Introduction aux Turbo-codes

unité RNUM
Pascale Jardin

Introduction

Les Turbo codes ont été présentés en 1993 par Berrou et Glavieux et Thitimajshima (ref 1). Ils ont apporté la plus grande avancée en codage depuis les travaux d' Ungerboeck qui avait introduit les Modulations codées (Trellis-Coded Modulation) en 1982. Ils exploitent toutes les potentialités des schémas de codes concaténés proposés par Forney en 1966.
On peut en effet voir les turbo-codes comme un raffinement d'une structure de codage concaténé avec la mise en place d'un décodage itératif à sortie souple. Cette sortie souple permettra de déterminer les symboles émis à partir d'une règle MAP (maximum a posteriori).

La structure basique d'un schéma de turbo (dé)codage est en effet d'utiliser une concaténation en parallèle d'au moins deux encodeurs avec un entrelaceur entre les encodeurs. Le décodage est basé sur le décodage alternatif des codes composant le codeur et de passer à chaque étape l'information « extrinsèque » issue de la décision souple d'un décodage à l'entrée du décodeur de l'étape suivante. C'est ce re-bouclage du décodage qui est à l'origine de l'appellation « turbo ».
La conception des turbo-codes n'a pas consisté en l'optimisation d'un nouveau critère mais est le résultat d'un processus expérimental où la simulation a été utilisée pour minimiser le taux d'erreurs sur une communication, ce qui est le véritable objectif du codage canal. En particulier la taille de l'entrelaceur s'est révélée déterminante pour obtenir de bons résultats.
Les performances étonnantes obtenues (beaucoup plus proches de la limite de Shannon (ref 2) que toutes les techniques présentées antérieurement) ont suscité un immense intérêt dans la communauté scientifique et un grand nombre de travaux, tant de « justifications » et développements théoriques que d'extensions de l'approche « turbo » pour certaines applications telles que l'égalisation et la détection multi-utilisateurs.

Nous rappellerons dans un premier temps l'objectif du codage, la limite théorique qui est donnée par la capacité du canal de transmission, les notions de rendement et de distance dans les codes « classiques » et le « gain de codage » résultant.
Nous aborderons ensuite les deux aspects fondamentaux dans l'approche turbo :
la décision souple qui permet une actualisation de probabilités d'évènements et permet de mieux exploiter l'information acquise lors d'un décodage.
l'itération de deux (ou plus) décodages utilisant l'information « extrinsèque » (dépendante uniquement des bits de parité) d'un décodage pour le décodeur suivant.
Ces deux notions seront illustrées dans l'exemple didactique d'un code de parité produit.
Nous décrirons alors les composantes fondamentales du turbo-code « classique » : deux codeurs convolutifs récursifs systématiques fonctionnant en parallèle et séparés par un entrelaceur et nous verrons comment ce choix permet d'obtenir des codes dont la distribution de distances est semblable à celle, optimale, des codes aléatoires.
Enfin nous étudierons la structure du décodeur itératif correspondant qui met en œuvre deux algorithmes BCJR-MAP.

Les turbo-codes sont actuellement utilisés dans les systèmes DVB. Ils seront intégrés dans la norme UMTS.

1. Petits rappels en théorie de l'information et approche classique du codage

1.1. Entropie d'une source

1.1.1. Définition

Si on considère une source numérique stationnaire délivrant des symboles X_k prenant leur valeur dans un alphabet $A=(x_1,x_2,\ldots,x_M)$ avec la distribution de probabilité $p_i=Pr(X_k=x_i)$. L'entropie (ou information moyenne) de cette source est donnée par :

$H(X) = -\sum_{i=1}^{M} p_i \log_2 p_i$ (en bits d'information ou Shannons)

ex : une source binaire avec des symboles 0 et 1 équiprobables a une entropie de 1 Sh
si la distribution de probabilité de la source est p(0)=.1 et p(1)=.9 l'entropie vaut 0.469 Sh

1.1.2. Probabilité d'une longue suite de symboles (approche intuitive)

Si on considère un message constitué de N symboles (avec N grand) , ce message contiendra avec une haute probabilité p_1N occurrences du symbole x_1, p_2N occurrences du symbole x_2, …, p_MN occurrences du symbole x_M.

La probabilité d'un tel message sera approximativement $p \simeq p_1^{p_1 N} p_2^{p_2 N} \ldots p_M^{p_M N}$.

L'information de ce message, donnée par le log en base 2 de l'inverse de cette probabilité, sera donc $\log_2 \frac{1}{p} \simeq H(X)N$.

On voit donc que les messages longs (d'une taille fixée N) peuvent appartenir à deux classes :
Une classe de très grande probabilité cumulée (≈1) contenant $2^{H(X)N}$ messages (presque) tous de même probabilité $p = 2^{-H(X)N}$.

L'autre classe contenant les autres ($M^N - 2^{H(X)N}$) de probabilité cumulée négligeable.

On peut encore dire que l'on pourra traiter les séquences de longueur N (N>>1) comme s'il y en avait exactement $2^{H(X)N}$ et qu'elles étaient équiprobables.

Ex : pour la source binaire non équiprobable considérée dans le paragraphe précédent, on peut considérer qu'il y a $L=2^{469}$ suites de 1000 symboles binaires de même probabilité 1/L (les $2^{1000} - 2^{469}$ autres sont de probabilité cumulée négligeable)

NB : l'auteur recommande l'article déjà cité de Shannon qui explicite davantage les « presque », les « comme si » et les « négligeable »…

1.2. Information mutuelle et Capacité d'un canal

Si on considère une communication à temps discret : $X_k \rightarrow Y_k$ avec X_k v.a. à valeurs dans $\{x_i\}$ et Y_k v.a. à valeurs dans $\{y_j\}$, l'information mutuelle moyenne représente l'information apportée par l'observation de la sortie Y_k sur la valeur prise par la source X_k, elle est donnée par :

$$I(X,Y) = \sum_{i,j} P(x_i, y_j) \log_2 \frac{P(x_i / y_j)}{P(x_i)}$$

Si le canal est très bruité $P(x_i/y_j) = P(x_i)$ l'information mutuelle est nulle,
Si le canal est parfait (sans bruit) $P(x_i/y_j)=1$ pour i=j et 0 pour i≠j et l'information mutuelle est égale à l'entropie de la source.

On montre que : I(X,Y) = H(X) – H(X/Y) = H(Y) – H(Y/X)

La capacité d'un canal est définie comme le maximum de l'information mutuelle, ce maximum étant pris sur toutes les distributions de probabilité des symboles possibles.
La capacité est donc donnée par : $C = \max_{\{P(x_i)\}} I(X,Y) = \max_{\{P(x_i)\}} \left(H(X) - H(X/Y) \right)$.

1.3. Théorème de Shannon sur le codage canal

Shannon a démontré qu'il était possible de transmettre un message sur un canal de transmission avec une probabilité d'erreur arbitrairement faible si le débit (d'information)D à l'entrée du canal était inférieur à sa capacité.

1.3.1. Idée de démonstration (voir ref 2)

Considérons une source S_0 qui atteint le débit limite égal à la capacité C et considérons les séquences que l'on peut transmettre et recevoir de longueur N (N>>1).
D'après ce qui précède on sait que les séquences transmises peuvent être réparties en deux groupes : un groupe de forte probabilité de taille $2^{H(X)N}$ et le reste de probabilité négligeable.
De même les séquences reçues sont réparties en $2^{H(Y)N}$ d'entre elles et les autres.
Chaque séquence reçue appartenant au premier groupe peut avoir été produite par $2^{H(X/Y)N}$ entrées (les autres possibilités étant de probabilité cumulée négligeable).

Considérons maintenant une autre source produisant de l'information au débit D <C. C'est cette source que l'on cherche à transmettre sans erreur. Cette source peut générer 2^{DN} messages . On souhaite associer à ces messages des séquences transmises (parmi les $2^{H(X)N}$) de façon à obtenir une faible probabilité d'erreur (se tromper sur le message étant donnée la séquence reçue). Cette association constitue ce que l'on appellera le codage canal.
On va considérer toutes les associations possibles (en utilisant uniquement les groupes de grande probabilité) et moyenner les erreurs sur ce très grand ensemble de codeurs, ce qui est équivalent à opérer un **codage aléatoire** et à calculer la probabilité d'erreur.

Il y a 2^{DN} messages distribués aléatoirement sur $2^{H(X)N}$ entrées (on appellera **mots de code** ces entrées correspondant à des messages). La probabilité pour qu'une entrée soit un mot de code est donc $2^{(D-H(X))N}$.
Considérons une séquence reçue particulière y_1 et calculons la probabilité pour que aucune des séquences d'entrée pouvant avoir produit y_1 (autres que la vraie) ne corresponde à un mot de code (ou bien toutes les séquences autres que la vraie (il y en a $2^{H(X/Y)N} - 1$) ne sont pas des mots de code). Cette probabilité est $P = \left(1 - 2^{(D-H(X))N}\right)^{2^{H(X/Y)N}-1}$.

Nous savons que D<C, donc D-H(X) = - H(X/Y)-ε (avec ε>0). Par conséquent :

$P = \left(1 - 2^{-(H(X/Y)+\varepsilon)N}\right)^{2^{H(X/Y)N}-1}$ qui tend vers $1-2^{-\varepsilon N}$ lorsque N tend vers l'infini.

La probabilité d'erreur tend donc vers 0.

La démonstration s'appuie sur un codage aléatoire avec des séquences de longueur infinie. Nous verrons que les turbo-codes ont des caractéristiques proches de ces codes.

1.3.2. Capacité pour un canal à bande limitée et à bruit additif gaussien et performances d'un système de transmission sur un tel canal.

Pour un canal à bande limitée (de largeur monolatérale W) et à bruit additif gaussien, blanc dans la bande W, indépendant de X et de variance N_0W où $N_0/2$ est la densité spectrale de puissance bilatérale du bruit, on montre que la capacité du canal (exprimée en bits par seconde) est égale à :

$C = W \log_2\left(1 + \frac{P}{N_0 W}\right)$ où P est la puissance du signal X .

Supposons une source binaire (de symboles 0 et 1 équiprobables) délivrant un symbole toutes les T_b secondes. Le débit d'information de cette source sera donc $D=1/T_b$. D/W représente l'efficacité spectrale de la transmission (en bits par seconde et par Hertz)

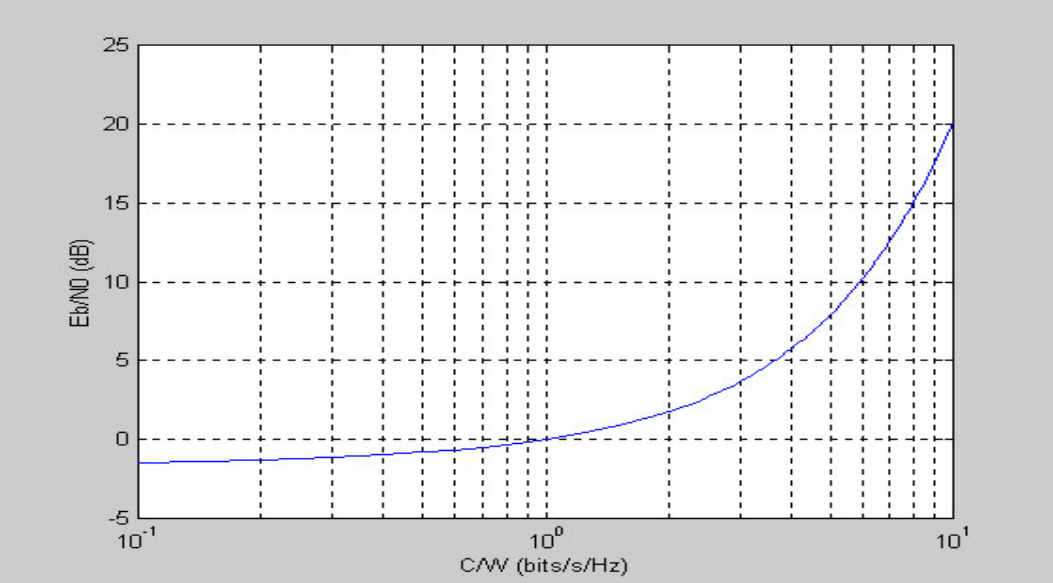
Shannon affirme que si D<C alors on peut (par codage de la source) obtenir une probabilité d'erreur aussi faible que l'on veut.
Réécrivons l'expression de la capacité du canal :

$$\frac{C}{W} = \log_2\left(1 + \frac{PT_b}{N_0}\frac{1}{WT_b}\right) = \log_2\left(1 + \frac{E_b}{N_0}\frac{D}{W}\right)$$

Si D=C, le rapport signal sur bruit E_b/N_0 permettant d'atteindre une probabilité d'erreur aussi petite que l'on veut est donné par :

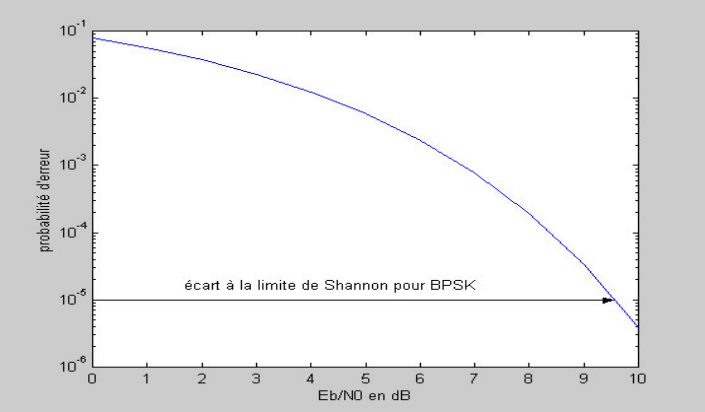
$$\frac{E_b}{N_0} = \frac{2^{C/W} - 1}{C/W}$$

Ce rapport signal sur bruit est appelé **limite de Shannon**. On l'a représenté sur la figure ci dessous en dB en fonction de l'efficacité spectrale limite C/W.



Lorsque C/W tend vers zéro, le rapport E_b/N_0 tend vers Ln(2)=-1.6dB . Ceci signifie qu'avec une largeur de bande infinie on peut transmettre avec une probabilité d'erreur arbitrairement faible un message avec un rapport signal sur bruit de –1.6 dB.

Pour la transmission d'un signal utilisant une modulation numérique BPSK (1 bit par symbole, $1/T_s$ bits/s) sur un canal de largeur monolatérale W on peut transmettre au débit maximal de $1/T_s$=W, on a donc une efficacité spectrale maximale de 1 bit/s/Hz . La courbe nous indique que pour un rapport E_b/N_0 de 0 dB on peut théoriquement transmettre cette information sans erreur.
Sans codage canal, avec une démodulation cohérente on parvient en fait à une probabilité d'erreur égale à $p = \frac{1}{2} erfc\left(\sqrt{\frac{E_b}{N_0}}\right)$ (figure ci dessous).



Si on se fixe l'objectif d'une probabilité d'erreur de 10^{-5} (valeur souvent prise comme référence) il faut un rapport E_b/N_0 de 9.5 dB . On dit que la modulation BPSK avec démodulation cohérente est à 9.5 dB de la limite de Shannon.

L'objectif du codage canal est de réduire cet écart et ce faisant d'obtenir un « gain de codage » qui représente le gain de RSB (par rapport à celui nécessaire sans codage) que l'on obtient pour transmettre avec une probabilité d'erreur fixée.

1.4. Notion de distance et gain de codage pour les codes classiques

Nous n'allons pas ici reprendre la théorie des codes en blocs ni celle des codes convolutifs (se reporter au cours de I4 ou à /ref4,/ref5 pour voir ces techniques) mais nous rappellerons simplement les principes de base de tels codes.
Dans la conception des codes en blocs et des codes convolutifs (techniques de codage classiques) on a cherché à diminuer l'incertitude que l'on a sur les messages d'information connaissant les séquences reçues en sortie du canal en maximisant la **distance minimale** entre les mots de code (pour les codes en bloc) ou les séquences codées (pour les codes convolutifs) émis(es). L'idée est alors de récupérer le message correspondant au mot de code le plus « proche » du mot reçu, ce qui suppose que les erreurs générées par le canal introduisent une distance entre le code reçu et le « vrai » code inférieure à la moitié de cette distance minimale .
La notion de distance étant plus facile appréhender dans le cas des codes en blocs nous nous limiterons dans la suite de ce paragraphe à ce type de code.

Pour obtenir ces mots de code (ou séquences codées) à partir de l'information à transmettre, on doit introduire une certaine redondance (qui sera l'équivalent du passage d'un espace à 2^{DN} messages dans un espace qui en contient $2^{H(X)N}$).
Pour les codes linéaires binaires en bloc C(n,k) on part de blocs d'information de k bits (il y a 2^k valeurs possibles) que l'on transforme par une opération linéaire (matrice génératrice ou polynôme générateur) en blocs de n bits (n>k) (2^n valeurs possibles) que l'on appelle mots de code.
On utilisera le plus souvent des codes ***systématiques*** pour lesquels les k premiers bits du mot de code seront les k bits d'information originaux. Les n-k bits suivants seront appelés bits de redondance ou ***bits de parité***. Le rendement d'un tel code est donc R=k/n. Les différents types de codes en bloc diffèrent par la technique utilisée pour construire la matrice génératrice. Ils sont conçus pour garantir une ***distance minimale*** d_{min} entre les mots de code (la distance considérée étant celle de Hamming qui donne le nombre de bits différents entre les mots) et donnent ainsi la possibilité de corriger un certain nombre t d'erreurs par mot reçu de n bits ($2t+1=d_{min}$) .
La probabilité d'erreur par élément binaire peut alors être majorée par :

$P_{eb} \le \frac{1}{n}\sum_{i=t+1}^{n}(i+t)\,C_n^i p^i (1-p)^{n-i}$ où p représente la probabilité d'erreur sur le canal de transmission.

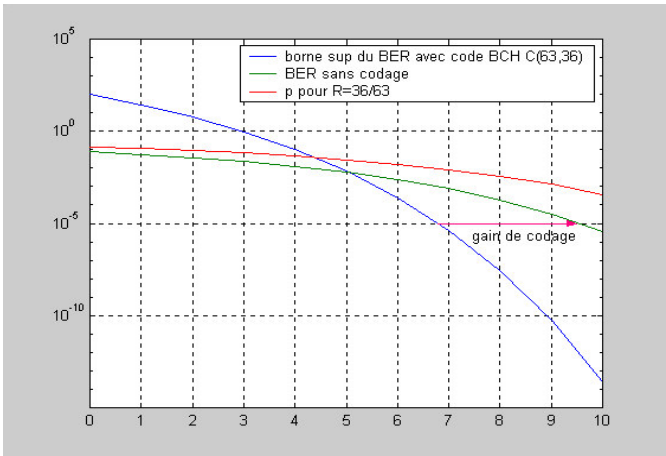
Cette borne est obtenue en voyant que :
Le mot sera mal décodé (et entraînera donc des erreurs) si $i \ge t+1$.
Dans le pire des cas le décodeur ajoutera t erreurs en essayant de corriger (d'où (i+t))
Les erreurs étant indépendantes, la probabilité d'avoir i erreurs est $C_n^i p^i (1-p)^{n-i}$
Lorsque p est suffisamment faible on a l'approximation suivante :

$$P_{eb} \approx \frac{1}{n}(2t+1)C_n^{t+1}p^{t+1}$$

La probabilité d'erreur p dépend du rapport signal sur bruit E_b/N_0. En l'absence de codage, tous les bits transmis sont des bits d'information et le rapport signal à bruit à l'entrée du récepteur est effectivement E_b/N_0 alors qu'en utilisant un code de rendement R=k/n et en n'augmentant pas la puissance P du signal émis , l'énergie reçue par bit transmis sera RE_b et le rapport signal sur bruit à prendre en compte pour calculer p est donc RE_b/N_0.
Par exemple pour un code BCH C(63,36) qui corrige 5 erreurs (par bloc de 63 bits)

$P_{eb} \approx \frac{11}{63}C_{63}^6 p^6$ et $p = \frac{1}{2}erfc\left(\sqrt{\frac{36E_b}{63N_0}}\right)$ (pour E_b/N_0 suffisamment grand pour que p<<1)

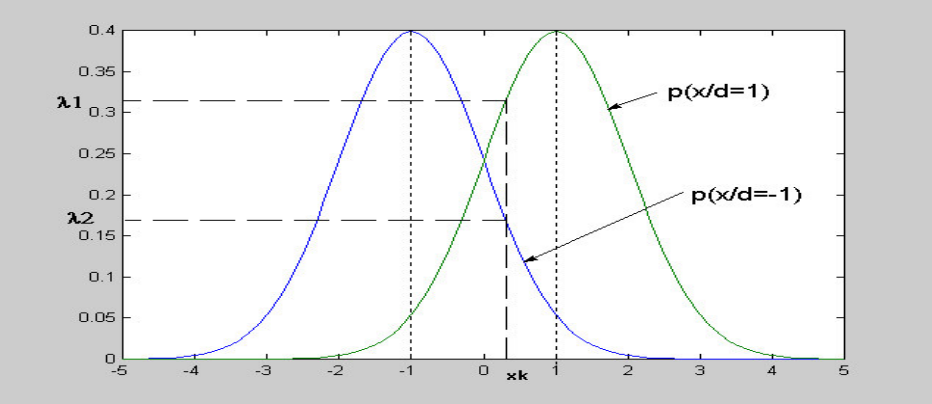
La figure ci-dessous montre l'évolution de P_{eb} en fonction de E_b/N_0 avec cette formule très approchée. Le gain de codage (pour $P_{eb}=10^{-5}$) vaut 2.8 dB. En revanche on voit que si le canal est très bruité (E_b/N_0 < 5 dB) il est préférable de ne pas coder l'information. Ceci se comprend car il devient assez probable d'avoir plus de 5 erreurs par bloc de 63 bits et que le décodage est alors sans efficacité (il rajoute même des erreurs).



2. Estimation MAP et calcul de Log-Likelihood Ratio – Principe du décodage itératif et information extrinsèque – étude d'un exemple « à la main »

2.1. Estimation MAP – LLR – décision souple

Supposons une transmission de symboles binaires 0 et 1 (représentés respectivement par les valeurs –1 et 1) sur un canal à bruit additif .
On recevra à chaque temps symbole le signal :
x = d + n où d=+/- 1 et n est la valeur prise par le bruit à cet instant.
Si le bruit est indépendant du signal et de distribution de probabilité connue on a accès aux fonctions de vraisemblance (ou probabilités à priori) p(x/d=1) et p(x/d=-1).
La figure suivante montre ces fonctions pour un bruit gaussien de variance unité.



La règle de décision dure la plus souvent utilisée, appelée maximum de vraisemblance consiste à prendre la décision qui maximise cette fonction de vraisemblance. Pour un signal reçu x, la règle est la suivante :

$$p(x/d=1) \underset{-1}{\overset{1}{>}} p(x/d=-1)$$

Pour un bruit gaussien ceci revient à détecter le signe de x :si x >0 on décide 1, sinon –1.

Pour les turbo-codes on utilisera la règle du maximum à posteriori (MAP) (ou règle du minimum d'erreur), qui prend en compte les probabilités à priori des symboles p(d=1) et p(d=-1) :

$$P(d=1/x) \underset{-1}{\overset{1}{\gtrless}} P(d=-1/x)$$

En utilisant le théorème de Bayes, on peut encore écrire cette règle sous la forme

$$\frac{p(x/d=1)P(d=1)}{p(x/d=-1)P(d=-1)} \underset{-1}{\overset{1}{\gtrless}} 1$$ appelée test du rapport de vraisemblance (Likelihood Ratio).

En prenant le logarithme de ce rapport de vraisemblance on obtient une mesure appelée log-likelihood ratio (LLR) ou encore log a posteriori probability ratio (LAPP). Ce nombre représentera la ***décision souple*** en sortie de décodeur, notée *L(d/x).* La ***décision dure*** finale associée sera le signe de cette valeur.

Ecrivons ce LAPP :

$$L(d/x) = \log\left(\frac{P(d=1/x)}{P(d=-1/x)}\right)$$

$$= \log\left(\frac{p(x/d=1)}{p(x/d=-1)}\right) + \log\left(\frac{P(d=1)}{P(d=-1)}\right)$$

$$= L(x/d) + L(d)$$

L(d/x) est le LLR qui servira à prendre la décision finale $\hat{d} = sign(L(d/x))$ on le notera pour cette raison (et en accord avec la référence (ref 3)) $L'(\hat{d})$

L(x/d) est le LLR qui représente la vraisemblance de la valeur mesurée x à la sortie du canal sous les deux conditions possibles d=1 et d=-1 . Cette valeur est entièrement dépendante du canal de transmission et sera notée $L_c(x)$.

Pour un canal à bruit additif blanc gaussien de variance σ^2 :

$$L_c(x) = \log\left(\frac{\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-1)^2}{2\sigma^2}\right)}{\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x+1)^2}{2\sigma^2}\right)}\right) = \frac{2x}{\sigma^2}$$

L(d) est le LLR a priori du symbole (bit) d.

En l'absence de codage et décodage canal on peut le plus souvent considérer que L(d)=0 (symboles équiprobables) et on se ramène donc au critère du maximum de vraisemblance.

L'introduction d'un processus de codage canal va lier l'information d aux autres bits du même bloc ou de la même séquence et va permettre (au moment du décodage) d'acquérir une nouvelle connaissance sur la vraisemblance de d due à l'exploitation de ce lien (ou

redondance ou parité) c'est à dire à partir des autres bits . On appelera ***information extrinsèque*** ce gain d'information .

On aura alors $L(\hat{d}) = L'(\hat{d}) + L_e(\hat{d}) = L_c(x) + L(d) + L_e(\hat{d})$ où $L_e(\hat{d})$ est le LLR extrinsèque dû au décodage. Ce gain d'information sera ré-injecté comme le nouveau $L(d)$ en entrée du décodeur suivant.

$L(\hat{d})$ (qui mesure la confiance que l'on a sur la valeur de d) va ainsi pouvoir devenir de plus en plus fiable (clairement positif ou négatif) lors d'un processus de décodage itératif à l'issue duquel on prendra la décision sign($L(\hat{d})$).

Ce processus ne peut fonctionner que s'il y a au moins deux (dé)codeurs et un entrelaceur (qui agit de sorte que la prochaine parité soit différente de celle qu'on vient d'exploiter).

2.2. Principe du décodage itératif pour un code produit élémentaire.

On considère le code produit suivant :

d_1	d_2	p_{12}
d_3	d_4	p_{34}
p_{13}	p_{24}	

Avec les relations suivantes :

$d_i \oplus d_j = p_{ij}$ $\quad d_i \oplus p_{ij} = d_j$ $\quad d_j \oplus p_{ij} = d_i$

$\oplus$ désigne l'addition définie par la table suivante:

d_1	d_2	$d_1 \oplus d_2$
1	1	-1
1	-1	1
-1	1	1
-1	-1	-1

Supposons que la séquence transmise $d_1 d_2\ d_3\ d_4\ p_{12}\ p_{34}\ p_{13}\ p_{24}$ soit perturbée par un canal additif blanc gaussien (de variance unité) de telle sorte que le récepteur reçoive la séquence $x_1\ x_2\ x_3\ x_4\ x_{12}\ x_{34}\ x_{13}\ x_{24}$.

Considérons l'exemple suivant : $d_1\ d_2\ d_3\ d_4 = 1\ -1\ -1\ 1$
La séquence transmise est 1 –1 –1 1 1 1 1 1.
Supposons que la séquence reçue soit : 0.75, 0.05, 0.10, 0.15, 1.25, 1.0, 3.0, 0.5.
Ce qui donne le tableau suivant des LLR $L_c(x)=2x$:

1.5	0.1	2.5
0.2	0.3	2.0
6.0	1.0	

Si le récepteur prenait les décisions directement sur $L_c(x)$ il ferait deux erreurs de décision (sur d_2 et d_3). S'il étudie les bits de parité il détecte une erreur sur chaque ligne et chaque colonne, ce qui indique une configuration d'erreur croisée (bits 2 et 3 faux ou bits 1 et 4 faux) en considérant que les décisions prises sur les bits de parité sont justes.
Nous allons montrer sur cet exemple comment un décodage itératif des lignes et des colonnes avec le passage de l'information extrinsèque d'une étape à l'autre permet de lever le doute et corriger ces erreurs.

Examinons en premier lieu ce que peut être le LLR extrinsèque $L_{eH}\left(\hat{d}_1\right)$ fourni par le premier décodage horizontal .

L'information sur d_1 lors de ce premier passage est fournie par :
L'information due au signal reçu x_1 : $L_c(x_1) = 2x_1$
L'information a priori sur d_1 : $L(d_1)=0$ car les bits sont a priori équiprobables
L'information due au fait que normalement (par codage) $d_1 = d_2 \oplus p_{12}$. Cette information peut se noter $L(d_2 \oplus p_{12}) = L(d_2) + L(p_{12})$. Cette information est l'information extrinsèque notée $L_{eH}\left(\hat{d}_1\right)$

On peut aisément montrer que pour deux variables binaires (1,-1) statistiquement indépendantes d_1 et d_2

$$L(d_1) + L(d_2) = L(d_1 \oplus d_2) = \log_e\left(\frac{e^{L(d_1)} + e^{L(d_2)}}{1 + e^{L(d_1)}e^{L(d_2)}}\right)$$

$$\approx (-1)\, sign\left(L(d_1)\right) sign\left(L(d_2)\right) \min\left(\left|L(d_1)\right|, \left|L(d_2)\right|\right)$$

On a donc $L_{eH}\left(\hat{d}_1\right) = (L_c(x_2)+L(d_2)) + L_c(x_{12})$ ce qui à la première étape donne :
$(0.1+0) + 2.5 = -0.1$

De même on aura les informations extrinsèques pour les autres décodages lignes (H) :

$L_{eH}\left(\hat{d}_2\right) = (L_c(x_1)+L(d_1)) + L_c(x_{12})$ (première étape : $(1.5+0) + 2.5 = -1.5$)

$L_{eH}\left(\hat{d}_3\right) = (L_c(x_4)+L(d_4)) + L_c(x_{34})$ (première étape : $(0.3+0) + 2.0 = -0.3$)

$L_{eH}\left(\hat{d}_4\right) = (L_c(x_3)+L(d_3)) + L_c(x_{34})$ (première étape : $(0.2+0) + 2.0 = -0.2$)

Ces valeurs vont être injectées dans le décodeur des colonnes (V) comme nouvelles valeurs $L(d_i)$. Ce décodeur va calculer de nouvelles informations extrinsèques :

$L_{eV}\left(\hat{d}_1\right) = (L_c(x_3)+L(d_3)) + L_c(x_{13})$ (première étape : $(0.2-0.3) + 6.0 = 0.1$)

$L_{eV}\left(\hat{d}_2\right) = (L_c(x_4)+L(d_4)) + L_c(x_{24})$ (première étape : $(0.3-0.2) + 1.0 = -0.1$)

$L_{eV}\left(\hat{d}_3\right) = (L_c(x_1)+L(d_1)) + L_c(x_{13})$ (première étape : $(1.5-0.1) + 6.0 = -1.4$)

$L_{eV}\left(\hat{d}_4\right) = (L_c(x_2)+L(d_2)) + L_c(x_{24})$ (première étape : $(0.1-1.5) + 1.0 = 1.0$)

Ces nouvelles valeurs seront fournies au décodeur lignes (H) comme nouvelles LLR $L(d_i)$…

Le procédé de décodage est donc le suivant :

Décodage itératif :

calculer les informations extrinsèques provenant du décodage des lignes : $L_{eH}\left(\hat{d}_i\right)$ i=1...4

considérer ces valeurs comme les nouvelles LLR $L(d_i)$ pour le décodage des colonnes.

calculer les informations extrinsèques provenant du décodage des colonnes : $L_{eV}\left(\hat{d}_i\right)$

considérer ces valeurs comme les nouvelles LLR $L(d_i)$ pour le décodage des lignes.

Les nouvelles variables de décision sont : $L\left(\hat{d}_i\right) = L_c\left(x_i\right) + L_{eH}\left(\hat{d}_i\right) + L_{eV}\left(\hat{d}_i\right)$

…

Mise en œuvre sur l'exemple :

Les tableaux suivants donnent l'évolution des $L\left(\hat{d}_i\right)$ après chaque étape de décodage ainsi que celle des vraisemblances extrinsèques transmises au décodeur suivant:

$L\left(\hat{d}_i\right)$	1.5	0.1
$=L_c(x_i)$	0.2	0.3

$L_{eh}(d_i)$	-0.1	-1.5	$L_{ev}(d_i)$	0.1	-0.1
	-0.3	-0.2		-1.4	1.0
$L\left(\hat{d}_i\right)$	1.4	-1.4	$L\left(\hat{d}_i\right)$	1.5	-1.5
	-0.1	0.1		-1.5	1.1

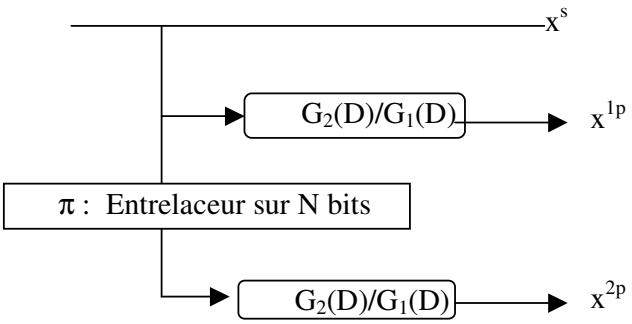
$L_{eh}(d_i)$	0.0	-1.6	$L_{ev}(d_i)$	1.1	-1.0
	-1.3	1.2		-1.5	1.0
$L\left(\hat{d}_i\right)$	1.5	-1.5	$L\left(\hat{d}_i\right)$	2.6	-2.5
	-1.1	1.5		-2.6	2.5

A l'issue de la première étape de décodage, la décision sign($L\left(\hat{d}_i\right)$) donne déjà un résultat sans erreur, les étapes suivantes ne font que conforter cette décision.

.

3. Encodeur (classique) d'un turbo-code – rôle de l'entrelaceur.

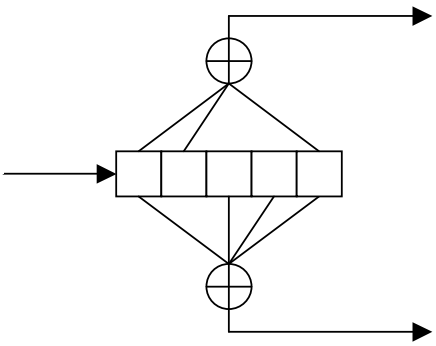
La figure suivante représente un encodeur turbo standard.



Il est constitué de deux encodeurs convolutionnels récursifs systématiques (RSC) séparés par un entrelaceur (ou permutateur) sur N bits.
Le rendement 1/3 d'un tel schéma peut être modifié par l'ajout d'un dispositif de poinçonnage sur les bits de parités x^{1p} et x^{2p} des encodeurs. Nous ne nous intéresserons pas à cette possibilité par la suite et nous supposerons (sans perte essentielle de généralité) que les deux encodeurs sont identiques.
Nous allons d'abord rappeler les bases sur les codeurs convolutionnels et les propriétés intéressantes des codes RSC puis nous étudierons le rôle de l'entrelaceur.

3.1. Encodeur convolutionnel (ou convolutif)

Nous nous placerons dans le cas (très utilisé en pratique) d'un encodeur de rendement ½ qui génère 2 bits de sortie à chaque bit d'entrée.
Ces 2 bits sont à l'instant n obtenus par 2 combinaisons des (m+1) bits d'entrée actuel u_n et précédents u_{n-i} (i=1…m).L'encodeur introduit donc un effet mémoire d'ordre m et (m+1) s'appelle la longueur de contrainte du code.
Un exemple est donné sur la figure suivante :



Le terme convolutif vient de ce que l'on peut écrire chacune des sorties comme produit de convolution entre l'entrée u_n et la séquence génératrice du codeur correspondant.

Ainsi la sortie du ième codeur (i=1,2) s'écrit : $c_n^i = \sum_{j=0}^{m} g_{ij} u_{n-j}$ avec $g_{ij} \in \{0,1\}$

Pour le codeur représenté sur la figure précédente les séquences génératrices sont :
g_1=[1 1 0 0 1] et g_2=[1 0 1 1 1].

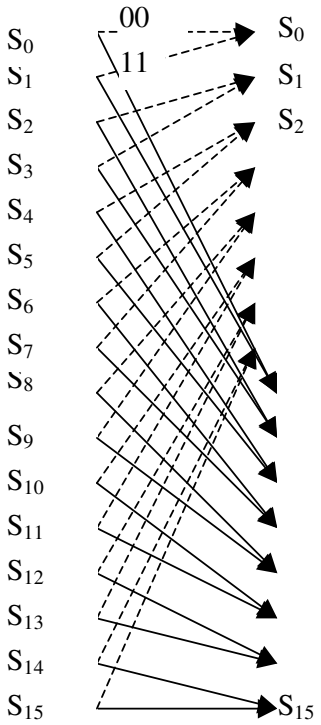
En général on spécifie ces séquences en octal ($g_1=31_{octal}$ et $g_2=27_{octal}$ dans l'exemple ci-dessus).

On définit encore les codes par leur polynômes générateurs : $g_i(D)=\sum_{j=0}^{m} g_{ij}D^j$ qui sont les constituants de la matrice génératrice G(D)= [g_1(D) g_2(D)].

Si on décrit une séquence d'entrée u_n par son polynôme associé $U(D)=\sum_n u_n D^n$, la séquence codée est décrite par C(D) = U(D) G(D) = [U(D)g_1(D) U(D)g_2(D)].
Cette description se rapproche de celle des codes en blocs et sera à nouveau utilisée pour aborder les codes récursifs systématiques mais elle ne permet pas d'aborder facilement le décodage (contrairement au cas des codes en blocs pour lesquels l'exploitation de la connaissance des racines du polynôme générateur permet de détecter et corriger les erreurs). Nous allons donner à présent une représentation plus efficace des codes convolutionnels qui donne accès à la notion de distance pour ces codes et à la technique classique de décodage. Il s'agit de la représentation treillis.

3.2. représentation treillis et principe du décodage

Comme nous l'avons vu précédemment, les sorties c_n^i du codeur à l'instant n dépendent de l'entrée u_n et de la mémoire du codeur [u_{n-1} u_{n-2} …u_{n-m}]. Cette mémoire constitue ce que l'on nomme l'état s(n) du codeur à l'instant n. Il y a 2^m états pour un codeur de longueur de contrainte m+1. A partir d'un état on pourra atteindre à l'instant suivant deux états suivant la valeur 0 ou 1 du bit d'entrée. On peut représenter l'évolution du codeur par n treillis dont les nœuds représentent les états par les nœuds du treillis et l'ensemble des transitions possibles par des branches ainsi que la valeur des bits codés (c_n^1, c_n^2) correspondante. Par convention on représente par un trait plein les branches (transitions) correspondant à u_n=1 et par un trait en pointillé celles pour lesquelles u_n=0.
Pour l'exemple précédent ($g_1=31_{octal}$, $g_2=27_{octal}$) le treillis a 2^4=16 états : S_0=0000,…,S_{15}=1111 et est représenté sur la figure suivante :



Nous n'avons reporté les valeurs de sortie (c_n^1, c_n^2) que sur les premières branches pour une meilleure lisibilité.

Ce treillis indique que lors du codage seules certaines séquences d'états (certains chemins dans le treillis) sont possibles (par exemple $S_{12}S_{15}S_1$ n'est pas possible) et donc seules les séquences de sortie associées à ces chemins correspondent à des séquences d'entrée codées. Le principe du décodage sera par conséquent de rechercher la séquence de sortie (permise) la plus vraisemblable compte tenu de la séquence reçue. En pratique on utilisera l'algorithme de Viterbi qui est basé sur le calcul itératif des distances minimales entre la séquence reçue et les séquences codées possibles.

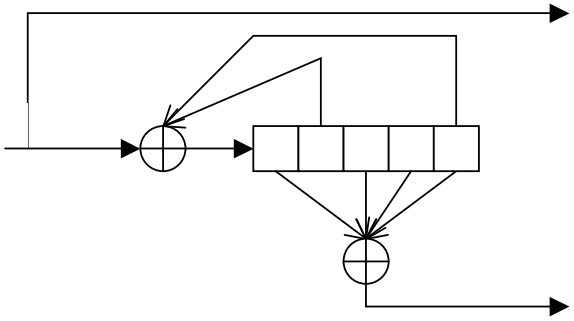
3.3. Distance libre

La notion de distance pour les codes convolutifs est semblable à celle des codes en blocs. Elle est liée à la distance de hamming entre séquences codées. Dans le cas convolutif ces séquences sont de longueur variable, deux séquences peuvent diverger sur quelques instants puis converger puis diverger à nouveau. On définit donc la **distance libre** d'un code convolutif comme la plus petite distance de hamming existant entre deux chemins (suites d'états) qui divergent puis convergent de nouveau. Comme un code convolutif est linéaire, la distance libre est encore égale au poids du chemin de plus faible poids divergeant du chemin « tous zeros » et re-convergeant vers ce chemin de poids nul.

3.4. Encodeur récursif systématique (RSC)

Les encodeurs figurant dans le schéma du turbo-encodeur classique sont des encodeurs récursifs systématiques.
Si on considère un encodeur convolutif de rendement ½ caractérisé par sa matrice génératrice $G_{NR}(D)= [g_1(D)\ g_2(D)]$, on peut définir l'encodeur récursif systématique équivalent par la matrice $G_R(D)= [1\ g_2(D)/g_1(D)]$.
En effet ces deux encodeurs ont le même treillis d'états et produisent les mêmes mots de code:
le code Non Récursif génère les mots $C(D)=U(D)G_{NR}(D) = [U(D)g_1(D)\ U(D)g_2(D)]$.
le code Récursif génère les mots $C'(D)=U'(D)G_R(D) = [U'(D)\ U'(D)g_2(D)/g_1(D)]$.
Si on considère le mot d'entrée $U'(D)=U(D)g_1(D)$ pour le codeur Récursif on obtient C(D) en sortie de ce codeur.

Pour l'exemple $(g_1,g_2)=(31,27)$ le schéma du codeur récursif systématique équivalent est représenté ci-dessous :



Le treillis est identique car la mémoire du codeur qui définit l'état évolue de la même façon que précédemment

Remarque importante :
D'après ce qui précède, une séquence codée par le codeur récursif C'(D) ne sera ne poids fini que si la séquence d'entrée U'(D) correspondante est divisible par $g_1(D)$.
Cette remarque entraîne les conséquences suivantes utiles pour comprendre les performances du turbo-encodeur :
un mot d'entrée de poids 1 produit un mot de code de poids infini car D^i n'est pas divisible par $g_1(D)$ (pour $g_1(D)$ non trivial).
pour $g_1(D)$ non trivial, il existe une famille d'entrées de poids 2 de la forme $D^i(1+D^{q-1})$ qui produisent des mots de code de poids fini (c'est à dire divisibles par $g_1(D)$). Rappelons pour cela les propriétés suivantes des polynômes :
si $g_1(D)$ est un polynôme primitif de degré m ($g_1(D) = (D-\alpha)(D-\alpha^2)...(D-\alpha^{2^{m-1}})$)

alors $g_1(D)$ divise $(1+D^{2^m-1})$. Dans ce cas $q-1 = 2^m - 1$.
de façon plus générale q-1 est la longueur de la séquence générée par $g_1(D)$.

Si on considère la représentation treillis des codeurs, le premier point signifie qu'une entrée U(D) de poids 1 crée un chemin qui ne rejoint jamais le chemin tous zéros, le deuxième dit qu'il existe des entrées de poids 2 correspondant à des chemins qui divergent du chemin tous zéros et re-convergent ensuite vers lui.

3.5. L'entrelaceur (ou permuteur)

Comme son nom l'indique, l'entrelaceur a la tâche de réarranger l'ordre des bis d'entrée avant le second codage. Il transforme chaque bloc de N bits d'entrée $(b_1,\ldots,b_N)$ en $(b_{\pi(1)}\ldots b_{\pi(N)})$. Dans les techniques de codage conventionnelles cet entrelaceur est généralement fixe et réarrange les bits de façon systématique. Pour les turbos-codes les meilleurs résultats seront obtenus pour un entrelaceur (pseudo)aléatoire. La taille N de cet entrelaceur est un paramètre important pour les performances, elle doit être suffisamment grande (N>1000 est généralement admis).

3.6. Performances du turbo-encodeur

Nous allons estimer les performances d'un décodeur qui décode des séquences (de N bits) au sens du maximum de vraisemblance (Maximum Likelihood). Nous verrons dans le paragraphe suivant qu'il n'est pas envisageable de mettre en œuvre un tel décodeur (ML) en raison de la trop grande complexité due à la présence de l'entrelaceur. Cependant nous décrirons un algorithme sous-optimal de décodage qui offrent des performances voisines de celles du décodeur optimal.
L'encodeur étant linéaire (les deux codeurs convolutifs le sont et l'entrelaceur l'est aussi), la différence entre deux séquences (une possible et une observée) est égale à la différence entre la séquence d'erreur et la séquence « tous zéros ; nous pouvons donc, pour évaluer les performances, nous ramener à prendre comme référence (séquence émise) cette séquence qui représente le 0 $^{\text{ième}}$ mot de code C_0 (qui correspond au mot d'entrée constitué de N zéros). Les autres séquences possibles sont les 2^N-1 mots de code C_k différents de C_0.

Le décodeur ML choisira C_k au lieu de C_0 avec la probabilité $Q\left(\sqrt{\frac{2d_k rE_b}{N_0}}\right)$ où r est le rendement (ici 1/3 s'il n'y a pas de poinçonnage) et d_k est le poids de C_k.

Le taux d'erreur bit (nb de bits erronés / nb de bits émis) est pour ce cas :

$$P_b(k/0) = Q\left(\sqrt{\frac{2d_k rE_b}{N_0}}\right) \text{ (mot de code faux/mot de code)}$$

$$\frac{1}{N} \text{ (mot de code/bit d'entrée)}$$

$$w_k \text{ (bits erronés/mot de code faux)}$$

où w_k est le poids du mot d'entrée U_k correspondant au mot de code C_k.

Si nous considérons maintenant l'ensemble des mots de codes C_k possibles, la probabilité d'erreur est majorée par la borne de l'union : $P_b \le \sum_{k=1}^{2^N-1} P_b(k/0)$.

Soit $P_b \le \sum_{k=1}^{2^N-1} \frac{w_k}{N} Q\left(\sqrt{\frac{2rd_k E_b}{N_0}}\right) = \sum_{w=1}^{N} \sum_{v=1}^{C_N^w} \frac{w}{N} Q\left(\sqrt{\frac{2rd_{wv} E_b}{N_0}}\right)$

Où d_{wv} est le poids du mot de code produit par le v^ième mot d'entrée de poids w.

Examinons les premiers termes dans la somme sur w :
w=1 : d'après l'explication qui précède les mots D_k de poids 1 produisent des chemins qui ne reconvergent pas vers le chemin C_0. Les mots produits sont donc de poids d_{1v} très grand, donc très improbablement sélectionnés par le décodeur et leur contribution à P_b est négligeable.
w=2 : sur les C_N^2 mots d'entrée de poids 2, seule une petite fraction va être divisible par $g_1(D)$ et parmi ceux-ci seulement certains produiront des mots de distance (poids) minimale $d_{2\min}^{CC} = \min_v d_{2v}^{CC}$ à la sortie du premier codeur (Codeur Constituant). Si un mot U_2 produit un tel mot il est très improbable que $\pi(U_2)$ soit lui même un mot divisible par $g_1(D)$. La distance (poids) minimale des mots produits sera minorée par : $d_{2\min}^{TC} \ge 2d_{2\min}^{CC} - 2$ (-2 car la partie systématique (poids 2) n'est générée que par le premier codeur). Si on note par n_2 le nombre de mots d'entrée de poids 2 qui produisent des mots de poids $d_{2\min}^{TC}$, le terme en w=2 de la borne de P_b est approximé par : $\sum_{v=1}^{C_N^2} \frac{2}{N} Q\left(\sqrt{\frac{2rd_{2v}E_b}{N_0}}\right) \simeq \frac{2n_2}{N} Q\left(\sqrt{\frac{2rd_{2\min}^{TC}E_b}{N_0}}\right)$.
w=3 : le même raisonnement que précédemment conduit à :
$\sum_{v=1}^{C_N^3} \frac{3}{N} Q\left(\sqrt{\frac{2rd_{3v}E_b}{N_0}}\right) \simeq \frac{3n_3}{N} Q\left(\sqrt{\frac{2rd_{3\min}^{TC}E_b}{N_0}}\right)$. Bien que C_N^3 soit (N-2)/3 fois plus grand que C_N^2 le nombre de mots de poids 3 divisible par $g_1(D)$ est du même ordre de grandeur que celui des mots de poids 2 divisible par $g_1(D)$. De plus, étant donné un tel mot U_3, la probabilité pour que $\pi(U_3)$ soit encore divisible par $g_1(D)$ est très faible : considérons en effet $U_3(D)= g_1(D)=1+D+D^4$, $\pi(U_3)$ sera divisible par $g_1(D)$si les trois « 1 » de U_3 se retrouvent en position i, i+1, i+4. Si l'entrelaceur est aléatoire il y a une probabilité 3! / N^3 pour que ceci se réalise

(pour w=2 un motif donné en sortie de l'entrelaceur se produisait avec la probabilité 2! / N^2) . Ceci signifie que n_3 sera très petit devant n_2.
w≥4 : même type de raisonnement : si il existe n_2 ou n_3 non nuls le terme en n_4 sera négligeable.

Pour résumer la probabilité d'erreur bit peut être approximée par le terme dominant de la somme :

$$P_b \simeq \max_{w\geq 2} \frac{w n_w}{N} Q\left(\sqrt{\frac{2 r d^{TC}_{w\min} E_b}{N_0}}\right).$$

Il est possible de concevoir des entrelaceurs qui pour $g_1(D)$ donné ne donnent pas de mots de poids 2 divisibles par $g_1(D)$.
On voit que la probabilité d'erreur décroît avec N d'où l'intérêt d'un entrelaceur long.

3.7. Résumé sur l'encodeur d'un turbo-code classique

Par l'utilisation d'un code récursif systématique d'une part et l'emploi d'un entrelaceur « pseudo-aléatoire » d'autre part, l'encodeur d'un turbo code se comporte comme un codage aléatoire qui maximise la distance moyenne du code. La distance minimale n'est pas forcément très faible mais elle est très rarement atteinte. Rappelons que l'objectif des codeurs classiques est de maximiser la distance minimale du code (et généralement cette distance minimale est très souvent atteinte).

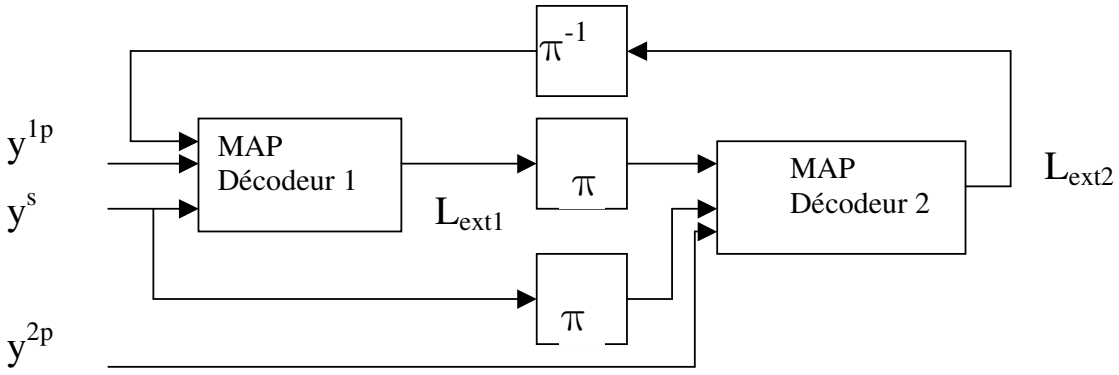
4. décodeur MAP

Le codeur présenté au paragraphe précédent (voir schéma du paragraphe 4) va donc encoder des blocs de N bits ($\mathbf{x^s=u}$) et délivrer 3N bits en sortie que l'on supposera codés en BPSK (valeurs –1 et 1) : (x^s_1, x^{p1}_1, x^{p2}_1, x^s_2, x^{p1}_2, x^{p2}_2,… ,x^s_N, x^{p1}_N, x^{p2}_N)
Le canal de transmission va dégrader le signal émis de sorte que le signal reçu (synchronisé et ré-échantillonné au rythme symbole (ici rythme bit) sera constitué d'une suite de 3N valeurs constituant la séquence reçue **y.**
Chaque décodeur va récupérer une séquence 2N valeurs :
le premier va récupérer les N valeurs de **y** correspondant aux bits systématiques(x^s) et les N valeurs correspondant aux bits de parité du premier encodeur (x^{1p}). On appellera $\mathbf{y_1}$ cette séquence
Le second va récupérer les N valeurs systématiques repassées dans l'entrelaceur (correspondant à $\pi(x^s)$) et les N valeurs correspondant aux bits de parité du deuxième encodeur (x^{2p}) . On appellera $\mathbf{y_2}$ cette séquence .
On rappelle (voir paragraphe 3.1) que la règle de décodage sera celle du MAP (maximum a posteriori). Chaque décodeur (i=1ou 2) doit donc fournir pour chaque symbole u_k (k=1 :N) du bloc original de N bits la valeur de Log vraisemblance à posteriori $L(u_k/\mathbf{y_i})$.

$$L(u_k / \mathrm{y_i}) = Ln\left(\frac{P(u_k = +1/\mathrm{y_i})}{P(u_k = -1/\mathrm{y_i})}\right)$$

Cette vraisemblance comportera une composante qui dépendra uniquement de bits de parité de y_i . Cette composante sera l'information extrinsèque $L_{ext,i}$ qui sera ré-injectée dans le décodeur suivant.

Le schéma du décodeur est donc le suivant



Le détail de l'algorithme MAP sera exposé en cours et peut être obtenu dans plusieurs références bibliographiques (par exemple dans la référence 4)

5. Bibliographie

Ref 1 : **C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error correcting coding and decoding: Turbo-codes," in Proc IEEE Intl Conf Cornrnun, (Geneva), pp. 1064-70, May 1993**

Ref 2 : Claude Shannon's ``A mathematical theory of communication" July and October 1948 editions of the *Bell System Technical Journal*

Ref 3: Bernard Sklar, "A primer on turbo code concepts," *IEEE Communications Magazine*, vol. 35, no. 12, pp. 94-102, Dec. 1997.

Ref 4 : W. Ryan, "A turbo code tutorial » : http://www.ece.arizona.edu/~ryan/turbo2c.pdf

Par ailleurs sur le site http://www.ee.vt.edu/~yufei/turbo.html vous trouverez les routines de base en Matlab pour programmer un turbo code classique

Introduction aux Turbo-codes 1
1. Petits rappels en théorie de l'information et approche classique du codage 3
1.1. Entropie d'une source 3
1.1.1. Définition 3
1.1.2. Probabilité d'une longue suite de symboles (approche intuitive) 3
1.2. Information mutuelle et Capacité d'un canal 3
1.3. Théorème de Shannon sur le codage canal 4
1.3.1. Idée de démonstration (voir ref 2) 4
1.3.2. Capacité pour un canal à bande limitée et à bruit additif gaussien et performances d'un système de transmission sur un tel canal. 5
1.4. Notion de distance et gain de codage pour les codes classiques 6
2. Estimation MAP et calcul de Log-Likelihood Ratio – Principe du décodage itératif et information extrinsèque – étude d'un exemple « à la main » 8
2.1. Estimation MAP – LLR – décision souple 8
2.2. Principe du décodage itératif pour un code produit élémentaire. 10
3. Encodeur (classique) d'un turbo-code – rôle de l'entrelaceur. 13
3.1. Encodeur convolutionnel (ou convolutif) 13
3.2. représentation treillis et principe du décodage 14
3.3. Distance libre 15
3.4. Encodeur récursif systématique (RSC) 15
3.5. L'entrelaceur (ou permuteur) 16
3.6. Performances du turbo-encodeur 16
3.7. Résumé sur l'encodeur d'un turbo-code classique 18
4. décodeur MAP 18
5. Bibliographie 19