BCH codes. However, instead of being encoded in pure binary, they are encoded in blocks of bits.
They are, again, easily decoded and can correct a burst (sequence) of errors. These codes see heavy use with technologies including: CD, DVD, HDTV, PC memory, deep space exp...
Other info:

- Cyclic symbol, error correcting code
- Codeword, N, made of Information, I, and Parity, P.
- Commonly a Symbol/info block is made of 8-bits (not individual bits as with other schemes)

Scheme:

Symbol/info = m binary bits

Block length, n = $(2^m - 1)$ Symbols = $m(2^m - 1)$ binary bits

Parity checks, $(n-k) = 2t$ Symbols = $2mt$ binary bits

} Capable of correcting up to "t" errors. Very important for Concatenating coding schemes

# Modifying known Codes



- Puncturing – Delete a parity symbol $[(n,k) \rightarrow (n-1,k)]$
- Shortening – Delete a message symbol $[(n,k) \rightarrow (n-1,k-1)]$

  This requires setting the first b information bits to zero (as such this data isn't transmitted). Shortened codes are non-cyclic

- Expurgating – Delete a subset of codewords $[(n,k) \rightarrow (n,k-1)]$

  For cyclic codes with modulo $(\frac{d_{min}}{2}) \neq 0$ ($d_{min}$ odd) we can expurgate by multiplying the generator polynomial $g(D)$ by a factor, $D+1$.

  The order of $g(D)$ increases by one and k decreases by one. This expurgated code is cyclic, has an even $d_{min}$ (even number of 1s)

- Extending – Adding a parity symbol $[(n,k) \rightarrow (n,k+1)]$

  Usually performed on Hamming codes: (As $d_{min}=3 \Rightarrow d_{min}=4$)

  Codeword len, n remains the same but parity check increases by 1.
  We will have an even value for $d_{min} \geq$ original $d_{min}$
  New code is no longer cyclic

# Linear Block Codes - Performance Evaluations

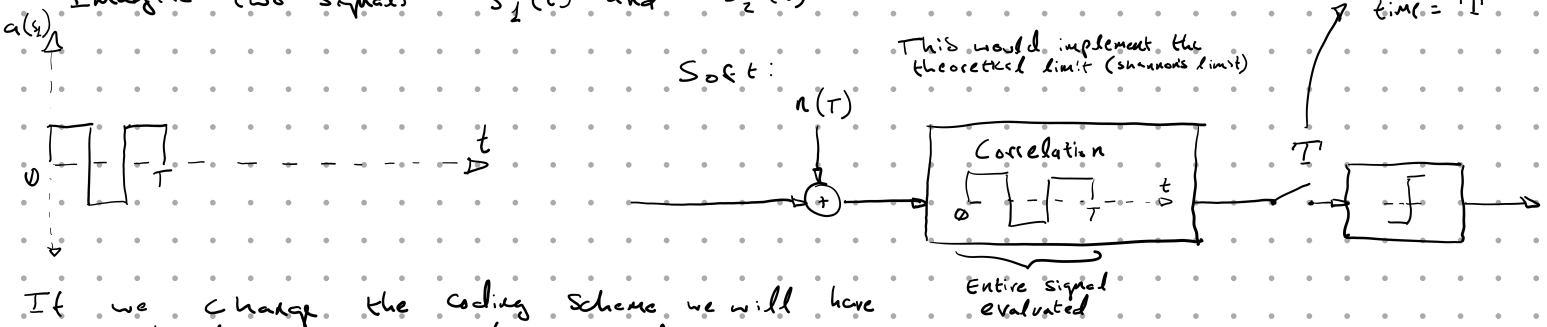


Soft Decision / Detection - optimal reciever, minimise Euclidean distance (in signal space (Euclidean)), and/or maximise the correlation of the likelihood function.

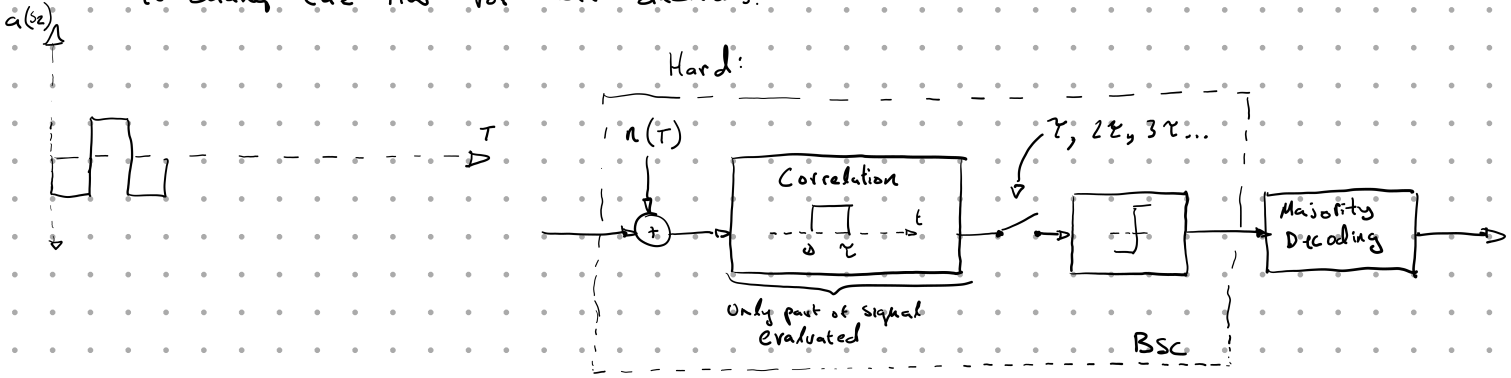
Hard Decision → uses vector / hamming space. (simpler than soft decision)
Usually has some signal loss → SNR drops ≈ 2 dB
Involves making a decision for each signal (deciding if it's 1 or 0) then assesing min. distance in the 1/0 vector space.
This is a sub-optimal scheme but is less complex to implement in HW.

IMPORTANT

Imagine two signals $S_1(t)$ and $S_2(t)$




If we change the coding scheme we will have to change the HW for soft decisions.




If we change the coding scheme will will only need to change the software for Hard decisions.

## Error Probability

We have two overall schemes to deal with here: hard decisions and soft decision. In the case of both the probability of having more than "t" errors:

$$P(E) = \sum_{h=t+1}^{N} \binom{N}{h} \varepsilon^h (1-\varepsilon)^{N-h}$$

Starts from t+1

As we will show this is the sum of the likelihood of having an error in just one bit.
i.e. We will only have errors when the number of errors possible exceeds t, the error correcting capability of the coding scheme.

For Gaussian channels with hard decision recievers we must define singular probability rate:

$$\varepsilon = Q\left(\sqrt{\frac{2E_b}{N_0}\frac{k}{N}}\right)$$

$k$ = No. of info bits
$N$ = No. of coduced bits
} $\frac{k}{N}$ = Code rate

$E_b$ = Energy of one bit
$N_0$ = Power Spectrum of noise.
} In Binary Anti-polar modulation $P(E) = Q\left(\sqrt{\frac{2E_b}{N_0}}\right)$

$\frac{2E_b}{N_0}$ = SNR

We can symptotically estimate this error:

$$P(E) \approx Q\left(\sqrt{\frac{2E_b}{N_0}\frac{K}{N}(t+1)}\right)$$

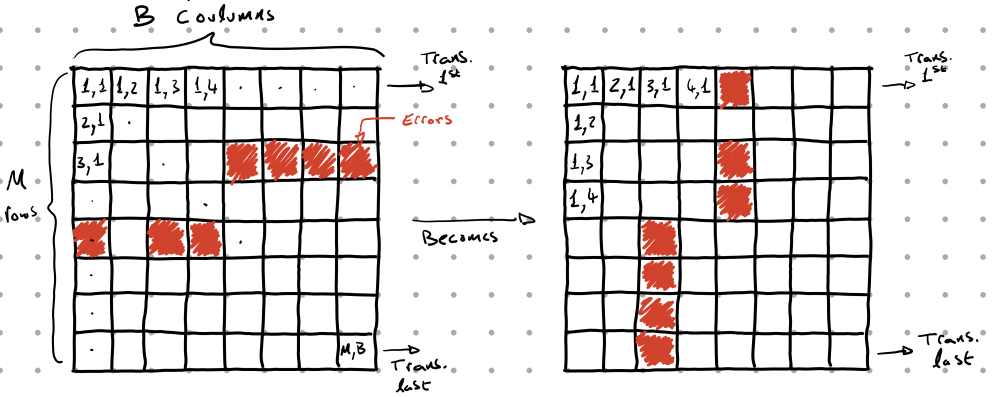
This should be compared to

Maximum likelihood Soft decoding:

$$Q\left(\sqrt{\frac{2E_b}{N_0}\frac{K}{N}\cdot d_{min}}\right)$$

## Interleaving

This relates to burst errors (sequential bit errors), this type of error is very common in wireless systems where an incorrect transmission T.F. causes a number of bits to be effected. Classically, this is solved by interleaving bits so the errors are separated.




This correction technique is no longer used (more sophisticated solutions, which are too complex for this course).
It requires more memory and a large amount of delay at differents steps of the process.

DMCC

State Diagrams



Trellis Digrams

Tree Diagrams