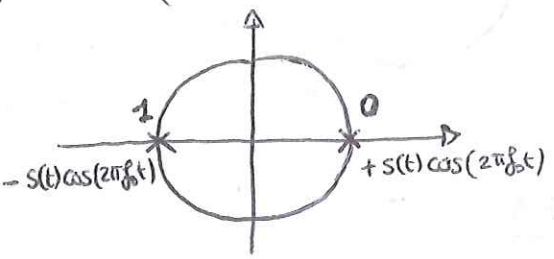
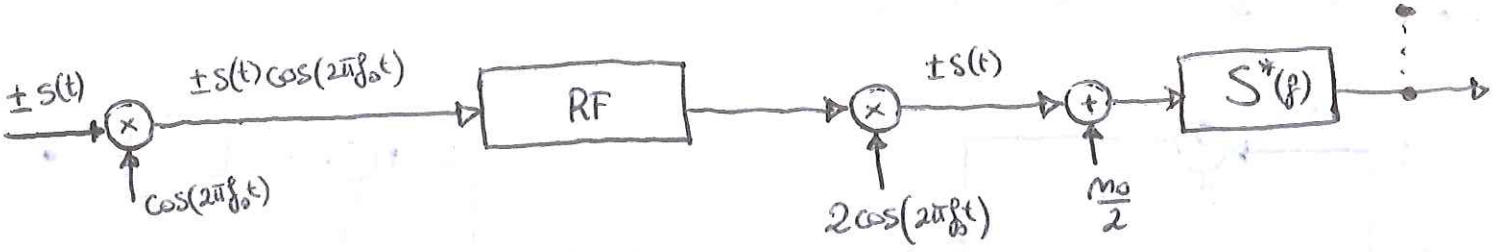
BANDA TRASLATA – MODULAZIONE

- **2PSK (Phase Shift Key):**

Segnale con 2 FASI: quindi $\cos(2\pi f_0 t)$ oppure $\cos(2\pi f_0 t + \pi) = -\cos(2\pi f_0 t)$

Quindi ho i segnali: $\pm s(t)\cos(2\pi f_0 t)$

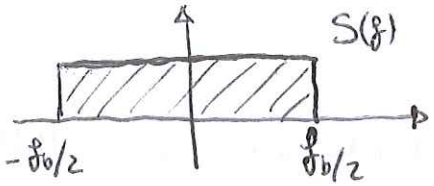




Ma, HO SOLO TRASLATO IL SEGNALE IN FREQUENZA.

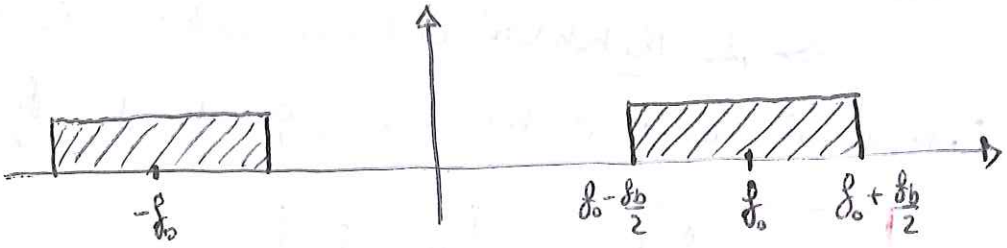
Infatti: $\pm s(t)\cos(2\pi f_0 t) = \pm \frac{s(t)}{2}\left[e^{j2\pi f_0 t} + e^{-j2\pi f_0 t}\right]$ e quindi passo da

BANDA BASE:



$$B_N = \frac{f_b}{2}$$

BANDA TRASLATA:



ATTENZIONE: ora la BANDA MONOLATERA è $B_N = f_0 + \frac{f_b}{2} - \left(f_0 - \frac{f_b}{2}\right) = f_b$!

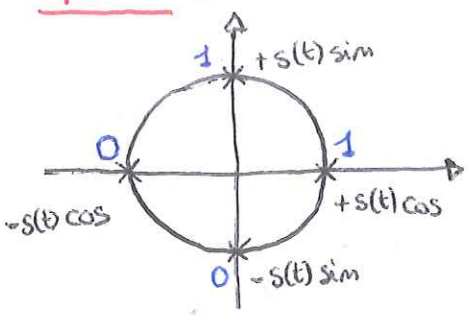
$$\Rightarrow \boxed{B_N = f_b}$$

* RADDOPPIA la BANDA! (svantaggio)

* Ma ho sempre 2 segnali, con 1 bit per segnale $\Rightarrow$ P_e e P_b NON CAMBIANO!

$$\boxed{P_b = \frac{e^{-\gamma}}{3\pi}}, \text{ ma } \boxed{B_N = f_b}$$

• 4PSK :

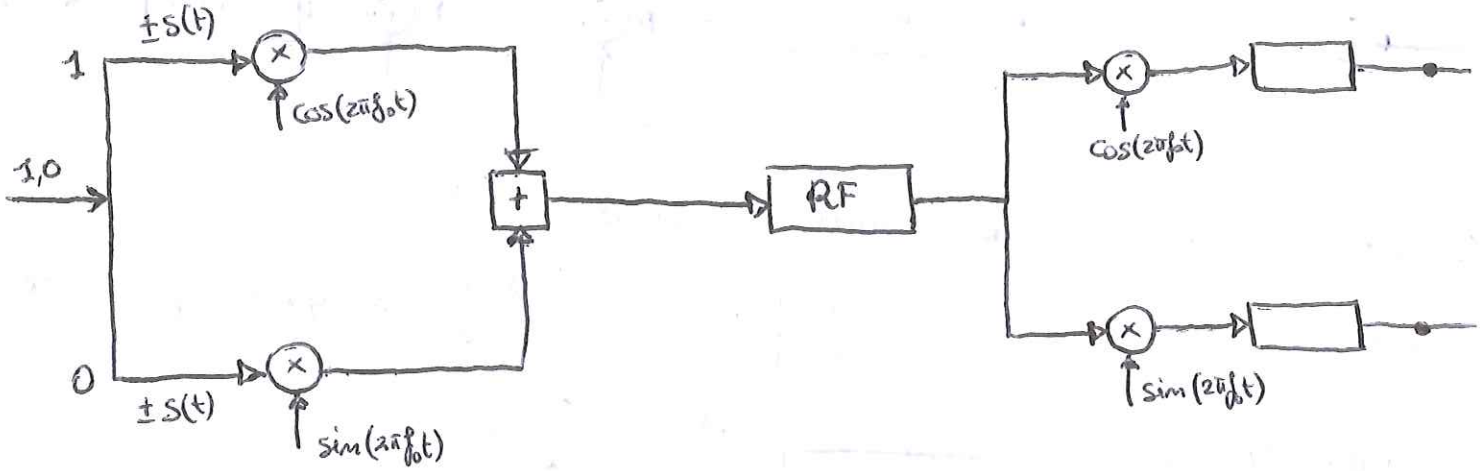


Ho i segnali : $\pm s(t)\cos(2\pi f_0 t)$
$\pm s(t)\sin(2\pi f_0 t)$

In pratica ho 2 CANALI BINARI, uno con fase $\cos(2\pi f_0 t)$ e l'altro con fase $\sin(2\pi f_0 t)$.
Su entrambi posso mandare 1 $(+s(t))$ oppure 0 $(-s(t))$.

Le FASI sono : $0, \frac{\pi}{2}, \pi, \frac{3}{2}\pi$.

Se il bit è 1 lo mando sopra, se è 0 lo mando sotto.



* ATTENZIONE : I due canali, in quanto ortogonali, sono INDIPENDENTI !

⇒ La Probabilità di Errore di bit è la STESSA di una TX BINARIA!

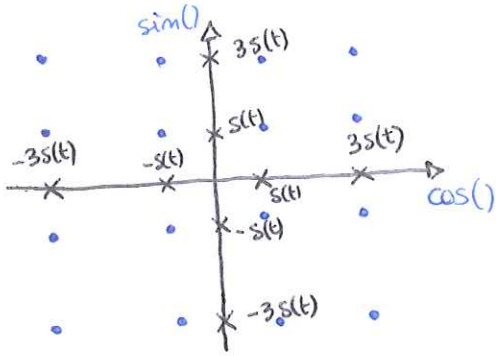
Tuttavia, il numero di bit è 2, quindi $f_s = \frac{f_b}{2}$ ⇒ La banda si dimezza!

$$P_b = \frac{e^{-\gamma}}{3\pi}$$

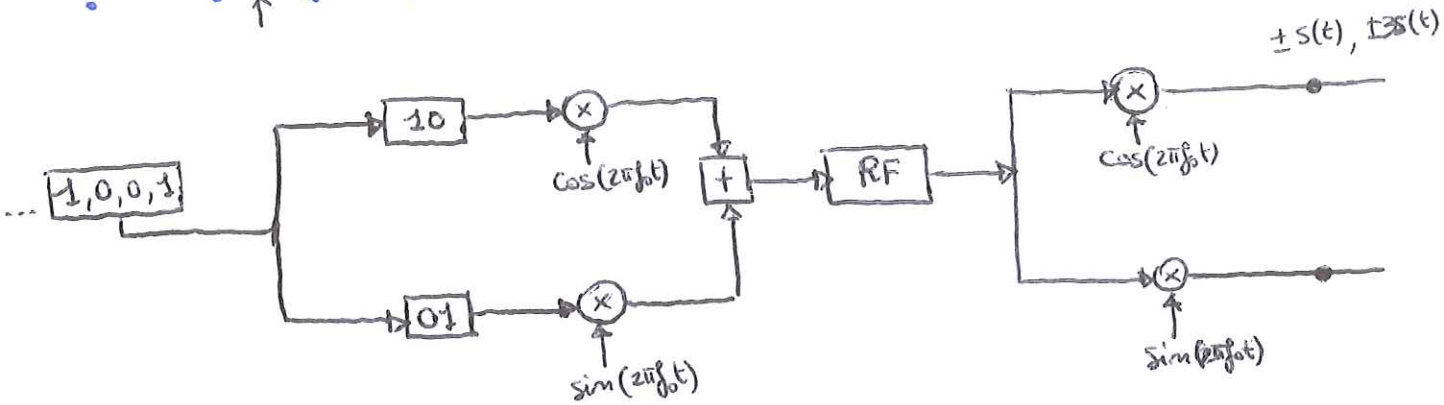
$$B_N = \frac{f_b}{2}$$

• 16 QAM (Quadrature Amplitude Modulation):

Ho sempre 2 CANALI con sin() e cos(), ma con TX QUATERNARIA;



Quindi ho: $\pm s(t)\cos(2\pi f_0 t)$
$\pm s(t)\sin(2\pi f_0 t)$
$\pm 3s(t)\cos(2\pi f_0 t)$
$\pm 3s(t)\sin(2\pi f_0 t)$



La probabilità di errore di bit è quella della TX QUATERNARIA:

$$P_b = \frac{1,5\, e^{-\gamma/2,5}}{2 \cdot 3\pi}$$

La BANDA MINIMA è $B_N = \frac{f_b}{4}$.

* La Probabilità che un simbolo sia corretto, equivale a non sbagliare alcun bit, quindi:

$$P_{S,\text{corretto}} = (1-P_b)^4$$

• 64 QAM:

2 CANALI OTTONARI $\Rightarrow$ $P_b = \frac{7\, e^{-\gamma/7}}{3 \cdot 4 \cdot 3\pi}$; $B_N = \frac{f_b}{6}$

• 256 QAM:

2 CANALI A 16 LIVELLI $\Rightarrow$ $P_b = \frac{2\, e^{-\gamma/21}}{4 \cdot 3\pi}$; $B_N = \frac{f_b}{8}$

CODICI A CORREZIONE D'ERRORE

Nella 4PSK, quindi con $B_N = \frac{f_b}{2} \rightarrow \frac{f_b}{B_N} = 2$, con un tasso d'errore $e \simeq 10^{-6}$, bisogna avere un $\gamma_{min} = \frac{E_b}{n_0} \simeq 10{,}5\ dB$.

SHANNON: si può fare un canale IDEALE con $\frac{E_b}{n_0} = 1{,}7\ dB$!

La distanza è enorme! Questi sistemi di telecomunicazione sono poco efficienti. Potrei abbassare γ, ma P_b aumenterebbe esponenzialmente!

SOLUZIONE: Abbasso γ, ma poi CORREGGO GLI ERRORI!

- Perché si sbaglia?

Si sbaglia quando almeno 1 bit viene invertito; prendiamo 2 parole:

1 0000 0000
0 0000 0000 $\quad I \quad (d=1)$

DISTANZA = # bit di differenza

Per poter correggere, devo sapere, nel caso di errore, la probabilità che quel simbolo sia realmente sbagliato o no → DEVO AUMENTARE d!

Aggiungo dei PARITY BIT, 11 sopra e 00 sotto, per aumentare la distanza:

111 000 000 ...
000 000 000 ... $\quad \updownarrow \ (d=3) \rightarrow$ se sbaglio 1 SOLO BIT, allora posso CORREGGERLO!

* Se però sbaglio 2 bit, il ricevitore commette un ERRORE.

⇒ Aggiungo 4 bit anziché 2:

11111 00000 00000 ...
00000 00000 00000 ... $\quad \updownarrow \ (d=5) \rightarrow$ Posso correggere FINO A 2 ERRORI!

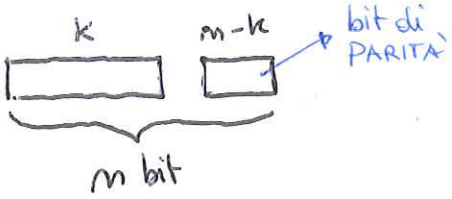
Dunque, abbiamo le seguenti importanti relazioni:

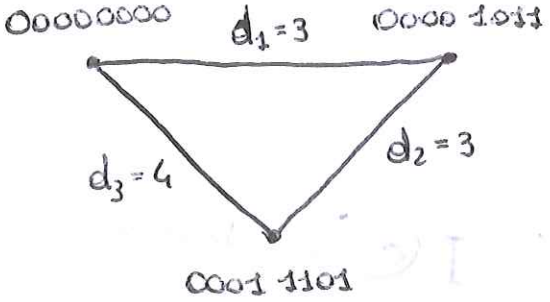
Sia t il numero di errori correggibili e d la distanza minima tra le parole:

- $\boxed{t = \frac{d-1}{2}}$
- $\boxed{d = 2t+1}$; per poter correggere t errori, la distanza minima tra le parole deve essere pari al # di errori che si vogliono correggere più 1!
- $\boxed{t_{rilevare} = d-1}$: se invece il codice RILEVA solo gli errori, con una distanza d posso rilevare fino a $d-1$ errori.

- Esempio: 3 PAROLE:

Siano le PAROLE INFORMATIVE di k bit e le PAROLE CODIFICATE di n bit:





Ho 2^k parole informative e 2^n parole codificate,

$d_{min} = d = 3 \Rightarrow$ posso correggere $t = \frac{d-1}{2} = 1$ ERRORE!

se d fosse stato minore di 3, non sarei stato in grado di correggere nessun errore!

(1) CODICI A BLOCCO:

Il flusso informativo originario viene diviso in blocchi di K bit e ad ognuno si aggiungono $n-k$ bit di parità, in modo da ottenere una PAROLA CODIFICATA di lunghezza n bit.

Definiamo CODING RATE: $\boxed{R_c = \frac{k}{n}}$ il rapporto tra bit informativi e bit totali.

- Se il BIT RATE è COSTANTE:

  allora non cambia la banda del canale, ma cambia il THROUGHPUT, poiché aggiungo overhead;
  ma, se la potenza di ricezione è la stessa, poiché i bit nell'unità di tempo trasmessi sono gli stessi $\Rightarrow$ γ' dopo codifica RIMANE lo STESSO di γ

  $$\boxed{\gamma' = \gamma}$$

- Se il THROUGHPUT è COSTANTE:

  allora AUMENTA la BANDA: $B' = B \cdot \frac{n}{k}$ e diminuisce il γ' dopo codifica: $\boxed{\gamma' = \frac{k}{n} \cdot \gamma}$

Quindi, la probabilità d'errore di bit DOPO CODIFICA è maggiore di quella di un bit non codificato (cioè $e^{-\gamma}$)!

Quindi: $\boxed{P_{bc} > P_b}$

(2) CODICI BCH:

E' un CODICE BINARIO: ogni simbolo è 1 bit.

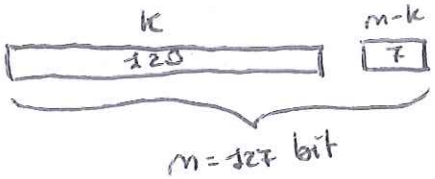
Supponiamo di avere una parola codificata di $n = 127$ bit;

- Come correggere 1 errore?

  Ho bisogno di:
  - 1 configurazione nel caso non ci siano errori;
  - 127 configurazioni per individuare dov'è l'errore.

$\Rightarrow$ 128 configurazioni $\Rightarrow$ Ho bisogno di $(128 = 2^7)$ $\boxed{7 \text{ BIT}}$!

Dunque ho $n-k=7$ bit di parità e codifico le parole nel seguente modo:



* Hamming trovò il codice che faceva questo, ma è molto matematico!

- Se volessi correggere 2 errori?

Ho bisogno di:
- 128 config. precedenti;
- 1 config. se non c'è il 2° errore;
- 126 config. per vedere dov'è il 2° errore.

} 255 configurazioni

$\Rightarrow 255 < 256 = 2^8$, ma non si ragiona così!

Ho bisogni di altre 127 config. $\Rightarrow$ **ALTRI 7 bit!**

$\Rightarrow n-k = 14$, $k = 127-14 = 113$!

In generale, ho codici BCH (n, k, t):

- BCH $(127, 120, 1)$: corregge 1 errore con 7 bit
- BCH $(127, 113, 2)$: corregge 2 errori con 14 bit
- BCH $(127, 106, 3)$: corregge 3 errori con 21 bit

...

* I codici BCH sono efficienti solo per un numero basso di errori!
Un codice BCH esiste per qualsiasi h ed è lungo $\boxed{n = 2^h - 1}$, dove h è anche il numero di bit necessari per correggere 1 errore.

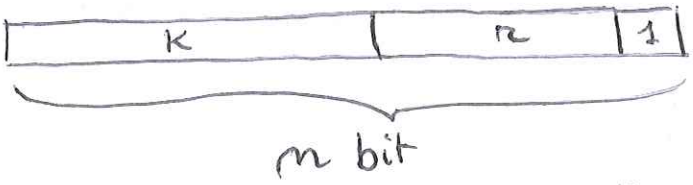
* Se n è PRIMO: p.es. $127 = 2^7 - 1$:
  - 7 bit → 1 errore
  - 2·7 bit → 2 errori
  - 3·7 bit → 3 errori
  - ...

* Se n non è PRIMO: p.es. $63 = 2^6 - 1$:

6 oppure dei SOTTOMULTIPLI di 6

- 6 bit → 1 err.
- 2·6 bit → 2 err.
- 3·6 bit → 3 err.
- 4·6 bit → 4 err.
- (4·6+3) bit → 5 err

Si può aggiungere 1 BIT di PARITÀ GLOBALE, in modo tale che n sia pari; in tal modo, aumenta la distanza di 1: $d' = d+1$!



* La capacità di CORREZIONE è la STESSA! Ma aumenta di 1 la capacità di rilevazione!

$$BCH(127, 113, 2) \longrightarrow BCH(128, 113, 2)$$

- PROBABILITÀ ERRORE DI PAROLA:

Sbaglio una parola se sbaglio almeno d_{min} bit, cioè quando sbaglio più bit di quelli che posso correggere!
Se il codice corregge t errori, non sbaglio se # errori $\leq t$!

$$1 - P_p = \sum_{h=0}^{t} \binom{n}{h} P_{bc}^{h} (1 - P_{bc})^{n-h}$$

dove:
P_p = Prob. errore di PAROLA CODIFICATA, lunga n bit
P_{bc} = Prob. errore di bit dopo codifica

- BIT RATE COSTANTE: $\gamma' = \gamma \Rightarrow$ $P_{bc} = P_b = \frac{e^{-\gamma}}{3\pi}$

- THROUGHPUT COSTANTE: $\gamma' = \frac{k}{n}\gamma = R_c \cdot \gamma \Rightarrow$ $P_{bc} = \frac{e^{-\gamma'}}{3\pi} = \frac{e^{-\gamma \cdot R_c}}{3\pi}$

In generale, si ha la seguente approssimazione:

$$P_{bc} \simeq P_p \cdot \frac{d}{n}$$

(8) CODICI DI READ SOLOMON :

Sono codici non binari, bensì ORIENTATI AL SIMBOLO

$n = 2^h - 1$; se, per esempio, $n = 127$ → ho 127 SIMBOLI !

⇒ 1 SIMBOLO = h bit !

- Di quanti simboli ho bisogno per correggere 1 errore?
  - 1 config. per 0 errori;
  - 127 config. per individuare in quale simbolo c'è l'errore
  - altre config. per vedere quale bit del simbolo presenta l'errore

} 128 = 7 bit = 1 SIMBOLO + 1 SIMBOLO per errore all'interno del simbolo

⇒ Per 1 errore ho bisogno di **2 SIMBOLI** ! ⇒ $n = 2 \cdot t$

* Per correggere t ERRORI → ho bisogno di **2t SIMBOLI**

- PROBABILITÀ ERRORE DI PAROLA :

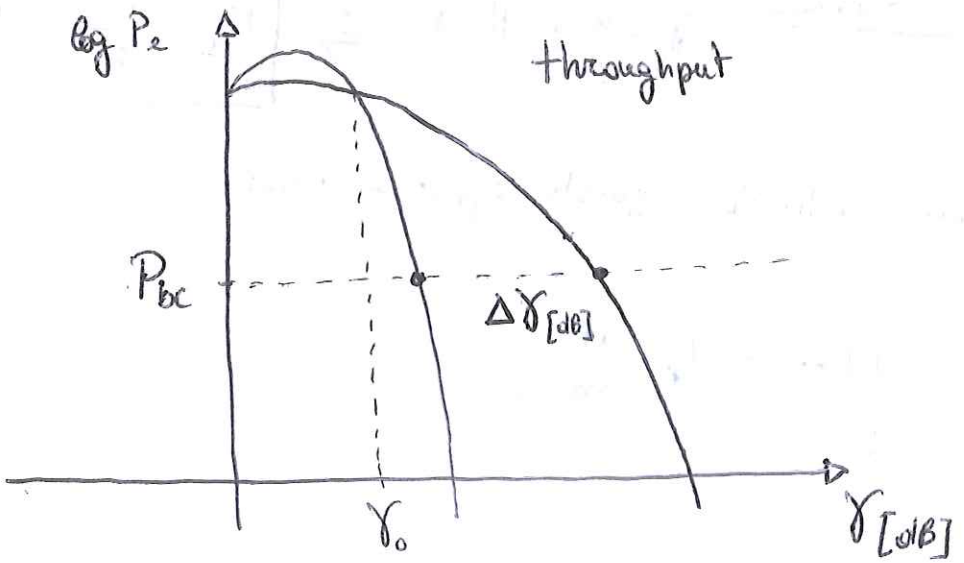
$$1 - P_p = \sum_{h=0}^{t} \binom{n}{h} P_{sc}^{h} \left(1 - P_{sc}\right)^{n-h}$$

dove: P_{sc} = Probabilità d'errore di SIMBOLO DOPO CODIFICA

- CODING GAIN :

E' il GUADAGNO DI CODICE, dovuto alla diminuzione di γ

GAIN : = $\Delta\gamma_{[dB]}$



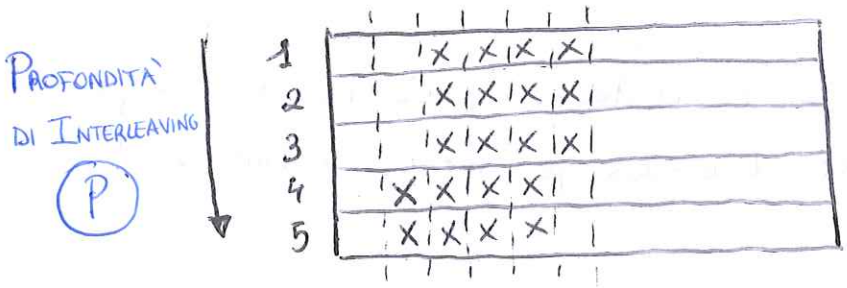
- **INTERLEAVER:**

I problemi seri sono dovuti agli ERROR BURST, cioè burst di errore che coinvolgono più bit o più simboli.

> **ERROR BURST** := intervallo che inizia e termina con 1 ERRORE e in mezzo ha un numero arbitrario di errori.

Supponiamo di avere un codice che corregge $t = 4$ errori:

trasmettiamo PER COLONNE 5 parole di n bit:



→ Può correggere al massimo 4 errori per parola

* **INTERLEAVER:** trasmette le parole per colonne, così da poter sostenere ERROR BURST più lunghi di t!

Se non ci sono altri errori:

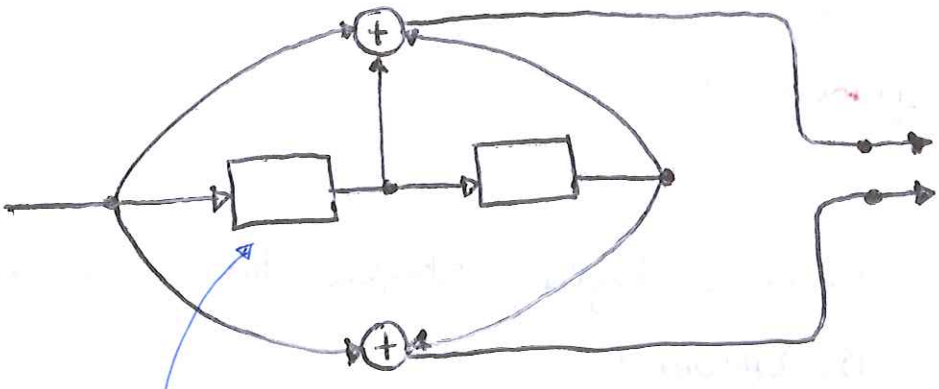
$$\text{ERROR BURST MAX} = e.b. = p \cdot t$$

, con p = profondità di interleaving
t = errori correggibili

* **SVANTAGGI:** c'è molto (RITARDO) dovuto alla MATRICE e alla sua codifica!

(4) CODICI CONVOLUZIONALI:

Usano il seguente codificatore "convoluzionale":



BUFFER DI RITARDO

Entra 1 bit e me escono 2!

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

Supponiamo di avere 2 parole a distanza $d=1$ in ingresso:

0000 0000
1000 0000

* Il sistema è LINEARE (LTI), quindi vale la sovrapposizione degli effetti e le seguenti considerazioni valgono per tutte le parole a distanza 1!

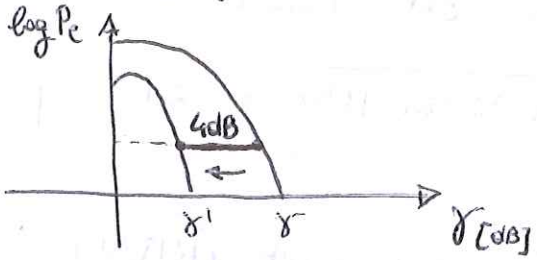
output 1° parola: 00 00 00 00 00 00 00 00
output 2° parola: 11 10 11 00 00 00 00 00

⇒ La DISTANZA è aumentata da $d=1$ a $d'=5$!

⇒* γ aumenta di un fattore 5 per la distanza, ma devo dividere per 2 perché, raddoppiando i bit, l'energia per bit E_b si DIMEZZA!

⇒ $$P_b = \frac{e^{-\frac{5}{2}E/N_0}}{3\pi}$$

Ho un CODING GAIN di $\Delta\gamma_{[dB]} = 10\log_{10}\left(\frac{5}{2}\right) \simeq \boxed{4dB}$, calcolando le PRESTAZIONI ASINTOTICHE:



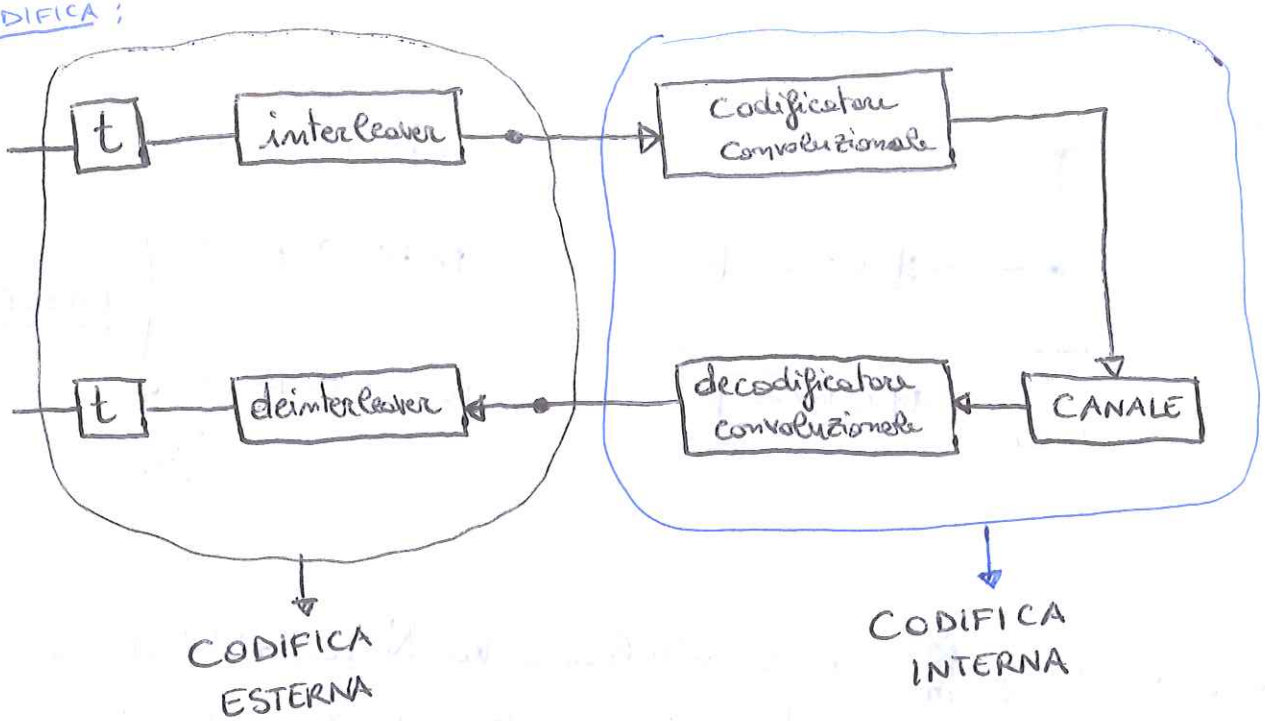
Se definiamo: d_{free} = distanza minima DOPO CODIFICA

Allora abbiamo: $$P_p \simeq \frac{e^{-d_{free}\cdot R_c\cdot\gamma}}{3\pi}$$

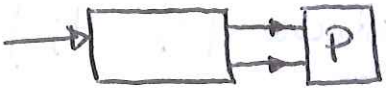
* Inoltre: $\boxed{P_b \equiv P_p}$ IMPORTANTE!!!

Quando un codice convoluzionale sbaglia, sbaglia tutto un blocco, producendo un BURST DI ERRORE!

- DECODIFICA:



- Un altro difetto è il RATE BASSO: di solito $R_c = \frac{1}{2}$ o $R_c = \frac{1}{3}$; si può risolvere utilizzando una PUNTURAZIONE: si scelgono solo alcuni dei bit in uscita dal codificatore:



; con $P = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ (bit 1, bit 2)

si prendono entrambi i bit d'uscita per il 1° bit in ingresso; invece si prende solo il bit di sotto per il 2° bit in entrata.

In questo modo: $R_c = \frac{2}{3}$, poiché entrano 2 bit e ne prendo 3 per codificare.

Se $P = \begin{bmatrix} 101 \\ 110 \end{bmatrix} \Rightarrow R_c = \frac{3}{4}$; se $P = \begin{bmatrix} 1011 \\ 1100 \end{bmatrix} \Rightarrow R_c = \frac{4}{5}$

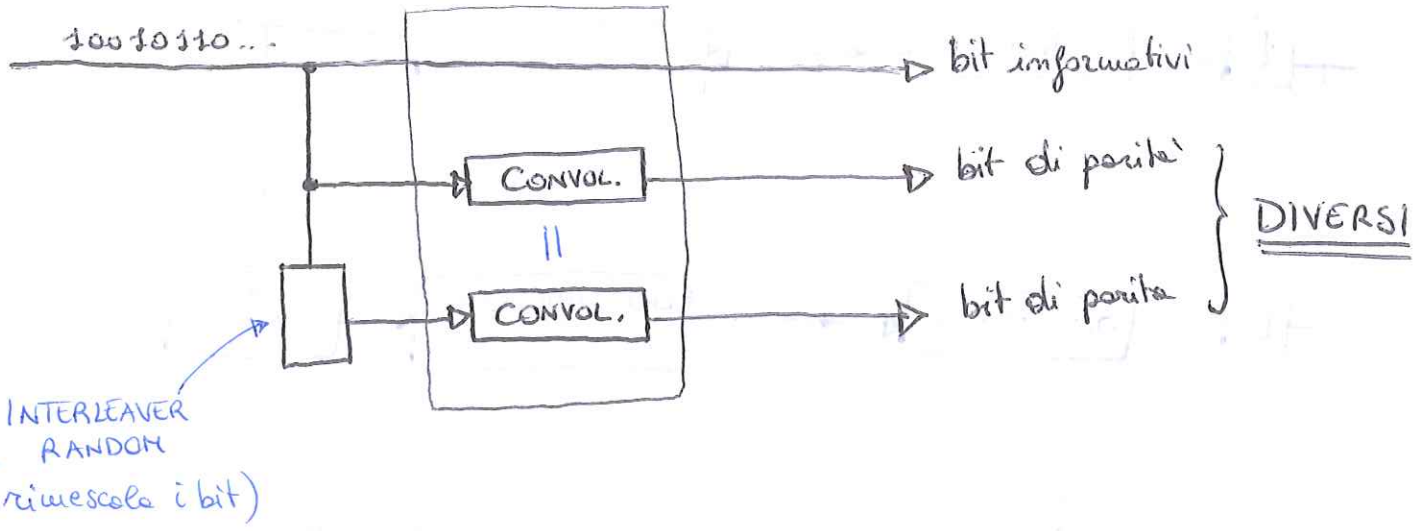
* COMPROMESSO:

purtroppo, con la 1ª punturazione, la DISTANZA DIMINUISCE a

$d_{free} = 3 \Rightarrow P_p = \frac{e^{-\gamma \cdot 3 \cdot \frac{2}{3}}}{3\pi} = \frac{e^{-2\gamma}}{3\pi}$

quindi ho un MINOR GUADAGNO $\Delta\gamma_{[dB]}$, ma un MIGLIOR CODING RATE R_c !

(5) TURBO CODICI:

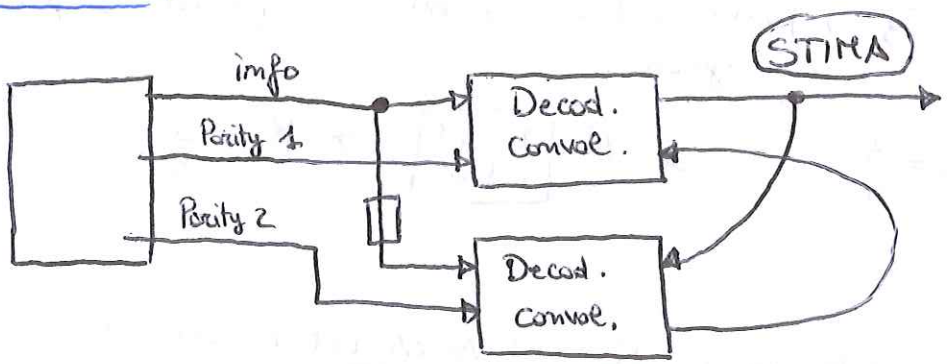


Con una parola lunga m, con l'interleaver ho N possibilità di rimescolare i suoi bit, quindi la probabilità che riesca la stessa parola e che, quindi, i 2 codificatori convoluzionali diano lo stesso risultato è $\frac{1}{N}$ → molto basso.

Il coding rate è $R_c = \frac{1}{3}$

Quindi: $P_p = \frac{1}{N} \frac{e^{-d_{free} \cdot \frac{1}{3} \cdot \gamma}}{3\pi}$; Anche in questo caso: $P_b \equiv P_p$!

- DECODIFICA:



CONFRONTANO LE STIME ITERATIVAMENTE per decidere bene!

* PROBLEMI: Parole molto grandi hanno N molto grande + iterazioni = (LENTO!)

Inoltre il CODING RATE $R_c = \frac{1}{3}$ è molto BASSO!

Però si avvicinano moltissimo ai 9 dB di Shannon.

(6) Low Density Parity Check (LDPC):

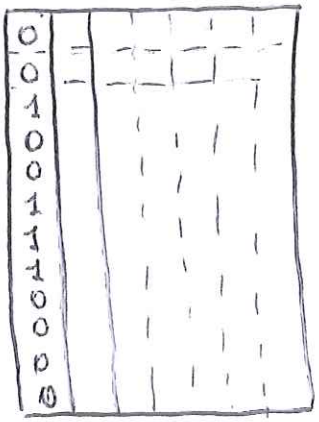
CODICE A BASSA DENSITÀ → ci sono molti bit di parità e posso allontanare le parole, ma le parole codificate sono molto lunghe!

informative	parità
0000 0000	0000 0000
1000 0000	1111 0000
0100 0000	0000 1111

$\Rightarrow d_{min} = d_{free} = 5$

- IN RICEZIONE:

Sia una parola ricevuta $C_1, C_2, \dots, C_{12}$; abbiamo la seguente LEGGE:



Se il PRODOTTO RIGHE per COLONNE dà 0 ⇒ non ci sono errori!

La MATRICE si costruisce con parametri matematici opposti!

$$C_3 + C_6 + C_7 + C_8 = 0$$

Se ho degli errori, il prodotto righe per colonne non verrà 0!
Allora, come nel caso di turbo codici, si cerca di STIMARE le probabilità delle varie parole di essere 0 oppure 1 per avere come risultato zero.

* VANTAGGI:

(1) VELOCE: non ho codici da confrontare, bensì equazioni;

(2) RATE AGGIUSTABILE: tramite punturazioni.