République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique
Département d'Electronique

Thèse

Présentée par
Maali Abdelmadjid

Pour l'obtention du titre de

Magister en Signaux et Systèmes

Option: **Image et Parole**

Thème

Les Turbo Codes et Le Décodage Itératif

Date de soutenance : 18 avril 2001

Composition du jury :

Président	:	F. Boudjema	Professeur à l'E.N.P
Directeurs de thèse	:	D. Berkani	Professeur à l'E.N.P
		A. Belouchrani	Docteur à l'E.N.P
Examinateurs	:	R. Aksas	Maître de conférences à l'E.N.P
		Z. Terra	Chargé de cours à l'E.N.P
		N. Achab	Chargé de cours à l'E.M.P

ملخص:

الرموز العنفية الأولى اقترحت بواسطة الرموز الالتفافية، المنهجية، التراجعية و المتوازية. هاته الرموز المستعملة لترميز القناة، كانت قادرة على الوصـــول إلى احتمال صغير للوقوع في الخطأ على البيت ب 10^{-5} و ذلك بحاصل SNR يعادل 0.7 ديسبال فقط . الرموز العنفية يمكن اســـتعمالها في كل أنظمة الاتصالات التي تفرض طاقة قليلة. الاتصالات في الفضاء، الاتصالات المتنقلة بواسطة القمر الصناعي الخلوي، الروابط ميكروموجات، إلخ ...، هي بعض الأمثلة للتطبيقات لهاته التقنية التشفير، هذا العمل يوصف قدرات التسلسل بالتوازي و بالتتابع لرامزتين التفـــافيتين منـــهجيتين تراجعيتين لنسب SNR (E_b/N_0 dB) ضعيفة جدا وذلك باستعمال حلال الرموز المكرر مع خوارزمية MAP (الحد الأقصى الاستدلالي) .باستعمال مجال اللوغاريتم المتشابه، نبين أن كل رامزة، تستعمل قيم مسبقة و تحرر القيمة المرمزة، القيمة المسبقة، و القيمة الظاهريـــة. القيمـــة الظاهريـــة تستعمل كقيمة مسبقة للإعادة المقبلة. النتائج تبين أن التبادلات تساعد الرامزات لتحسين قدراتهن على تصحيح الأخطاء محتفظة بالمعلومـــات الظاهرية و المعطيات المستلمة بدون علاقة بينهما، كما تبين أيضا تأثير مختلف المتحولات على مر دودية الرموز العنفية. النتائج المحصل عليها جد مرضية.

كلمات مفتاحيه: الرموز العنفية، الرموز المتسلسلة، حل الرموز بالتكرار، الرموز المصححة للأخطاء.

Abstract :

The first Turbo Codes have been introduced by parallel, recursive and systematic convolutional codes. These codes used in channel coding were capable to achieve an arbitrarily small Bit Error Rate (BER) of 10^{-5} with an SNR of just 0.7 decibels. Turbo codes can be used in any communication systems that require the minimum of energy. Deep space communications, mobile satellite/cellular communications, microwave links, etc.., are some of the possible applications of this new coding technique. This work describes the performance of the parallel and serial concatenation of two recursive systematic convolutional codes at very low SNR (E_b/N_0 dB) iteratively decoded with the MAP (maximum a posteriori) algorithm. Using log-likelihood domain, we show that any decoder can uses inputs a priori values and delivers decoded output, a priori inputs and the extrinsic value. The extrinsic value is used as an a priori value for the next iteration. Results of simulation show how interleavers helps the decoders to improve their capability of correction by keeping the extrinsic information with the received data uncorrelated, and show also the different parameters that influence turbo codes performances. The obtained results are very satisfying.

Keywords: Turbo codes, concatenated codes, iterative decoding, error correction codes.

Résumé :

Les premiers Turbo Codes ont été introduit par des codes convolutifs systématiques récursifs et parallèles. Ces codes utilisés en codage de canal, étaient capables d'atteindre un taux d'erreur (BER) arbitrairement faible de 10^{-5} avec un SNR de 0.7 dB. Les turbo codes peuvent être utilisés dans tous les systèmes de communications qui exigent le minimum d'énergie. Les communications dans l'espace, les communications mobiles par satellite/cellulaire, les liaisons micro-ondes, etc.., sont quelques exemples d'applications de cette nouvelle technique de codage. Ce travail décrit les performances de la concaténation parallèle et série de deux codeurs convolutifs systématiques récursifs pour de faibles SNR (E_b/N_0 dB) itérativement décodés avec l'algorithme MAP (maximum a posteriori). En utilisant le domaine du logarithme de vraisemblance, nous montrons que chaque décodeur utilise à son entrée des valeurs a priori et délivre la sortie décodée, les entrées a priori et la valeur extrinsèque. La valeur extrinsèque est utilisée comme une valeur a priori pour la prochaine itération. Les résultats de simulation montrent comment les entrelaceurs aident les décodeurs pour améliorer leur capacité de correction en gardant l'information extrinsèque et les données reçues incorrelées, ils montrent aussi l'influence des différents paramètres sur les performances des turbo codes. Les résultats obtenus sont très satisfaisants.

Mots clés : Turbo codes, codes concaténés, décodage itératif, codes correcteurs d'erreurs.

Dédicace :

A mes parents,

A ma femme Yamina,

A ma petite fille Lamis,

A mes frères et sœurs,

A tous mes amis.

Remerciements

Il me faut tout d'abord remercier M. Berkani Daoud, professeur à l'ENP, et M. Belouchrani Adel docteur à l'ENP, pour l'honneur qu'ils m'ont fait de diriger ce travail, et pour avoir suivi mon travail tout au long de cette thèse.

J'exprime ma profonde reconnaissance à tous mes enseignants de l'ENP.

Je tiens à remercier M. Boudjema Fares, professeur à l'Ecole Nationale Polytechnique, pour avoir présidé le jury de cette thèse.

Je remercie vivement M. Aksas Rachid, maître de conférences à l'ENP, M. Terra Zidane, chargé de cours à l'ENP, et M. Achab Noureddine chargé de cours à l'EMP, pour avoir accepté de juger mon travail en participant au jury.

Merci plus spécialement à tous les cadres de l'U.E.R.Automatique de l'EMP.

Merci à tous mes collègues et à tous ceux qui m'ont soutenu.

Enfin, merci à ma femme pour sa compréhension.

Sommaire

Introduction

La demande croissante pour l'échange d'information est une caractéristique commune de la plupart des régions de la civilisation moderne. Le transfert d'information de la source à sa destination doit être fait de façon que la qualité de l'information reçue soit aussi proche que possible de la qualité de l'information transmise. Un système de communication typique peut être représenté par le schéma bloc qui suit.

L'information à transmettre peut être produite par la machine (e.g, images, données de l'ordinateur, machine parlante ou enregistrement), ou produite par l'être humain (e.g., la parole). Sans se soucier de sa source, l'information doit subir certaines transformations avant de l'envoyer sur le canal de transmission. La première étape est d'éliminer la partie redondante pour maximiser le taux de transmission de l'information. Cela est accompli par le codeur de source montré dans le schéma bloc de la page suivante. Pour assurer la confidence de l'information que nous voulons transmettre, un cryptage de données peut être utilisé. Les données doivent être aussi protégées contre les perturbations qui pourraient mener à la mauvaise interprétation du message transmis; cette protection est assurée par le codeur de canal en utilisant les techniques de contrôle des erreurs (codes correcteurs d'erreurs). Le bloc du modulateur produit un signal adapté pour le canal de transmission.

Le canal de transmission peut être un support physique, liaison aérienne, câble ou fibre par exemple, permettant le transport ou la transmission de l'information. Il peut aussi n'être qu'un simple support de stockage. Entre le codage et le décodage, l'information peut satisfaire certaines propriétés ou subir des altérations. On dit que l'information dans le canal est soumise à un bruit.

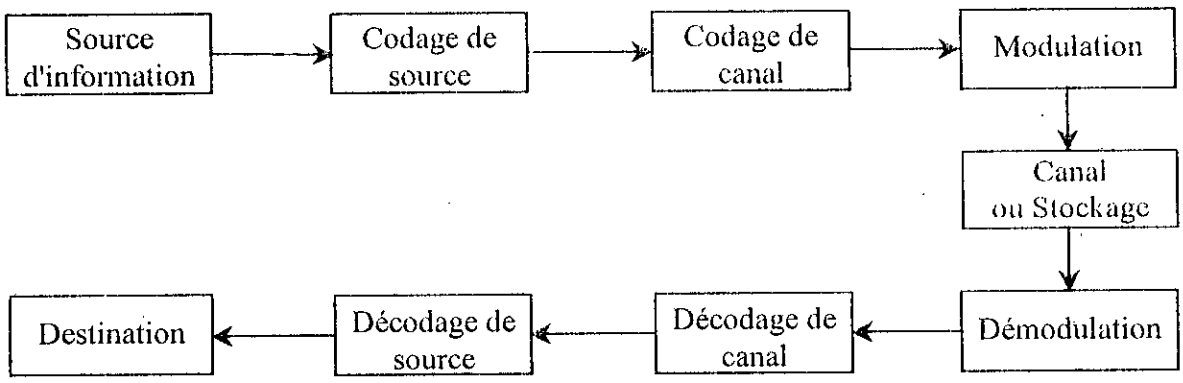


Schéma bloc d'un système de communication

Le codage de canal s'applique à un flux de bits (bit stream) émis par le codeur de source; il consiste en la transformation des blocs de bits d'information émis par le codeur de source à de nouveaux blocs de bits qui sont alors envoyés dans le canal de transmission. Cette transformation est assurée par l'ajout de bits supplémentaires (bits de parité) à la sortie du codeur de source. Pendant que ces bits supplémentaires ne transportent pas de l'information, ils rendent capable le décodeur de canal de détecter et/ou de corriger quelques erreurs dans les bits d'information.

A un codage de canal sont associés plusieurs paramètres permettant d'en apprécier la qualité. L'un d'eux est le taux de transmission (rendement du codage) qui est par définition la longueur du bloc de bits divisé par la longueur de l'image du bloc de bits obtenu par la fonction de codage. En compression de données ce taux est supérieur à un, par contre, il est inférieur à un dans le codage de canal. Toutes les techniques utilisées cherchent dans le cadre de leurs contraintes à maximiser ce taux. Un autre paramètre fondamental est la complexité du codage et du décodage. Les méthodes de codage utilisées en pratique sont souvent fondées sur des fonctions d'un type simple calculables par un automate fini.

Le travail présenté dans cette thèse traite une nouvelle classe de codes correcteurs d'erreurs introduite en 1993 par les chercheurs Berrou, Glavieux et Thitimajshima de l'ENST de Bretagne. Leur article intitulé "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes", reprenait l'idée de concaténation de codes introduite par Forney il y a plus de trente ans et montrait qu'il était possible pour un taux d'erreur de 10^{-5} et un

rendement de 0.5 d'atteindre la limite de Shannon à 0.5 dB près. Il s'agissait de concaténer deux codes convolutifs en parallèle en séparant leurs entrées par un entrelaceur : chaque code composant codant donc une version entrelacée différente de la même séquence d'information. Le décodage appliqué ensuite est itératif, chaque décodeur fournissant au suivant une information souple (valeur de confiance) lui permettant d'affiner ses propres décisions. C'est à ce caractère itératif qu'est dû le nom de "turbo" qui rappelle le fonctionnement du moteur turbo. Aujourd'hui, les turbo-codes sont devenus l'un des axes de recherche les plus importants du codage, le principe turbo tant appliqué aux codes en blocs, codes convolutifs, la concaténation tant étudiée en parallèle comme en série.

Les turbo codes ont été proposés pour les systèmes de communication exigeant le minimum d'énergie, c'est-à-dire la construction des codes qui sont capables de corriger un grand nombre d'erreurs avec une probabilité d'erreur arbitrairement faible et un faible rapport signal sur bruit (faible SNR).

Le but de ce travail est de montrer l'intérêt du codage et du décodage turbo en exposant les principes théoriques et en présentant l'influence des paramètres du codage sur le décodage itératif, ce dernier est réalisé par l'algorithme MAP (maximum a posteriori) en utilisant le domaine du logarithme de vraisemblance. Il s'agit d'étudier le plus complètement possible cette nouvelle technique de codage à la pointe du domaine des communications numériques. Des simulations du turbo-décodage appliqué à des cas simples sont présentées dans cette thèse.

Ce travail se divise en quatre chapitres.

Le premier chapitre est consacré à des généralités et à des définitions sur la théorie du codage de canal.

Le deuxième chapitre est consacré à l'introduction et à la description de quelques codes correcteurs d'erreurs.

Le troisième chapitre concerne la nouvelle technique de codage et de décodage des codes convolutifs concaténés.

Le quatrième chapitre porte sur les résultas obtenus et les performances des turbo codes.

Chapitre 1

Théorie du Codage de Canal

Entre la source d'information et sa destination, il doit y avoir un milieu à travers lequel l'information puisse se transmettre. Ce milieu, y compris l'équipement nécessaire à la transmission, s'appel canal.

Le codage de canal opère une transformation entre l'espace des symboles à l'entrée du canal et l'espace des symboles à sa sortie.

Le canal est dit discret, lorsque l'espace à l'entrée et celui à la sortie sont discrets. Lorsque la transformation du symbole x introduit à l'entrée, en symbole y, à la sortie du canal, ne dépend pas des transformations antérieures, le canal est dit sans mémoire.

1.1 Capacité du Canal

Soit une source d'information discrète et sans mémoire définie par son alphabet φ qui prend les valeurs $\{s_0,s_1,...,s_{K-1}\}$ avec respectivement les probabilités d'apparition $p_0,p_1,...,p_{K-1}$. La quantité suivante:

$$H(\varphi)=\sum_{k=0}^{K-1} p_k \log_2\left(\frac{1}{p_k}\right) \qquad (1\text{-}1)$$

représente l'entropie de la source discrète sans mémoire, qui mesure la quantité d'information moyenne par message de la source. Elle mesure aussi l'incertitude de la source d'information. Considérons maintenant un canal discret sans mémoire avec un alphabet d'entrée $\mathcal{X}$, de sortie $\mathcal{Y}$ et de probabilités de transition $p(y_k/x_j)$.

On définit l'entropie de l'alphabet $\mathcal{X}$ conditionnellement à la réception de $Y=y_k$ par :

$$H(\mathcal{X}/Y=y_k)=\sum_{j=0}^{J-1} p(x_j/y_k)\log_2\left[\frac{1}{p(x_j/y_k)}\right] \qquad (1\text{-}2)$$

Et la valeur moyenne de $H(\mathcal{X}/Y=y_k)$ de l'alphabet de sortie $\mathcal{Y}$ est donnée par:

$$H(\mathcal{X}/\mathcal{Y})=\sum_{k=0}^{K-1}\sum_{j=0}^{J-1} p(x_j,y_k)\log_2\left[\frac{1}{p(x_j/y_k)}\right] \qquad (1\text{-}3)$$

Avec J et K respectivement les nombres de symboles d'entrée et de sortie.

Cette dernière quantité représente l'incertitude moyenne sur l'entrée du canal après l'observation de sa sortie. La différence $H(\mathcal{X})$-$H(\mathcal{X}/\mathcal{Y})$ représente l'incertitude de l'entrée du canal avec l'observation de la sortie du canal. Cette importante quantité est appelée l'information mutuelle du canal et est notée par [28] :

$$I(\mathcal{X};\mathcal{Y})=H(\mathcal{X})-H(\mathcal{X}/\mathcal{Y}) \qquad (1\text{-}4)$$

Comme dans l'équation (1-1), l'entropie de l'alphabet $\mathcal{X}$ est : $H(\mathcal{X})=\sum_{j=0}^{J-1} p(x_j)\log_2\left(\frac{1}{p(x_j)}\right)$

En remplaçant $H(\mathcal{X})$ et $H(\mathcal{X}/\mathcal{Y})$ par leurs expressions, on obtient alors:

$$I(\mathcal{X};\mathcal{Y})=\sum_{k=0}^{K-1}\sum_{j=0}^{J-1} p(x_j,y_k)\log_2\left[\frac{p(y_k/x_j)}{p(y_k)}\right] \qquad (1\text{-}5)$$

Avec: $\mathcal{X}=\{x_0,x_1,...,x_{I-1}\}$ et $\mathcal{Y}=\{y_0,y_1,...,y_{K-1}\}$.

On définit la capacité d'un canal discret sans mémoire par la moyenne maximale de l'information mutuelle $I(\mathcal{X},\mathcal{Y})$, qui s'écrit sous la forme suivante:

$$C=\max_{\{p(x_j)\}} I(\mathcal{X},\mathcal{Y}) \qquad (1\text{-}6)$$

A noter que la capacité du canal C est en fonction uniquement des probabilités de transition $p(y_k/x_j)$ qui définissent le canal.

Exemple (1-1):

Supposons un canal binaire symétrique avec $I=K=2$. Le canal possède deux symboles d'entrée $(x_0=0, x_1=1)$ et deux symboles de sortie $(y_0=0, y_1=1)$. Le canal est dit symétrique parce que la probabilité de recevoir "1" lorsque "0" est émis est égale à la probabilité de recevoir "0" lorsque "1" est émis. Cette probabilité conditionnelle d'erreur ou probabilité de transition est notée par **p**.

Le diagramme des probabilités de transition de ce canal binaire symétrique (BSC) est le suivant:

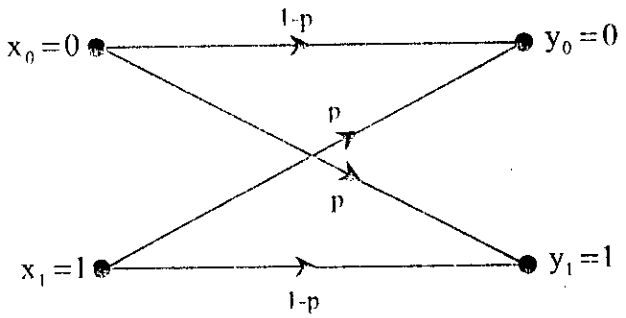


Figure (1-1): Canal binaire symétrique

Soit maintenant $p(x_0)=p(x_1)=\frac{1}{2}$, de la figure ci-dessus, on a:

$$p(y_0/x_1)=p(y_1/x_0)=p$$

et $$p(y_0/x_0)=p(y_1/x_1)=1-p$$

La capacité du canal sera calculée pour: $C = I(\mathcal{X},\mathcal{Y})\Big|_{p(x_0)=p(x_1)=\frac{1}{2}}$ (1-7)

En substituant ces probabilités de transition dans l'équation (1-5) avec J=K=2, en tenant compte que $p(x_0)=p(x_1)$, on trouve que la capacité du canal binaire symétrique est de la forme suivante:

$$C=1+p\log_2(p)+(1-p)\log_2(1-p) \tag{1-8}$$

Si on définit la fonction d'entropie par :

$$\mathcal{H}(p)=p\log_2\left(\frac{1}{p}\right)+(1-p)\log_2\left(\frac{1}{1-p}\right) \tag{1-9}$$

L'équation (1-8) peut s'écrire maintenant comme suit:

$$C = 1 - \mathcal{H}(p) \tag{1-10}$$

La capacité du canal varie en fonction de la probabilité d'erreur "p" comme le montre la figure (1-2).

1. Lorsque le canal est sans bruit c.a.d: p = 0, la capacité du canal C atteint sa valeur maximale de 1 bit, qui est égale à la quantité d'information de l'entrée du canal. Pour cette valeur de p, l'entropie $\mathcal{H}(p)$ atteint sa valeur minimale de zéro.
2. Lorsque le canal est bruité, produisant la probabilité conditionnelle d'erreur p = 1/2, la capacité du canal atteint sa valeur minimale de zéro, où la fonction d'entropie atteint sa valeur maximale de l'unité.

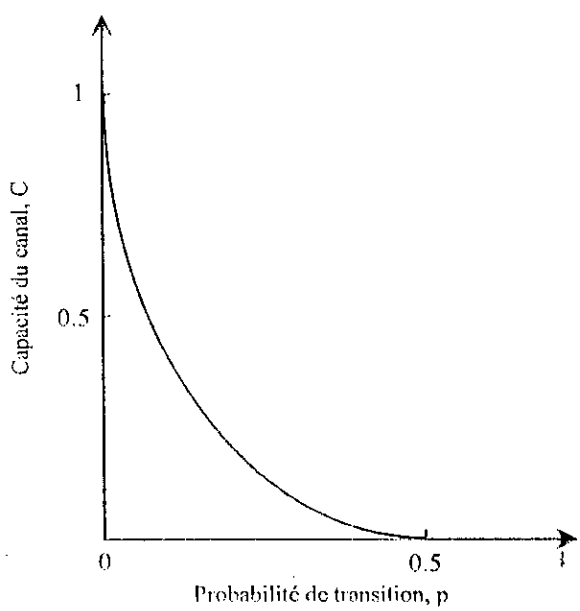


Figure (I-2): Evolution de la capacité du canal binaire symétrique en fonction de la probabilité de transition p

1.2 Théorème du Codage de Canal

Le codage de canal consiste à adapter la source au canal afin de minimiser la probabilité d'erreur, ou aussi il consiste à augmenter la résistance du système de communication pour combattre le bruit de canal.

Le codage de canal se résume en deux opérations : l'une est l'opération du codage, l'autre est l'opération de décodage, comme le montre la figure suivante:

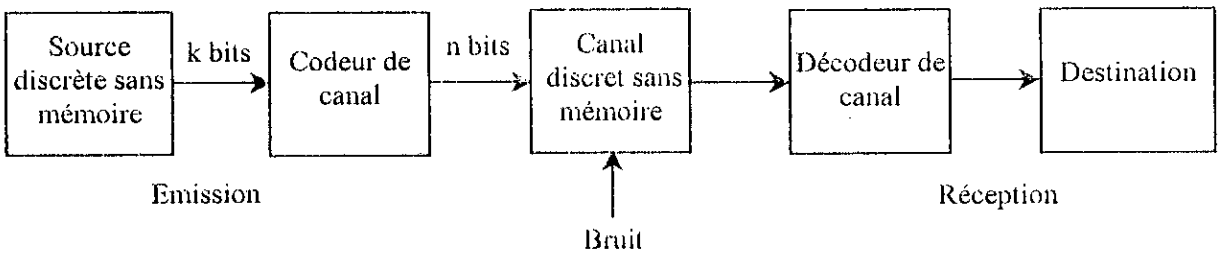


Figure (1-3): Système de transmission de l'information

Dans la figure ci-dessus, on définit le taux du codage R qui est donné par : $R=\frac{k}{n}$, avec k est la longueur du message d'entrée et n est la longueur de la séquence de sortie, par contre, n - k représente le nombre de bits redondants dans chaque bloc transmis de longueur n. Le théorème du codage de canal annonce ce qui suit:

Soit une source discrète sans mémoire possédant un alphabet φ avec une entropie $H(\varphi)$ et produisant des symboles chaque T_s secondes, soit aussi un canal discret sans mémoire avec une capacité C qui est utilisé pendant T_c secondes. Donc si:

$$\frac{H(\varphi)}{T_s} \leq \frac{C}{T_c} \qquad (1\text{-}11)$$

On dira qu'il existe un codage pour que la sortie de la source puisse être transmise à travers le canal, et que celle ci sera reconstruite avec une probabilité d'erreur minimale.

1.2.1 Application au Canal Binaire Symétrique:

Considérons une source discrète sans mémoire émettant des symboles binaires (0 et 1) équiprobables chaque T_s secondes. Ce qui donne une entropie $H(\varphi)=1$ bit/symbole, le taux

d'information de la source est de $\frac{1}{T_s}$ bits/seconde. La séquence de la source est appliquée au codeur de canal avec un taux de codage de R. Le codeur produit des symboles chaque T_c secondes avec un taux de transmission de $\frac{1}{T_c}$ symboles/seconde, et comme la capacité du canal par unité de temps est de $\frac{C}{T_c}$ bits/seconde, l'équation (1-11) devient alors:

$$\frac{1}{T_s} \leq \frac{C}{T_c} \tag{1-12}$$

Et le rapport $\frac{T_c}{T_s}$ est égal au taux de codage du codeur :

$$R = \frac{T_c}{T_s} \tag{1-13}$$

De l'équation (1-12) on trouve : $R \leq C$ (1-14)

Donc pour $R \leq C$, il existe un code avec un taux de codage inférieur ou égal à C capable de donner une faible probabilité d'erreur.

1.3 Théorème de la Capacité du Canal

Considérons une variable aléatoire continue X avec une densité de probabilité $f_x(x)$. Du fait qu'il est impossible de définir l'entropie H(X) d'une variable continue, il a été défini une entropie différentielle de X comme suit:

$$h(X) = \int_{-\infty}^{+\infty} f_x(x) \log_2 \left[\frac{1}{f_x(x)} \right] dx \tag{1-15}$$

On démontre que le maximum de h(X) est atteint pour une densité de probabilité gaussienne [28] donnée par:

$$f_x(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right) \tag{1-16}$$

Avec μ et σ^2 : respectivement la moyenne et la variance de la variable aléatoire gaussienne X.

En remplaçant $f_x(x)$ par son expression dans l'équation (1-15) et après calcul de cette intégrale, on obtient

$$h(X)=\frac{1}{2}\log_2(2\pi e\sigma^2) \qquad (1\text{-}17)$$

Soit maintenant un processus stationnaire X(t) de moyenne zéro et de bande de fréquence limitée à B hertz. Si on échantillonne le processus X(t) à la fréquence 2B échantillons par seconde, alors on obtient les variables aléatoires continues X_k, avec $k=1,2,...,n$. Ces échantillons sont transmis en T secondes à travers un canal bruité possédant aussi la même bande passante de B hertz.

Alors le nombre d'échantillons est donné par

$$n=2BT \qquad (1\text{-}18)$$

La sortie du canal est perturbée par un bruit blanc gaussien de moyenne zéro, de densité spectrale de puissance $N_0/2$ ($N_0=C^{te}$) et de bande de fréquence de B hertz, produisant les variables aléatoires continues Y_k ($k=1,2,...,n$) après échantillonnage du signal reçu. Les Y_k sont de la forme:

$$Y_k=X_k+N_k \qquad k=1,2,...,n \qquad (1\text{-}19)$$

Le bruit N_k est gaussien, de moyenne zéro et de variance donnée par :

$$\sigma^2=N_0B \qquad (1\text{-}20)$$

On définit la puissance moyenne P du signal transmis par :

$$E\left[X_k^2\right]=P \quad k=1,2,...,n \qquad (1\text{-}21)$$

Par conséquent : la capacité du canal devient alors [28] :

$$C=\max_{f_{X_k}(x)}\left\{I(X_k,Y_k):E\left[X_k^2\right]=P\right\} \qquad (1\text{-}22)$$

Où $I(X_k,Y_k)$ est l'information mutuelle entre les échantillons du signal transmis X_k et les échantillons du signal reçu, Y_k.

En comparant l'équation (1-6) avec l'équation (1-22), on remarque que dans cette dernière on doit définir la puissance moyenne P du signal transmis.

Par analogie avec l'expression (1-4), l'information mutuelle $I(X_k,Y_k)$ s'écrit :

$$I(X_k,Y_k)=h(X_k)-h(X_k/Y_k) \qquad (1\text{-}23)$$

Et comme l'information mutuelle est symétrique $I(X_k,Y_k)=I(Y_k,X_k)$ alors :

$$I(X_k,Y_k)=h(Y_k)-h(Y_k/X_k) \tag{1-24}$$

Les variables aléatoires X_k et N_k sont indépendantes, alors l'entropie différentielle et conditionnelle de Y_k, sachant X_k, est égale à l'entropie différentielle de N_k [28]:

$$h(Y_k/X_k)=h(N_k) \tag{1-25}$$

De cette expression, l'équation (1-24) devient alors

$$I(X_k,Y_k)=h(Y_k)-h(N_k) \tag{1-26}$$

Et comme $h(N_k)$ est indépendante de la distribution de X_k, la maximisation de $I(X_k,Y_k)$ revient alors à maximiser $h(Y_k)$. Pour maximiser $h(Y_k)$, Y_k doit être une variable aléatoire gaussienne, et comme N_k est gaussienne par hypothèse alors la variable X_k du signal transmis doit être gaussienne aussi.

Donc, la maximisation de la capacité du canal de l'expression (1-22) est atteinte en choisissant les échantillons du signal émis à partir d'un processus gaussien, de puissance moyenne P. L'équation (1-22) sera reformulée comme suit :

$$C=I(X_k,Y_k):X_k \text{ Gaussienne}, E\left[X_k^2\right]=P \tag{1-27}$$

Où l'information mutuelle est donnée par l'équation (1-26).

Pour évaluer la capacité du canal, on procède comme suit:

1. La variance de l'échantillon Y_k du signal reçu est égale à $P+\sigma^2$ et en utilisant l'équation (1-17), l'entropie différentielle de Y_k est :

$$h(Y_k)=\frac{1}{2}\log_2\left[2\pi e(p+\sigma^2)\right] \tag{1-28}$$

2. La variance du bruit N_k est égale à σ^2 et son entropie différentielle est :

$$h(N_k)=\frac{1}{2}\log_2\left(2\pi e\sigma^2\right) \tag{1-29}$$

3. En substituant l'équation (1-28) et l'équation (1-29) dans la relation (1-26), on trouve le résultat suivant:

$$C=\frac{1}{2}\log_2\left(1+\frac{P}{\sigma^2}\right) \text{ bits/canal utilisé} \tag{1-30}$$

Le canal est utilisé en transmission par n échantillons du processus X(t) pendant T secondes et le nombre d'échantillons/seconde est de 2B ; donc la capacité du canal par unité du temps est :

$$C = B\log_2\left(1+\frac{P}{N_0 B}\right) \text{ bits/s} \tag{1-31}$$

Le théorème de la capacité du canal annonce ce qui suit : soit un signal à transmettre de puissance moyenne P et un canal de bande passante B ; alors, on peut transmettre de l'information à travers ce canal avec un taux de C bits/s et avec une probabilité d'erreur arbitrairement faible à condition que le débit de transmission reste inférieur à la capacité du canal.

Pour approximer cette limite, le signal à transmettre doit avoir approximativement les même propriétés statistiques que le bruit blanc gaussien [28].

1.3.1 Canal Idéal

On définit un système idéal lorsqu'on transmet des données avec un débit de R_b égal à la capacité du canal C. Dans ce cas, on exprime la puissance moyenne du signal transmis par :

$$P = E_b C \tag{1-32}$$

Avec E_b : l'énergie moyenne transmise par bit. Le système idéal est défini par l'équation

$$\frac{C}{B} = \log_2\left(1+\frac{E_b C}{N_0 B}\right) \tag{1-33}$$

De cette expression, on tire le rapport E_b/N_0 qui vaut

$$\frac{E_b}{N_0} = \frac{2^{C/B}-1}{C/B} \tag{1-34}$$

Le tracé de R_b/B en fonction de E_b/N_0 est représenté dans la figure (1-4). On remarque ce qui suit:

- Pour une bande passante infinie, le rapport signal sur bruit approche la limite suivante:

$$\lim_{B\to\infty}\left(\frac{E_b}{N_0}\right) = \ln(2) = 0.693 \tag{1-35}$$

Cette valeur est appelée la limite de Shannon, en décibels, elle vaut -1.6 dB. La limite correspondante de la capacité du canal est obtenue en calculant la limite de l'équation

(1-31), ce qui donne : $$\lim_{B \to \infty} C = \frac{P}{N_0} \log_2(e) \qquad (1\text{-}36)$$

- La limite de Shannon définie aussi le rapport signal sur bruit E_b/N_0 qu'on peut retenir pour qu'une transmission avec un taux d'erreur arbitrairement faible soit possible.

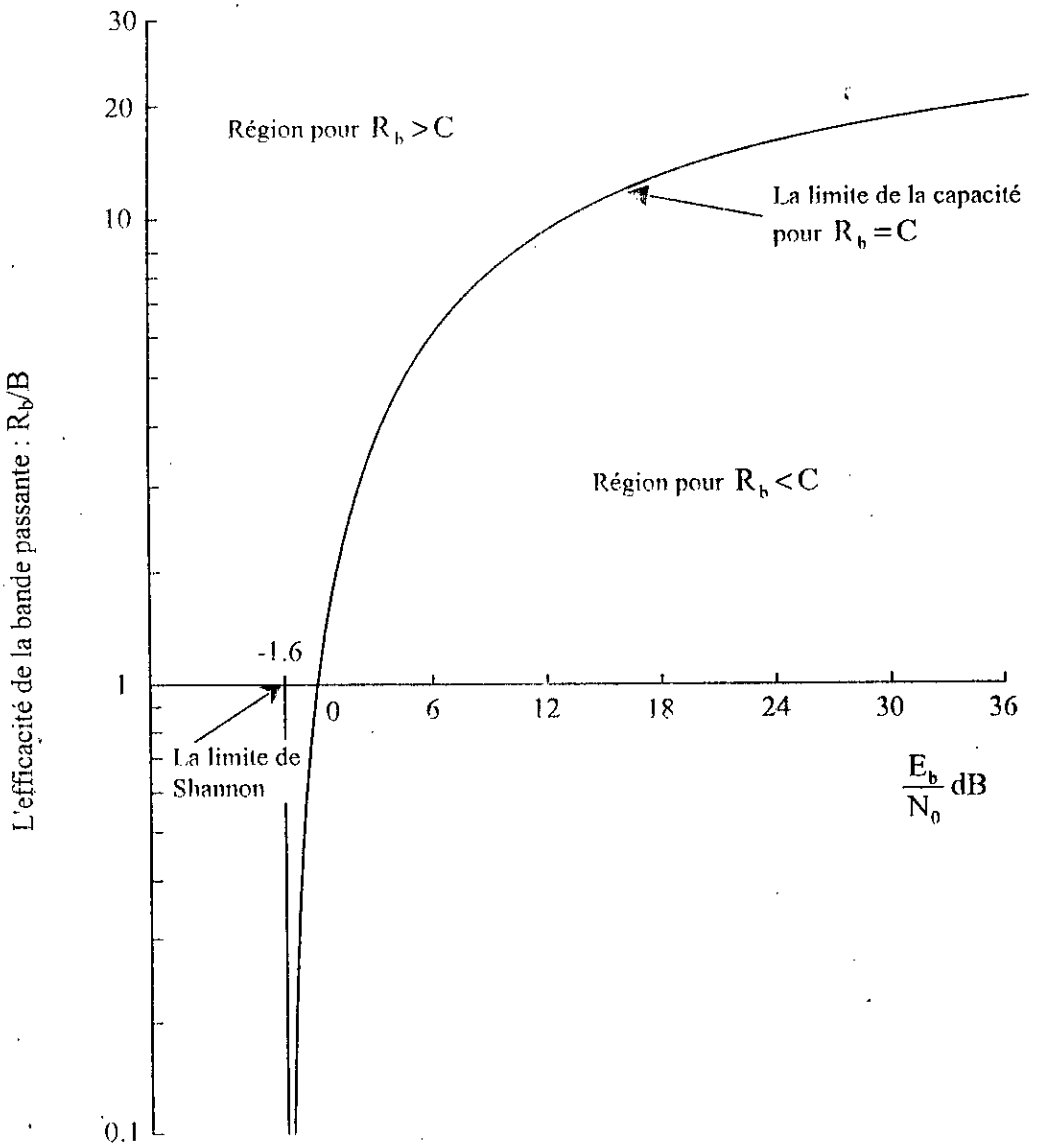


Figure (1-4): Le diagramme d'efficacité de la bande passante

Chapitre 2

Codes Correcteurs d'Erreurs

Les codeurs et les décodeurs étant utilisés dans les systèmes de transmission d'information ou de stockage de données, leurs caractéristiques doivent être adaptées aux contraintes physiques et technologiques des systèmes auxquels ils sont destinés. Nous présentons quelques grandes familles de systèmes de codage opérant sur des séquences binaires, parmi lesquelles on trouve les codes destinés à la détection et/ou la correction d'erreurs. Cette présentation permet de situer la place de ces derniers, qui sont appelés aussi les codes correcteurs d'erreurs.

2.1 Codage Linéaire par Blocs

Dans un codage par blocs, les suites de symboles de l'alphabet source sont partagées en blocs successifs de longueurs constantes, la longueur étant le nombre de symboles du bloc. Ces blocs ne se chevauchent donc pas. Chaque bloc est transformé en un autre bloc constitué de symboles d'un autre alphabet. Ce dernier est l'alphabet d'entrée du canal de transmission. Comme les blocs sont de longueurs constantes, aussi bien à la source que dans le canal, le taux de transmission d'un tel codage est égal au rapport de la longueur d'un bloc source par la longueur d'un bloc canal.

Le codage par bloc est fréquemment utilisé en télécommunication pour la correction et la détection d'erreurs. L'alphabet de la source est, comme l'alphabet du canal, égal à {0,1}. Chaque bloc de longueur k est codé par un bloc de longueur n avec $n \geq k$. Les k premiers bits du bloc codé sont une simple recopie du bloc source, les n-k restants sont obtenus par une combinaison linéaire des k premiers. La terminologie usuelle traduit cette différence de nature entre les k premiers bits, qui sont dits bits d'information, et les n-k restants que l'on nomme bits de parité (bits de correction). Nous présentons ci-dessous un exemple de codage de ce type emprunté à la famille des codes de Hamming.

Considérons maintenant un code linéaire par blocs de dimension (n,k), composé d'une séquence de message $m_0,m_1,\ldots,m_{k-1}$ et de bloc de parité $b_0,b_1,\ldots,b_{n-k-1}$, ce type de codage produit un mot code $x_0,x_1,\ldots,x_{n-1}$:

$$\text{Avec : } x_i = \begin{cases} m_i & \text{pour } i = 0,1,\ldots,k-1 \\ b_{i-k} & \text{pour } i = k,k+1,\ldots,n-1 \end{cases} \qquad (2\text{-}1)$$

Les b_i $(i = 0,1,\ldots,n-k-1)$ sont calculés linéairement par l'expression suivante :

$$b_i = p_{i,0}m_0 + p_{i,1}m_1 + \ldots + p_{i,k-1}m_{k-1} \qquad (2\text{-}2)$$

Où : $p_{i,j} = 1$ ou 0, l'opération d'addition dans l'équation (2-2) se fait en modulo-2 $(1+1=0, 0+1=1, 1+0=1, 0+0=0)$.

Matriciellement, on peut écrire :

$$\mathbf{x} = [\mathbf{m} : \mathbf{b}] \qquad (2\text{-}3)$$

$$\text{Avec} \quad \mathbf{m} = [m_0, m_1, \ldots, m_{k-1}] \text{ de dimension } (1,k) \qquad (2\text{-}4)$$

$$\mathbf{b} = [b_0, b_1, \ldots, b_{n-k-1}] \text{ de dimension } (1,n-k) \qquad (2\text{-}5)$$

$$\text{Et} \quad \mathbf{b} = \mathbf{mP} \qquad (2\text{-}6)$$

P est définie par les $k \times (n-k)$ coefficients de la matrice suivante :

$$\mathbf{P} = \begin{bmatrix} p_{00} & p_{10} & \cdots & p_{n-k-1,0} \\ p_{01} & p_{11} & \cdots & p_{n-k-1,1} \\ \vdots & \vdots & & \vdots \\ p_{0,k-1} & p_{1,k-1} & \cdots & p_{n-k-1,k-1} \end{bmatrix} \tag{2-7}$$

En remplaçant **b** par son expression dans l'équation (2-3), on obtient alors: $\mathbf{x}=[\mathbf{m}:\mathbf{b}]=[\mathbf{m}:\mathbf{mP}]=\mathbf{m}[\mathbf{I}_k:\mathbf{P}]$

Où $\mathbf{I}_k$ est la matrice identité de $k \times k$:

$$\mathbf{I}_k = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{2-8}$$

On définit la matrice génératrice **G** de dimension (k,n) par : $\mathbf{G}=[\mathbf{I}_k:\mathbf{P}]$, donc on écrira :

$$\mathbf{x}=\mathbf{mG} \tag{2-9}$$

On définit maintenant la matrice de parité **H** de $(n-k)\times n$ par:

$$\mathbf{H}=[\mathbf{P}^t:\mathbf{I}_{n-k}] \tag{2-10}$$

Où $\mathbf{P}^t$ est une matrice de $(n-k)\times k$, qui représente la matrice transposée de **P** :

La matrice de parité **H** peut être utilisée, pour vérifier si un mot code **x** est généré par la matrice génératrice $\mathbf{G}=[\mathbf{I}_k:\mathbf{P}]$. Cette vérification peut être faite comme suit : **x** est un mot code du code linéaire par bloc (n,k) généré par **G** si et seulement si $\mathbf{xH}^t=\mathbf{mGH}^t=\mathbf{0}$.

Ce qui donne: $$\mathbf{GH}^t=\mathbf{0} \tag{2-11}$$

Pendant que la matrice génératrice est utilisée dans l'opération du codage, la matrice de parité est utilisée dans l'opération de décodage comme suit: considérons un code linéaire par blocs (n,k) avec une matrice génératrice $\mathbf{G}=[\mathbf{I}_k:\mathbf{P}]$ et une matrice de parité $\mathbf{H}=[\mathbf{P}^t:\mathbf{I}_{n-k}]$. Soit **x** un mot code qui a été transmis sur un canal à bruit additif et **y** le mot code reçu. Le mot code **y** est la somme du mot code original **x** et du vecteur d'erreur **e**, ce qui donne, **y = x+e**.

Le récepteur ne connaît pas **x** et **e**; sa fonction est de décoder (extraire) **x** de **y**, et le message **m** de **x**. Le récepteur fait l'opération de décodage en déterminant le vecteur **S** de dimension (1,n-k) qui est défini par : $\mathbf{S} = \mathbf{y}\mathbf{H}^t = (\mathbf{x}+\mathbf{e})\mathbf{H}^t$.

Le vecteur S est appelé le vecteur syndrome de **y**. Et comme $\mathbf{x}\mathbf{H}^t = \mathbf{0}$, alors on obtient :

$$\mathbf{S} = \mathbf{e}\mathbf{H}^t \tag{2-12}$$

Le vecteur **S** est utilisé pour la détection des erreurs dans le message émis.

2.1.1 Distance Minimale du Codage Linéaire par Blocs.

On définit la distance de Hamming entre deux vecteurs ou mots de code par le nombre de positions des bits dont ils diffèrent. La distance minimale d_{min} est définie par la plus petite distance de Hamming entre les paires des vecteurs de code.

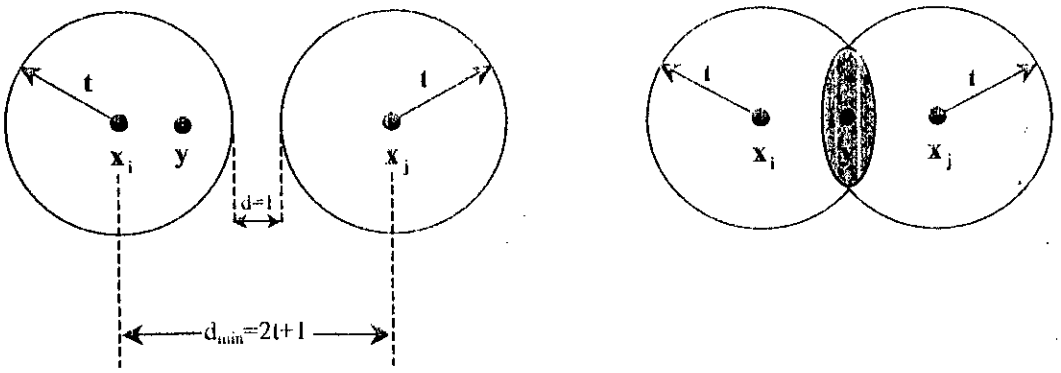
La distance minimale d_{min} du codage linéaire par blocs est un paramètre d'une très grande importance. Il détermine la capacité de détection et de correction des erreurs dans le code.

Soit **t** le nombre d'erreurs qu'on peut corriger, alors **t** doit vérifier la relation suivante [30]:

$$\mathbf{t} \le \frac{1}{2}(\mathbf{d}_{min} - 1) \tag{2-13}$$

Pour démontrer cette condition, supposons qu'on a transmis le vecteur code $\mathbf{x}_i$; à la réception, on aura le vecteur $\mathbf{y} = \mathbf{x}_i + \mathbf{e}$, on cherche à la sortie du décodeur d'obtenir $\hat{\mathbf{x}} = \mathbf{x}_i$, en tenant compte de la condition (2-13) , le poids de Hamming de l'erreur **e** doit vérifier : $\mathbf{w(e)} \le \mathbf{t}$, où le poids de Hamming **w** mesure le nombre de bits non nuls. La meilleure stratégie que pourra suivre le décodeur, est de chercher le vecteur code $\hat{\mathbf{x}}$ entre tous les vecteurs code existants, vérifiant la plus petite distance de Hamming, c.a.d $d(\hat{\mathbf{x}}, \mathbf{y})$ soit minimale.

Et pour mieux illustrer ça, nous représentons deux vecteurs code $\mathbf{x}_i$ et $\mathbf{x}_j$ dans un espace de dimension n, entourés par deux sphères de rayons **t**. La condition $d(\mathbf{x}_i, \mathbf{x}_j) \ge 2\mathbf{t}+1$ sera vérifiée dans le cas où les deux sphères seraient disjointes comme le montre la figure (2-1(a)), pour ce cas, la distance de Hamming entre le vecteur reçu **y** et le mot code $\mathbf{x}_i$ vérifie la condition $d(\mathbf{x}_i, \mathbf{y}) \le \mathbf{t}$ et le décodeur nous donne le vecteur code $\mathbf{x}_i$ qui est le plus proche de **y**. Par contre, si le vecteur reçu **y** appartient à l'espace commun entre les deux sphères comme le montre la figure (2-1(b)), le décodeur peut commettre une erreur en donnant à la sortie le vecteur code $\mathbf{x}_j$ qui vérifie la condition $d(\mathbf{x}_i, \mathbf{x}_j) < 2\mathbf{t}$.



(a) (b)

Figure (2-1): (a) distance de Hamming où $d(\mathbf{x}_i, \mathbf{x}_j) \geq 2t+1$,
(b) distance de Hamming où $d(\mathbf{x}_i, \mathbf{x}_j) < 2t$.

Exemple (2-1):

Prenons un code linéaire par blocs de dimension (6,3) avec les paramètres suivants:

$n = 6$: la longueur du bloc du mot code.

$k = 3$: le nombre de bits du message à transmettre.

$n - k = 3$: le nombre de bits de parité.

Et soit la matrice génératrice suivante:

$$\mathbf{G} = \underbrace{\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{array}\right]}_{I_k \qquad P} \tag{2-14}$$

Si on applique la définition de l'équation (2-10), on trouve la matrice de parité suivante:

$$\mathbf{H} = \underbrace{\left[\begin{array}{ccc|ccc} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right]}_{P^t \qquad I_{n-k}} \tag{2-15}$$

Pour k = 3, on a 8 mots de message différents, qui sont représentés dans le tableau suivant avec leurs mots code et leurs poids de Hamming.

mots de message	mots code	poids des mots code
0 0 0	0 0 0 0 0 0	0
0 0 1	0 0 1 1 0 1	3
0 1 0	0 1 0 0 1 1	3
0 1 1	0 1 1 1 1 0	4
1 0 0	1 0 0 1 1 0	3
1 0 1	1 0 1 0 1 1	4
1 1 0	1 1 0 1 0 1	4
1 1 1	1 1 1 0 0 0	3

Tableau (2-1): mots code du codage (6,3) de Hamming

Dans le tableau ci-dessus, on remarque bien que le plus petit poids au sens de Hamming des mots code existants excepté le mot code zéro est de 3, qui est la distance minimale de Hamming.

Si on suppose que le vecteur d'erreur **e** possède un seul bit d'erreur, alors le vecteur syndrome **S** de l'équation (2-12) donne les résultats suivants:

Vecteurs syndrome	Vecteurs d'erreurs
0 0 0	0 0 0 0 0 0
1 1 0	1 0 0 0 0 0
0 1 1	0 1 0 0 0 0
1 0 1	0 0 1 0 0 0
1 0 0	0 0 0 1 0 0
0 1 0	0 0 0 0 1 0
0 0 1	0 0 0 0 0 1

Tableau (2-2): Tableau de décodage du codage de Hamming (6,3) défini par le tableau (2-1)

Comme le montre le tableau (2-2), le vecteur **S** nous renseigne sur la position du bit dans le message affecté par une erreur et on remarque bien que pour ce code, on ne pourra pas corriger plus d'une erreur, comme le montre l'exemple suivant.

Exemple (2-2):

Soit maintenant à simuler une erreur et deux erreurs du code linéaire (6,3) précédant pour le mot code suivant: **x** = (1 0 1 0 1 1).

Pour le cas d'une erreur, on prend le vecteur erreur **e** = (0 0 0 1 0 0), ce qui donne le vecteur code reçu **y** = (1 0 1 1 1 1). Le calcul du vecteur syndrome **S** par l'équation $\mathbf{S} = \mathbf{yH}^t$ donne le résultat suivant **S** = (1 0 0) et en utilisant le tableau (2-2), on obtient le vecteur erreur **e** = (0 0 0 1 0 0) qui nous permet de corriger l'erreur par la relation **x = y+e**.

Et dans le cas de deux erreurs, on prend le vecteur erreur **e** = (0 1 0 1 0 0). A la réception on aura le vecteur **y** = (1 1 1 1 1 1), dans ce cas, le vecteur syndrome **S** aura la valeur (1 1 1) qui n'existe pas dans le tableau (2-2).

Donc en conclusion, le code linéaire par blocs (6,3) défini par l'exemple (2-1) peut corriger uniquement une seule erreur et peut détecter la présence de deux erreurs.

2.2 Le Code Cyclique

Un code binaire est un code cyclique s'il vérifie les deux propriétés suivantes :

- **Propriété linéaire :** La somme de deux mots code est un mot code.
- **Propriété cyclique :** Chaque décalage cyclique d'un mot code est aussi un mot code.

Soit le mot code suivant : $(x_0, x_1, ..., x_{n-1})$, le code est un code cyclique si et seulement si, les vecteurs suivants $(x_{n-1}, x_0, x_1, ..., x_{n-2})$,

$$(x_{n-2}, x_{n-1}, ..., x_{n-3}),$$
$$\vdots$$
$$(x_1, x_2, ..., x_0)$$

sont tous des mots code.

Cette formulation de la propriété cyclique peut être représentée par une forme polynomiale du mot code et sera comme suit :

$$x(D) = x_0 + x_1 D + ... + x_{n-1} D^{n-1} \qquad (2\text{-}16)$$

Les puissances de D représentent les décalages cycliques vers la droite avec la condition $D^n = 1$ (lorsqu'il n'y a pas de décalage), par exemple pour un seul décalage cyclique, on écrit :

$$Dx(D) \bmod (D^n - 1) = x_{n-1} + x_0 D + ... + x_{n-2} D^{n-1} \qquad (2\text{-}17)$$

$D^i x(D)$ est aussi un mot code pour un décalage cyclique de i positions.

2.2.1 Le Polynôme Générateur

Soit un code cyclique (n,k) qui est obtenu par un polynôme générateur g(D), le degré de g(D) est égal au nombre de bits de parité du mot code.

Le polynôme générateur g(D) est équivalent à la matrice génératrice G, le degré minimal de g(D) est (n-k) et l'expression de g(D) est la suivante [30]:

$$g(D) = 1 + \sum_{i=1}^{n-k-1} g_i D^i + D^{n-k} \quad \text{avec} \quad g_i = 0,1. \tag{2-18}$$

La structure du mot code **x** est de la forme suivante: $\underbrace{(b_0, b_1, \ldots, b_{n-k-1}}_{\text{n-k bits de parité}}, \underbrace{m_0, m_1, \ldots, m_{k-1})}_{\text{k bits de message}}$.

Pour avoir cette structure le polynôme du message sera multiplié par D^{n-k}, ce qui donne

$$D^{n-k} m(D) = m_0 D^{n-k} + m_1 D^{n-k+1} + \cdots + m_{k-1} D^{n-1}$$

Et si on divise $D^{n-k} m(D)$ par le polynôme générateur g(D), en obtenant le quotient a(D) et le reste b(D), alors on peut écrire:

$$D^{n-k} m(D) = a(D)g(D) + b(D) \tag{2-19}$$

Où, généralement,

$$a(D) = a_0 + a_1 D + \ldots + a_{k-1} D^{k-1} \tag{2-20}$$

et

$$b(D) = b_0 + b_1 D + \ldots + b_{n-k-1} D^{n-k-1} \tag{2-21}$$

En modulo-2, $b(D) = -b(D)$, et en réarrangeant l'équation (2-19), on obtient

$$b(D) + D^{n-k} m(D) = a(D)g(D) \tag{2-22}$$

Ce polynôme, qui est le multiple du polynôme générateur g(D), représente le polynôme du mot code du code cyclique (n,k), généré par g(D). Et finalement, on écrit

$$x(D) = b(D) + D^{n-k} m(D) \tag{2-23}$$

En résumé, pour la génération du code cyclique (n,k) on procède comme suit :

1. Multiplier le message polynomial m(D) par D^{n-k}.
2. Diviser $D^{n-k} m(D)$ par le polynôme générateur g(D) pour obtenir le reste de la division b(D).
3. Additionner b(D) à $D^{n-k} m(D)$ pour obtenir le mot code $x(D) = b(D) + D^{n-k} m(D)$.

2.2.2 Le Polynôme de Parité

Le polynôme de parité doit vérifier comme la matrice de parité : $\mathbf{HG^t = 0}$, La relation suivante:

$$h(D)g(D) \bmod (D^n - 1) = 0 \tag{2-24}$$

En multipliant l'équation (2-24) par a(D) et comme $x(D) = a(D)g(D)$, on trouve alors

$$h(D)x(D) \bmod (D^n - 1) = 0 \tag{2-25}$$

A partir de l'expression (2-24) on peut déduire la relation suivante :

$$h(D)g(D) = 1 + D^n \tag{2-26}$$

Cette dernière relation sera utilisée pour la sélection du polynôme générateur ou du polynôme de parité du code cyclique[30].

Exemple (2-3):

Soit la longueur du mot code n = 7 et la longueur du message k = 4, donc la condition $h(D)g(D) = 1 + D^n = 1 + D^7$ nous donne : $1 + D^7 = (1 + D)(1 + D^2 + D^3)(1 + D + D^3)$.

Et comme le degré minimal de g(D) est de (n-k) = 3, alors $g(D) = \begin{cases} 1 + D + D^3 \\ \text{ou} \\ 1 + D^2 + D^3 \end{cases}$

Soit à prendre : $g(D) = 1 + D + D^3$;

Ce qui donne $h(D) = (1 + D)(1 + D^2 + D^3) = 1 + D + D^2 + D^4$.

Pour illustrer la procédure de construction du mot code en utilisant le polynôme générateur g(D), on se donne une séquence de message à coder : 1001, dont le polynôme correspondant est donné par:

$$m(D) = 1 + D^3$$

En multipliant m(D) par $D^{n-k} = D^3$, on obtient : $D^{n-k}m(D) = D^3(1 + D^3) = D^3 + D^6$.

Le reste de la division de $D^{n-k}m(D)$ par g(D) est égal à $b(D) = D + D^2$ et en tenant compte de l'équation (2-23), on trouve $x(D) = D + D^2 + D^3 + D^6 \equiv 0111001$.

2.3 Le Code Convolutif

Un autre type important de codage est le codage dit de convolution. La suite à coder est découpée en blocs successifs de longueur constante mais l'image d'un élément d'indice i ne dépend que des éléments d'indices (i-M, i-(M-1), . . ., i-1, i) de la suite à coder (M est le nombre de mémoires du registre à décalage du codeur).

Le codeur convolutif avec un taux de codage de $\frac{1}{n}$ est obtenu en utilisant M bascules (mémoires) d'un registre à décalage, n additionneur modulo-2 et un multiplexeur pour sérialiser la sortie.

Pour un codeur convolutif dont l'entrée est une séquence de longueur de N bits produisant un mot code de longueur n(N + M), son taux de codage est : $R = \frac{N}{n(N+M)}$ bits/symbole.

Lorsque $N >> M$, on obtient $R \approx \frac{1}{n}$.

Le principe du codeur convolutif (mono entrée et double sortie R=1/2) est le suivant :

On applique à l'entrée du codeur convolutif la séquence du message suivant $(m_0, m_1, m_2 ...)$, le codeur génère deux séquences en sortie $\{x_i^1\}$ et $\{x_i^2\}$ qui sont données par les deux expressions suivantes :

$$x_i^1 = \sum_{l=0}^{M} g_l^{(1)} m_{i-l} \qquad i = 0,1,2,....$$
$$x_i^2 = \sum_{l=0}^{M} g_l^{(2)} m_{i-l} \qquad i = 0,1,2,.... \qquad (2\text{-}27)$$

Les $g_l^{(j)}$ représentent les séquences génératrices du codeur qui s'appellent aussi les réponses impulsionnelles du codeur.

Après le calcul du produit de convolution, les deux séquences $\{x_i^1\}$ et $\{x_i^2\}$ sont combinées par le multiplexeur pour produire le mot code $\{x_i\}$ qui est égal à :

$$\{x_i\} = \{x_0^{(1)}, x_0^{(2)}, x_1^{(1)}, x_1^{(2)}, x_2^{(1)}, x_2^{(2)}, ...\}.$$

Exemple (2-4):

Soit le codeur décrit dans la figure (2-2) avec M = 2, pour ce type de codeur on déduit que :

$(g_0^{(1)}, g_1^{(1)}, g_2^{(1)}) = (1,1,1) = 7$ en forme octale.

$(g_0^{(2)}, g_1^{(2)}, g_2^{(2)}) = (1,0,1) = 5$ en forme octale.

Et soit maintenant le message d'entrée suivant $(m_0,m_1,m_2,m_3,m_4) = (10011)$, en utilisant les deux expressions de (2-27) on obtient :

$$\left\{x_0^{(1)},x_1^{(1)},x_2^{(1)},x_3^{(1)},x_4^{(1)},x_5^{(1)},x_6^{(1)}\right\}=(1111001)$$
$$\left\{x_0^{(2)},x_1^{(2)},x_2^{(2)},x_3^{(2)},x_4^{(2)},x_5^{(2)},x_6^{(2)}\right\}=(1011111)$$

Finalement en multiplexant les deux séquences de sorties, on trouve le mot code suivant :

$x_i = \{11,10,11,11,01,01,11\}$

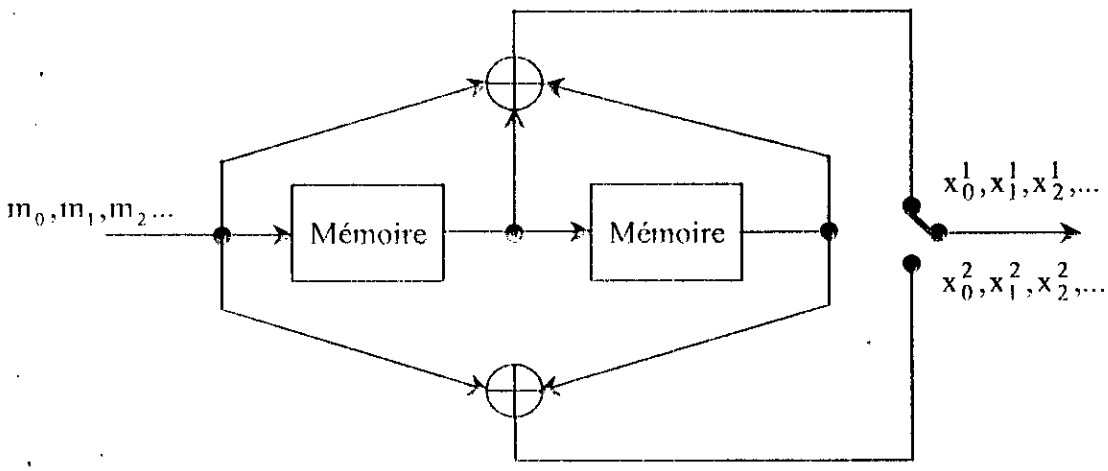


Figure (2-2) : Codeur convolutif avec R =1/2 et M=2

2.3.1 La Structure en Treillis du Codeur Convolutif

Pour illustrer bien cette structure, on prend l'exemple précédent où M = 2 et n = 2 comme le montre la figure (2-3).

Les quatre états (a,b,c,d) du codeur représentent le contenu du registre à décalage (ensemble de mémoires), à chaque instant le codeur contient (prend) un seul état.

Dans cette structure, les branches dont les codes sont produits par des entrées binaires "0" sont tracées en trait fort, par contre, celles dont les codes sont produits par des entrées "1" sont tracées en pointillé.

Donc, chaque séquence binaire d'entrée lui correspond un chemin spécifique dans cette structure en treillis.

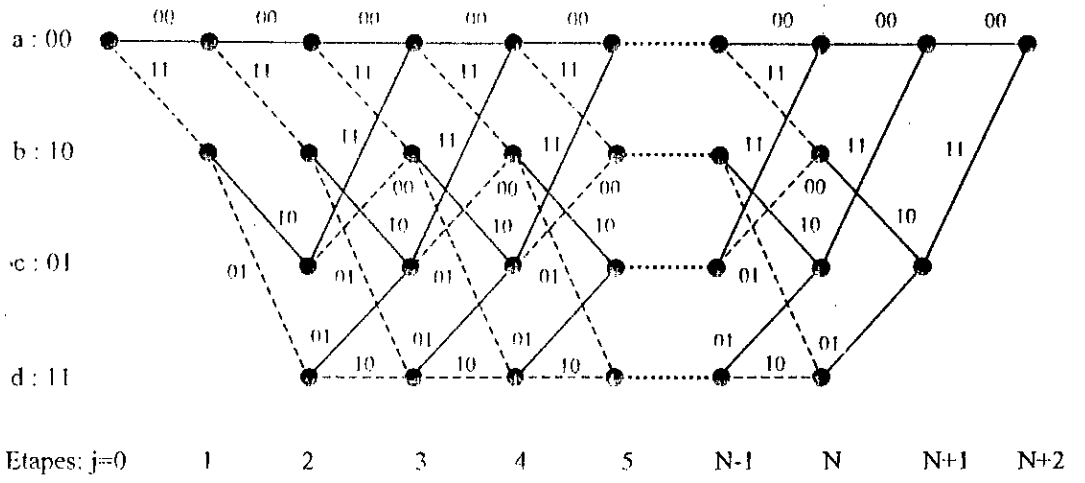


Figure (2-3): Structure en treillis du code convolutif R =1/2 et M=2

Si on suppose maintenant $j \geq 2$, les différentes transitions d'une étape à l'autre sont les suivantes :

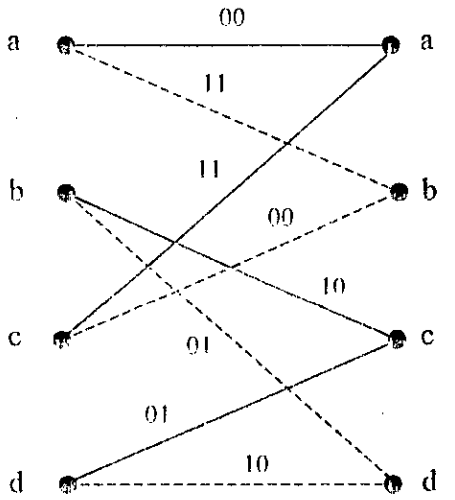


Figure (2-4): Section de la structure en treillis

Les chiffres binaires des branches-code correspondent à la sortie instantanée du codeur convolutif. En réception, on démontre que le maximum de vraisemblance du décodage du code convolutif est obtenu en cherchant un vecteur code y qui minimise la distance de Hamming avec le vecteur message reçu [28].

2.3.2 Algorithme de VITERBI

Le décodage du code convolutif est obtenu par la recherche d'un chemin code dans la structure en treillis vérifiant la distance minimale au sens de Hamming avec la séquence reçue.

Exemple (2-5):

Soit la séquence suivante 0100010000, reçue à partir d'un codeur de la figure (2-2). Pour cinq étapes, on obtient les chemins code suivants, dont les distances sont minimales :

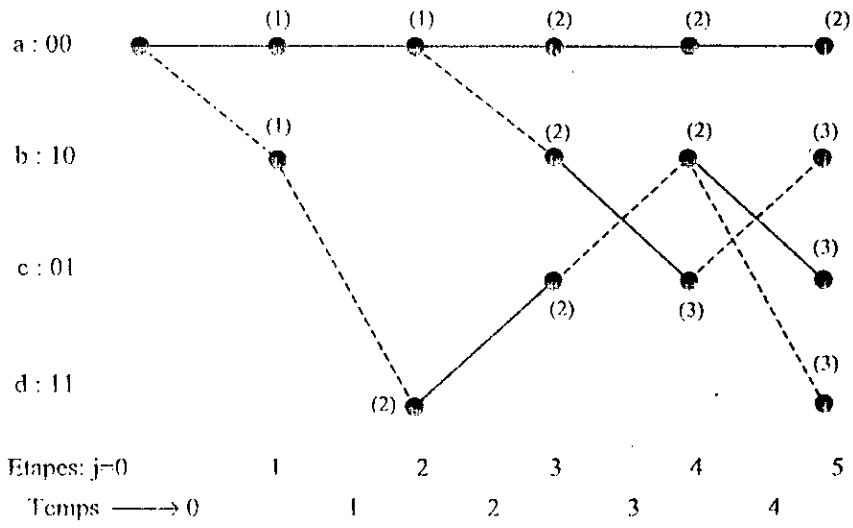


Figure (2-5): Exemple d'illustration de l'algorithme de Viterbi

La figure (2-5) illustre la progression de l'algorithme dans le temps, à t = 0, la séquence 01 arrive au récepteur. Le décodeur calcul la distance de Hamming entre la séquence reçue et les mots code correspondants aux deux branches issues de l'état 0. Après, la séquence 00 arrive. Le décodeur calcul maintenant quatre distances de branches, à partir de l'étape $j \geq 2$ le décodeur calcul huit distances de branches. Deux chemins entrent dans chaque état de cette figure, l'algorithme de Viterbi retient uniquement le chemin qui vérifie une distance cumulée minimale.

Les nombres entre parenthèses désignent les distances cumulées minimales. Pour la figure ci-dessus, on remarque bien que le chemin du plus haut, est celui qui vérifie la distance minimale avec la séquence reçue, donc on décide que le message 0000...0 est le message émis.

La distance cumulée du chemin trouvé indique le nombre d'erreurs corrigées par l'algorithme de Viterbi.

Exemple (2-6):

On suppose maintenant que la séquence reçue est 11000100 qui contient trois erreurs en comparaison avec la séquence émise tous zéros. Dans ce cas, les chemins donnants des distances minimales sont les suivants :

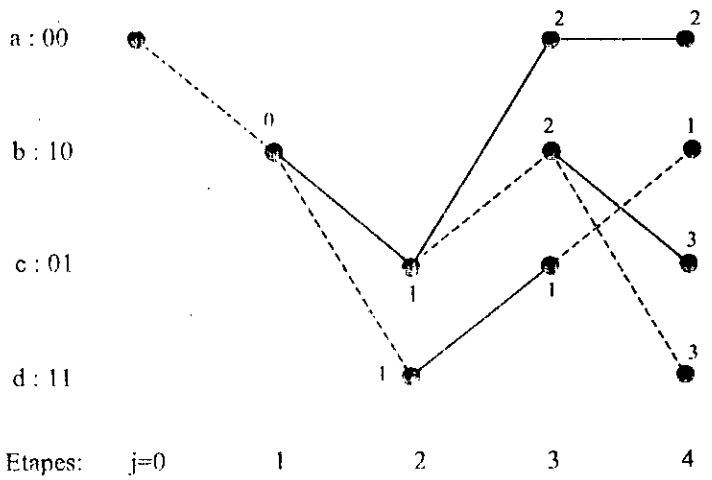


Figure (2-6): Exemple d'inefficacité de correction pour R=1/2 et M=2

La distance cumulée minimale est égale à "1" et le chemin correspondant est celui dont la sortie décodée est 11010100.

Pour cet exemple, on remarque que l'algorithme de Viterbi n'arrive pas à corriger trois erreurs pour un codeur convolutif de taux de codage R =1/2 et de mémoire M =2.

Cette raison d'inefficacité est toujours vérifiée par la formule suivante : $\mathbf{t \leq \frac{1}{2}(d_{min} - 1)}$.

Dans la partie qui suit, on va chercher la valeur de d_{min}.

2.3.3 La Propriété de la Distance dans le Codage Convolutif

La performance du codeur convolutif ne réside pas uniquement dans l'algorithme de décodage mais aussi dans la propriété de la distance du code.

On définit cette distance par d_{min} qui est la plus petite distance de Hamming entre deux mots code dans le code. Le calcul de d_{min} est obtenu par le calcul de la fonction génératrice du

codeur convolutif; la fonction génératrice, nous renseigne sur les performances du codeur. Pour calculer cette fonction génératrice, on fait appel à la section de la structure en treillis de la figure (2-4) qui sera modifier comme suit :

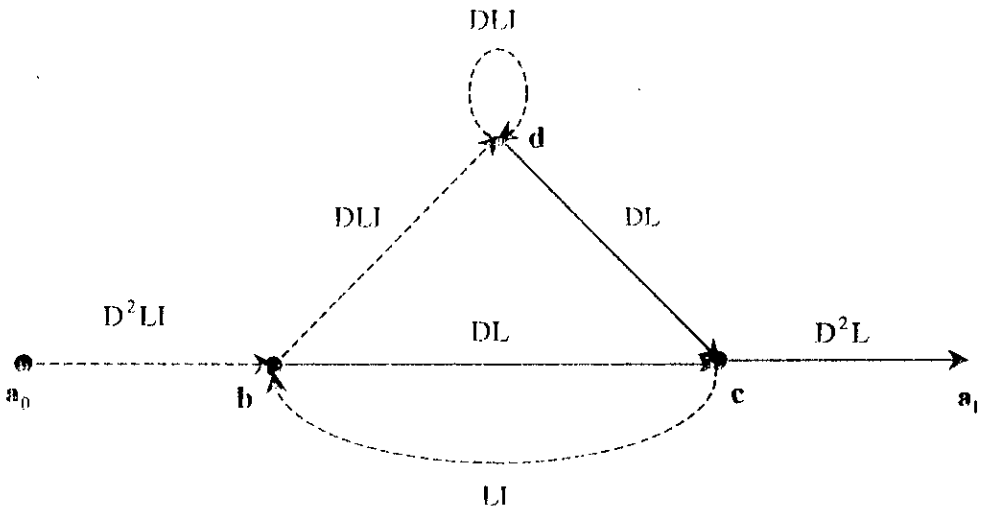


Figure (2-7): Diagramme d'états de la figure (2-4)

Les tous zéros de l'état "a" sont éclatés à l'état initial "a_0" et à l'état final " a_1 ". Les lettres des branches code représentent ce qui suit :

- La puissance de D représente la longueur de Hamming de la sortie du codeur de la branche correspondante.
- La puissance de I décrit la longueur de Hamming du message d'entrée donc : $I^0 = 1$ et $I^1 = I$.
- La puissance de L est toujours égale à 1, correspondant à la longueur de la branche.

On définit le chemin fondamental par le chemin qui mène de l'état initial a_0 à l'état final a_1. Soit maintenant $T_{d,l,i}$ le nombre de chemins de a_0 à a_1 avec l'étiquette $D^dL^lI^i$, on définit la fonction génératrice T(D,L,I) par tous les chemins qui mènent de a_0 à a_1, soit :

$$T(D,L,I) = \sum_{d=1}^{\infty} \sum_{l=1}^{\infty} \sum_{i=1}^{\infty} T_{d,l,i} D^d L^l I^i \qquad (2\text{-}28)$$

Exemple (2-7):

Pour l'exemple de la figure (2-7), on trouve :

$$T(D,L,I)=\frac{D^5L^3I}{1-DLI(1+L)} \tag{2-29}$$

En développant la quantité $\frac{1}{1-DLI(1+L)}$, on obtient alors :

$$T(D,L,I)=D^5L^3I+D^6L^4I^2(1+L)+D^7L^5I^3(1+L)^2+\cdots \tag{2-30}$$

Dans cette expression, on remarque ce qui suit :

- Il existe un seul chemin fondamental de distance 5 au chemin tous zéros, il diffère de ce dernier d'une seule entrée binaire " 1 " tout en parcourant trois branches uniquement.
- Il y'a deux chemins fondamentaux de distance 6.

Dans le codeur convolutif, la plus petite distance de Hamming est égale à la distance minimale entre le chemin fondamental et le chemin tous zéros ; pour cet exemple $d_{min}=5$, ce qui explique que pour ce codeur on ne peut pas corriger plus de deux erreurs.

2.3.4 La Limite de la Probabilité d'Erreur

Supposons un codeur convolutif avec un taux de codage $R=\frac{k}{n}$ et une large longueur du message d'entrée. L'algorithme de décodage est celui de Viterbi.

La probabilité d'erreur par bit est limitée par [28] :

$$BER \le \frac{1}{k}\left.\frac{\partial T(D,I)}{\partial I}\right|_{I=1,D=Z} \tag{2-31}$$

Le paramètre Z représente la fonction de probabilité de transition du canal. Dans cette expression de limite, on ne s'intéresse pas à la longueur du chemin; et comme la distance minimale est d_{min}, donc on peut écrire ce qui suit :

$$T(D,I)=\sum_{d=d_{min}}^{\infty}\sum_{i=1}^{\infty}T_{d,i}D^dI^i \tag{2-32}$$

d'où

$$\left.\frac{\partial T(D,I)}{\partial I}\right|_{I=1,D=Z}=\sum_{d=d_{min}}^{\infty}\sum_{i=1}^{\infty}iT_{d,i}Z^d \tag{2-33}$$

On pose :

$$A_d=\sum_{i=1}^{\infty}iT_{d,i} \tag{2-34}$$

On trouve : $$BER \le \frac{1}{k} \sum_{d=d_{min}}^{\infty} A_d Z^d \qquad (2\text{-}35)$$

Exemple (2-8) : Canal Symétrique Binaire (BSC)

La fonction de probabilité de transition d'un tel canal est donnée par : $Z = 2\sqrt{p(1-p)}$ où p est la probabilité de transition [28].

Pour une modulation de phase binaire (modulation BPSK), la probabilité d'erreur est donnée par :

$$P_e = \frac{1}{2} \operatorname{erfc}\left(\sqrt{\frac{E_c}{N_0}}\right) \qquad (2\text{-}36)$$

Pour le code convolutif dont le taux de codage est R avec : $E_c = RE_b$, d'où $P_e = \frac{1}{2} \operatorname{erfc}\left(\sqrt{\frac{RE_b}{N_0}}\right)$

Pour un canal binaire symétrique : $P_e = p$.

Le BER pour ce type de codage sera comme suit :

$$BER \le \frac{1}{k} \sum_{d=d_{min}}^{\infty} A_d Z^d = \frac{1}{k}\left[A_{d_{min}} Z^{d_{min}} + A_{d_{min}+1} Z^{d_{min}+1} + \ldots\right.$$

En remplaçant Z par son expression, on obtient :

$$BER << \frac{1}{k}\left[A_{d_{min}} 2^{d_{min}} (p(1-p))^{\frac{d_{min}}{2}} + A_{d_{min}+1} 2^{d_{min}+1} (p(1-p))^{\frac{d_{min}+1}{2}} + \ldots\right. \qquad (2\text{-}37)$$

Dans notre exemple ($R = \frac{1}{2}$ et $k = 1$), on a $A_{d_{min}} = 1$ et avec $P_e << 1$ on approxime le BER au premier terme, d'où le résultat suivant :

$$BER \approx 2^{d_{min}} \left[p(1-p)\right]^{\frac{d_{min}}{2}} \qquad (2\text{-}38)$$

On remarque bien dans cette dernière relation que le taux d'erreurs par bit est inversement proportionnel à la plus petite distance de Hamming d_{min}.

Chapitre 3

Les Turbo Codes Parallèles et Séries

Nous étudions dans ce chapitre les deux concaténations parallèle et série des codes convolutifs. Les Turbo Codes initialement découverts ont été construits par deux codes convolutifs systématiques récursifs, concaténés en parallèle et séparés par un entrelaceur [26]. Il est possible de généraliser la concaténation parallèle à plusieurs codes convolutifs séparés par plusieurs entrelaceurs. Vu le rendement très faible d'une concaténation à plus de deux niveaux, nous limitons notre étude aux turbo codes parallèles formés seulement de deux codes convolutifs constituants, ceci est justifié aussi par les excellentes performances des turbo codes à deux niveaux après un décodage itératif [7].

De la même manière, la concaténation série revient à mettre en cascade plusieurs codes convolutifs séparés par des entrelaceurs [21]. Pour les mêmes raisons citées ci-dessus, nous limitons notre étude aux turbo codes séries construits par la cascade de deux codes convolutifs séparés par un seul entrelaceur.

3.1 Le Premier Turbo Code

Soit un taux de codage R=1/2 d'un codeur convolutif possédant M mémoires. Le mot code d'un codeur classique non systématique (NSC) est représenté par le couple (X_k,Y_k) qui peut s'écrire par :

$$X_k = \sum_{i=0}^{M} g_{1i} d_{k-i} \quad \text{mod.2 avec : } g_{1i}=0,1 \tag{3-1}$$

$$Y_k = \sum_{i=0}^{M} g_{2i} d_{k-i} \quad \text{mod.2 avec : } g_{2i}=0,1 \tag{3-2}$$

$G_1=g_{1i}$ et $G_2=g_{2i}$ sont les fonctions génératrices, généralement, écrites en forme octale. Par exemple, pour $G_1=37$ et $G_2=21$, on obtient le codeur suivant :

Avec: $\begin{cases} G_1=(1,1,1,1,1) \\ G_2=(1,0,0,0,1) \end{cases}$ en forme binaire

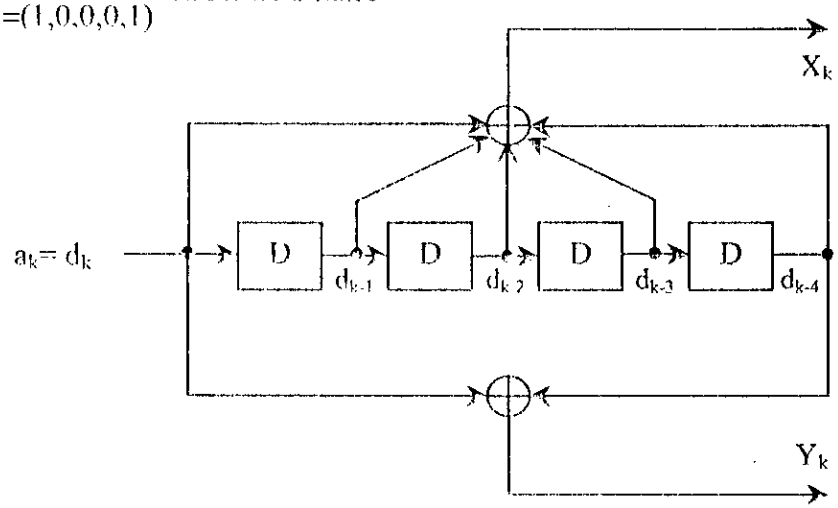


Figure (3-1) : Codeur convolutif non systématique et non récursif NRNSC

Le code systématique récursif d'un taux R=1/2 est obtenu à partir du code classique non systématique en utilisant la boucle de contre réaction des sorties X_k ou Y_k avec l'entrée d_k et en posant l'une de ses sorties égale à l'entrée d_k. Si par exemple $X_k=d_k$, la sortie Y_k est égale à l'équation (3-2) en substituant d_k par a_k et la variable a_k est récursivement calculée par :

$$a_k = d_k + \sum_{i=1}^{M} g_{1i} a_{k-i} \quad \text{mod. 2} \tag{3-3}$$

Un des codeurs systématique récursif (RSC) de mémoire M=4 obtenu à partir du codeur de la figure (3-1) est schématisé en figure (3-2).

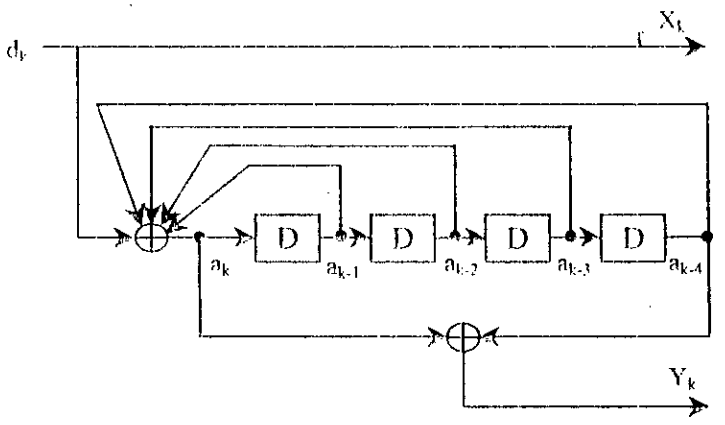


Figure (3-2): Codeur convolutif systématique et récursif RSC

3.1.1 Concaténation Parallèle des Codes RSC

Dans cette concaténation, on considère deux codes RSC identiques mis en parallèle. Pour le premier codeur (C1) son entrée est simplement d_k; par contre, le deuxième codeur (C2) son entrée est d_k passée dans un entrelaceur (Interleaver).

La sortie du codeur est le couple suivant (X_k, Y_{1k}) ou (X_k, Y_{2k}) selon la position du multiplexeur comme le montre la figure ci-dessous :

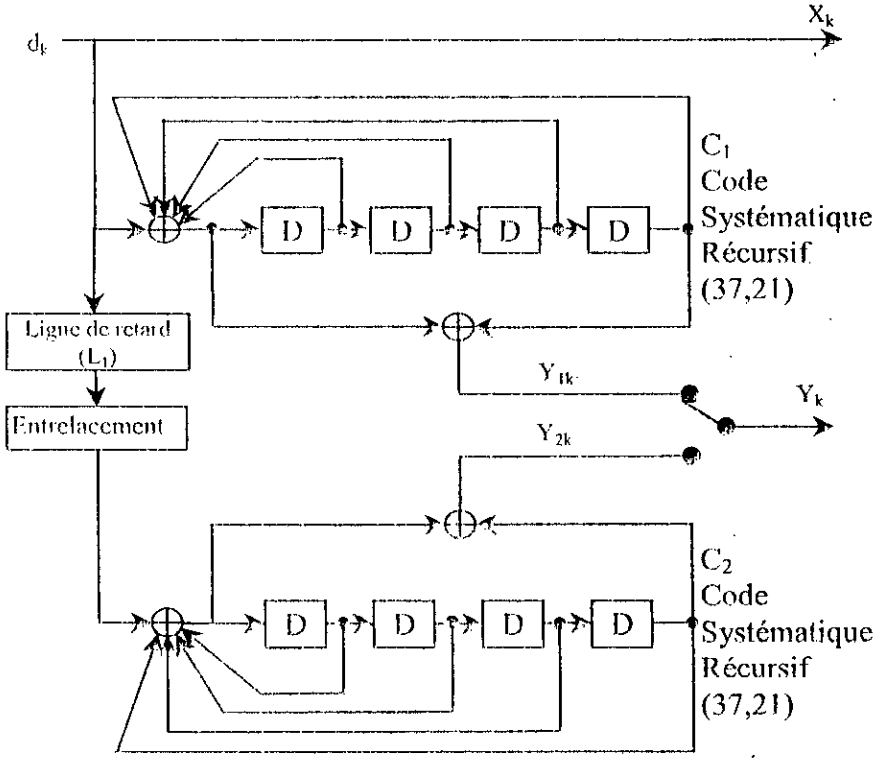


Figure (3-3) : Concaténation parallèle des codes systématiques récursifs

Le décodeur DEC représenté dans la figure (3-4) est composé de deux décodeurs élémentaires DEC1 et DEC2 mis en cascade. Les erreurs de sortie du décodeur DEC1 sont dispersées par l'entrelaceur et le délai du codage L_1 est pris en compte par le décodeur DEC1 .

Pour un canal gaussien discret sans mémoire et une modulation de phase binaire (BPSK), l'entrée R_k du décodeur DEC est composée de deux variables aléatoires x_k et y_k, qui peuvent s'écrire par les équations suivantes:

$$x_k = \pm 1 + i_k = (2d_k - 1) + i_k \tag{3-4}$$

$$y_k = \pm 1 + q_k = (2Y_k - 1) + q_k \tag{3-5}$$

Avec i_k et q_k deux bruits gaussiens, indépendants, de moyennes zéro et de variances $\sigma^2 = N_0 / 2E_c$. N_0 est une constante et $E_c = RE_b$ désigne l'énergie moyenne par bit codé.

L'information redondante y_k est demultiplexée et est envoyée vers le décodeur DEC1 lorsque $Y_k = Y_{1k}$ et vers le décodeur DEC2 lorsque $Y_k = Y_{2k}$ comme le montre la figure (3-4).

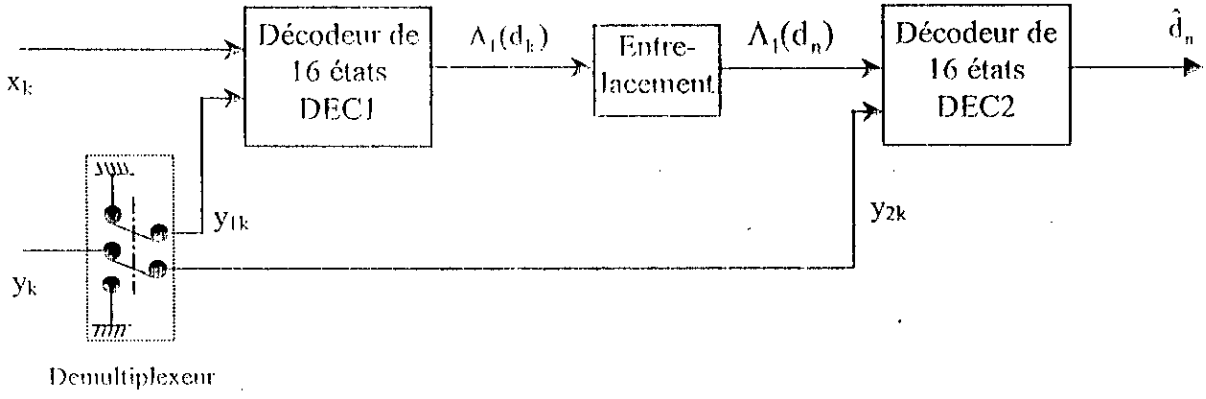


Figure (3-4) : Schéma de principe du décodeur

Le premier décodeur nous fournit la quantité suivante :

$$\Lambda_1(d_k) = \log \frac{P\{d_k = 1 / \text{observation}\}}{P\{d_k = 0 / \text{observation}\}} \tag{3-6}$$

La quantité $\Lambda_i(d_k)$ est appelée le logarithme du rapport de vraisemblance LLR (Logarithm of Likelihood Ratio) où la probabilité $P\{d_k = i /observation\}$, i=0,1; est la probabilité a posteriori de l'information d_k.

3.1.1.1 Procédé d'Entrelacement

Le but du procédé d'entrelacement (en anglais : interleaving) est de permettre l'utilisation des codes destinés à corriger des erreurs indépendantes dans le cas des canaux à paquets d'erreurs.

Le type d'entrelacement est un facteur important dans les performances des turbo codes [6]. On donne deux exemples d'entrelaceurs qui sont utilisés dans ce travail.

3.1.1.1.1 Entrelaceur Ligne-Colonne (Uniforme)

L'entrelaceur le plus simple est une mémoire dont les bits sont écrits ligne par ligne et lus colonne par colonne. Par exemple les bits sont écrits comme le montre le tableau suivant:

b_1	b_2	b_3	b_4
b_5	b_6	b_7	b_8
b_9	b_{10}	b_{11}	b_{12}
b_{13}	b_{14}	b_{15}	b_{16}

Tableau (3-1) : Ecriture des bits ligne par ligne dans une mémoire pour une taille N=16

Le procédé d'entrelacement consiste en la lecture des bits colonne par colonne comme suit :

b_1	b_5	b_9	b_{13}	b_2	b_6	b_{10}	b_{14}	b_3	b_7	b_{11}	b_{15}	b_4	b_8	b_{12}	b_{16}

Tableau (3-2) : Lecture des bits colonne par colonne de la mémoire

3.1.1.1.2 Entrelaceur Pseudo-Aléatoire

L'entrelaceur pseudo-aléatoire est défini par la génération de nombres entiers de 1 à N pseudo-aléatoirement, N étant la taille de la séquence des bits d'information.

b_{14}	b_6	b_{11}	b_2	b_{12}	b_{15}	b_1	b_{13}	b_7	b_5	b_{16}	b_8	b_4	b_{10}	b_3	b_9

Tableau (3-3) : Génération de positions des bits pseudo-aléatoirement

3.1.2 Modification de l'Algorithme de BCJR[31] pour le Code RSC

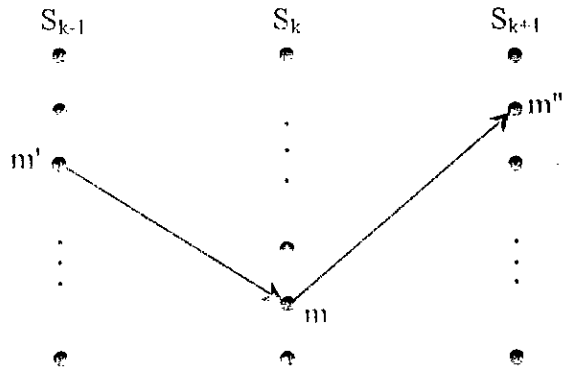
Considérons un code RSC avec M mémoires, à tout instant k, le codeur peut être représenté par son état qui est le suivant : $S_k=(a_k,a_{k-1}, \ldots,a_{k-M})$ (3-7)

Soit maintenant N bits indépendants entrant dans le codeur, prenant les valeurs 0 et 1 avec une même probabilité. En supposant que l'état initial S_0 et l'état final S_N du codeur sont égaux à zéro :

$$S_0=S_N=(0,0,\ldots,0) = \mathbf{0} \quad (3\text{-}8)$$

Comme le montre la figure (3-5), les bits d'information $d_1^N = \{d_1,d_2,...,d_k,...,d_N\}$ causent la transition du treillis de S_{k-1} à S_k, le mot code de sortie correspondant $C_1^N = \{C_1,C_2,...,C_k,...,C_N\}$, devient l'entrée du canal gaussien discret sans mémoire dont la séquence de sortie est de la forme $R_1^N = \{R_1,R_2,...,R_k,...,R_N\}$.

Où $R_k=(x_k,y_k)$ est défini par les équations (3-4) et (3-5).



Bits d'information	d_k	d_{k+1}
Mots code de sortie	C_k	C_{k+1}
Signaux reçus	R_k	R_{k+1}

Figure (3-5): Les variables de l'algorithme de BCJR modifié

La probabilité a posteriori du décodage de la donnée d_k peut être calculée à partir de la probabilité conjointe $\lambda_k^i(m)$ définie par : $\lambda_k^i(m)=P\{d_k=i,S_k=m/R_1^N\}$ avec S_k est l'état du codeur à l'instant k, d'où la probabilité a posteriori du décodage de la donnée d_k est égale à :

$$P\{d_k=i/R_1^N\}=\sum_m \lambda_k^i(m),\ i=0,1 \tag{3-9}$$

Le logarithme du rapport de vraisemblance s'écrit comme suit :

$$\Lambda(d_k)=\log\frac{\sum_m \lambda_k^1(m)}{\sum_m \lambda_k^0(m)} \tag{3-10}$$

Finalement, en comparant la quantité $\Lambda(d_k)$ avec le seuil zéro, on décide sur d_k :

$$\begin{aligned}\hat{d}_k&=1 \quad si \quad \Lambda(d_k)\geq 0\\ \hat{d}_k&=0 \quad si \quad \Lambda(d_k)<0\end{aligned} \tag{3-11}$$

Pour calculer la probabilité $\lambda_k^i(m)$, on introduit les fonctions de probabilités suivantes :

$\alpha_k^i(m)$, $\beta_k(m)$ et $\gamma_i(R_k,m',m)$.

Avec :

$$\alpha_k^i(m)=\frac{P\{d_k=i,S_k=m,R_1^k\}}{P\{R_1^k\}}=P\{d_k=i,S_k=m/R_1^k\} \tag{3-12}$$

$$\beta_k(m)=\frac{P\{R_{k+1}^N/S_k=m\}}{P\{R_{k+1}^N/R_1^k\}} \tag{3-13}$$

$$\gamma_i(R_k,m',m)=P\{d_k=i,R_k,S_k=m/S_{k-1}=m'\} \tag{3-14}$$

La probabilité conjointe $\lambda_k^i(m)$ peut être réécrite en :

$$\lambda_k^i(m)=\frac{P\{d_k=i,S_k=m,R_1^k,R_{k+1}^N\}}{P\{R_1^k,R_{k+1}^N\}}$$

On obtient : $$\lambda_k^i(m)=\frac{P\{d_k=i,S_k=m,R_1^k\}}{P\{R_1^k\}}\times\frac{P\{R_{k+1}^N/d_k=i,S_k=m,R_1^k\}}{P\{R_{k+1}^N/R_1^k\}}$$

D'où : $$\lambda_k^i(m)=\alpha_k^i(m).\beta_k(m) \tag{3-15}$$

3.1.2.1 Evaluation des Probabilités $\alpha_k^i(m)$ et $\beta_k(m)$

Dans l'expression (3-12), $\alpha_k^i(m)$ est égale à :

$$\alpha_k^i(m)=\frac{P\{d_k=i,S_k=m,R_1^{k-1},R_k\}}{P\{R_1^{k-1},R_k\}}=\frac{P\{d_k=i,S_k=m,R_k/R_1^{k-1}\}}{P\{R_k/R_1^{k-1}\}} \tag{3-16}$$

Et comme $P\{d_k=i,S_k=m,R_k/R_1^{k-1}\}=\sum_{m'}\sum_{j=0}^{1}P\{d_k=i,d_{k-1}=j,S_k=m,S_{k-1}=m',R_k/R_1^{k-1}\}$ (3-17)

$$=\sum_{m'}\sum_{j=0}^{1}\frac{P\{d_{k-1}=j,S_{k-1}=m',R_1^{k-1}\}}{P\{R_1^{k-1}\}}\cdot P\{d_k=i,S_k=m,R_k/d_{k-1}=j,S_{k-1}=m',R_1^{k-1}\} \tag{3-18}$$

En tenant compte de l'expression de $\alpha_k^i(m)$ et de $\gamma_i(R_k,m',m)$, on obtient :

$$P\{d_k=i,S_k=m,R_k/R_1^{k-1}\}=\sum_{m'}\sum_{j=0}^{1}\gamma_i(R_k,m',m)\cdot\alpha_{k-1}^j(m') \tag{3-19}$$

Le dénominateur de l'expression (3-16) peut être exprimé par :

$$P\{R_k/R_1^{k-1}\}=\sum_{m}\sum_{i=0}^{1}P\{d_k=i,S_k=m,R_k/R_1^{k-1}\} \tag{3-20}$$

En remplaçant $P\{d_k=i,S_k=m,R_k/R_1^{k-1}\}$ par son expression, on obtient :

$$P\{R_k/R_1^{k-1}\}=\sum_{m}\sum_{m'}\sum_{i=0}^{1}\sum_{j=0}^{1}\gamma_i(R_k,m',m)\cdot\alpha_{k-1}^j(m') \tag{3-21}$$

Finalement, $\alpha_k^i(m)$ peut être exprimée par :

$$\alpha_k^i(m)=\frac{\sum_{m'}\sum_{j=0}^{1}\gamma_i(R_k,m',m)\cdot\alpha_{k-1}^j(m')}{\sum_{m}\sum_{m'}\sum_{i=0}^{1}\sum_{j=0}^{1}\gamma_i(R_k,m',m)\cdot\alpha_{k-1}^j(m')} \tag{3-22}$$

De la même façon, on calcul $\beta_k(m)$:

$$\beta_k(m)=\frac{P\{R_{k+1}^N/S_k=m\}}{P\{R_{k+1}^N/R_1^k\}}=\frac{\sum_{m''}\sum_{i=0}^{1}P\{d_{k+1}=i,S_{k+1}=m'',R_{k+2}^N,R_{k+1}/S_k=m\}}{P\{R_{k+1}^N/R_1^k\}} \tag{3-23}$$

Et le numérateur de $\beta_k(m)$ devient :

$$P\{R_{k+1}^N/S_k=m\}=\sum_{m''}\sum_{i=0}^{1}P\{R_{k+2}^N/S_{k+1}=m''\}\cdot P\{d_{k+1}=i,S_{k+1}=m'',R_{k+1}/S_k=m\} \tag{3-24}$$

Et comme: $$P\{R_{k+1}^N/R_1^k\}=P\{R_{k+1}/R_1^k\}\cdot P\{R_{k+2}^N/R_1^{k+1}\} \tag{3-25}$$

D'où : $$\beta_k(m)=\frac{\sum_{m''}\sum_{i=0}^{1}\gamma_i(R_{k+1},m,m'')\cdot\beta_{k+1}(m'')}{\Pr\{R_{k+1}/R_1^k\}} \quad (3\text{-}26)$$

Le dénominateur s'écrit comme dans l'expression (3-21) par :

$$\Pr\{R_{k+1}/R_1^k\}=\sum_{m}\sum_{m''}\sum_{i=0}^{1}\sum_{j=0}^{1}\gamma_i(R_{k+1},m,m'')\cdot\alpha_k^j(m) \quad (3\text{-}27)$$

Finalement, on obtient :

$$\beta_k(m)=\frac{\sum_{m''}\sum_{i=0}^{1}\gamma_i(R_{k+1},m,m'')\cdot\beta_{k+1}(m'')}{\sum_{m}\sum_{m''}\sum_{i=0}^{1}\sum_{j=0}^{1}\gamma_i(R_{k+1},m,m'')\cdot\alpha_k^j(m)} \quad (3\text{-}28)$$

3.1.2.2 Evaluation de la Probabilité $\gamma_i(R_k,m',m)$

La probabilité $\gamma_i(R_k,m',m)$ peut être déterminée par les probabilités de transition du canal gaussien discret sans mémoire et par les probabilités de transition de treillis du codeur.

Soit : $\gamma_i(R_k,m',m)=P\{d_k=i,S_k=m,R_k/S_{k-1}=m'\}$

$=P\{R_k/d_k=i,S_k=m,S_{k-1}=m'\}\cdot P\{d_k=i,S_k=m/S_{k-1}=m'\}$

$=P\{R_k/d_k=i,S_k=m,S_{k-1}=m'\}\cdot P\{d_k=i/S_k=m,S_{k-1}=m'\}\cdot P\{S_k=m/S_{k-1}=m'\}$ (3-29)

et comme $R_k=(x_k,y_k)$ alors :

- $$\begin{cases}P\{R_k/d_k=i,S_k=m,S_{k-1}=m'\}\\ \quad=P\{x_k/d_k=i,S_k=m,S_{k-1}=m'\}\cdot P\{y_k/d_k=i,S_k=m,S_{k-1}=m'\}\end{cases} \quad (3\text{-}30)$$

Avec :

- $P\{d_k=i/S_k=m,S_{k-1}=m'\}$:est égale à 0 ou à 1. (3-31)
- $P\{S_k=m/S_{k-1}=m'\}=\frac{1}{2}$ Parce que le codeur ne possède que deux possibilités de transition

à savoir : $P\{d_k=1\}=P\{d_k=0\}=\frac{1}{2}$ (3-32)

En conclusion :

$$\gamma_i(R_k,m',m)=\begin{cases}\frac{1}{2}P\{R_k/d_k=i,S_k=m,S_{k-1}=m'\} & \text{si l'état du codeur passe de } m' \rightarrow m \text{ par une entrée } d_k=i \\ 0 & \text{ailleurs}\end{cases} \quad (3\text{-}33)$$

Les différentes étapes de l'algorithme de BCJR modifié sont les suivantes :

- **Etape 0 :** Les probabilités $\alpha_0^i(m)$ et $\beta_N(m)$ vont être initialisées comme suit :

$$\begin{aligned}&\alpha_0^i(0)=1,\ \alpha_0^i(m)=0\ \ \forall m\neq 0,\ \ i=0,1\\&\beta_N(0)=1,\ \beta_N(m)=0\ \ \forall m\neq 0\end{aligned} \quad (3\text{-}34)$$

- **Etape 1 :** Pour chaque observation R_k, on calcul les probabilités $\alpha_k^i(m)$ et $\gamma_i(R_k,m',m)$ respectivement par leurs expressions (3-22) et (3-33).

- **Etape 2 :** Quand la séquence R_1^N sera complètement reçue, la probabilité $\beta_k(m)$ sera calculée par son expression (3-28) et les probabilités $\alpha_k^i(m)$ et $\beta_k(m)$ seront multipliées pour obtenir $\lambda_k^i(m)$.

3.1.3 L'Information Extrinsèque du Codeur RSC

On a cité dans la partie précédente que le logarithme du rapport de vraisemblance est donné par :

$$\Lambda(d_k)=\log\frac{\sum_m \lambda_k^1(m)}{\sum_m \lambda_k^0(m)}=\log\frac{\sum_m \alpha_k^1(m)\beta_k(m)}{\sum_m \alpha_k^0(m)\beta_k(m)} \quad (3\text{-}35)$$

En remplaçant $\alpha_k^i(m)$, $(i=0,1)$, par son expression dans l'équation (3-35), on obtient alors :

$$\Lambda(d_k)=\log\frac{\sum_m\sum_{m'}\sum_{j=0}^{1}\gamma_1(R_k,m',m)\alpha_{k-1}^j(m')\cdot\beta_k(m)}{\sum_m\sum_{m'}\sum_{j=0}^{1}\gamma_0(R_k,m',m)\alpha_{k-1}^j(m')\cdot\beta_k(m)} \quad (3\text{-}36)$$

Le fait que le codeur soit systématique ($x_k=d_k$), la probabilité de transition $P\{x_k/d_k=i,S_k=m,S_{k-1}=m'\}$ dans l'expression de $\gamma_i(R_k,m',m)$ est indépendante des états de S_k et de S_{k-1}, d'où :

$$\gamma_1(R_k,m',m)=\frac{1}{2}P\{x_k/d_k=1\}\cdot P\{y_k/d_k=1,S_k=m,S_{k-1}=m'\}\cdot P\{d_k=1/S_k=m,S_{k-1}=m'\}$$
$$\gamma_0(R_k,m',m)=\frac{1}{2}P\{x_k/d_k=0\}\cdot P\{y_k/d_k=0,S_k=m,S_{k-1}=m'\}\cdot P\{d_k=0/S_k=m,S_{k-1}=m'\} \quad (3\text{-}37)$$

Conditionnellement à $d_k=1$ (respectivement, $d_k=0$), la variable gaussienne x_k est de moyenne 1 (respectivement, -1) et de variance σ^2. Ce qui donne en utilisant la densité de probabilité de x_k :

$$\Lambda(d_k)=\frac{2}{\sigma^2}x_k+W_k \quad (3\text{-}38)$$

Avec :

$$W_k=\Lambda(d_k)\Big|_{x_k=0}=\log\frac{\sum_m\sum_{m'}\sum_{j=0}^{1}\gamma_1(y_k,m',m)\alpha_{k-1}^{j}(m')\cdot\beta_k(m)}{\sum_m\sum_{m'}\sum_{j=0}^{1}\gamma_0(y_k,m',m)\alpha_{k-1}^{j}(m')\cdot\beta_k(m)} \quad (3\text{-}39)$$

On appel la quantité W_k l'information extrinsèque, et qui ne dépend plus de l'entrée x_k du décodeur. Cette propriété sera utilisée pour le décodage des deux codeurs parallèles concaténés [26].

3.1.4 Plan de Décodage de la Concaténation Parallèle de deux codeurs RSC

On a déjà décrit dans la figure (3-4) le schéma de base du décodeur global; maintenant, on donne les détails de chaque décodeur comme le montre la figure (3-6).

La fonction $\Lambda_1(d_k)$ est donnée par : $\Lambda_1(d_k)=\frac{2}{\sigma^2}x_k+W_{1k}$ (3-40)

De la même façon, le décodeur DEC2 possède comme entrées les variables $\overline{\Lambda}_1(d_n)$ et y_{2k} qui sont indépendantes, ce qui donne : $\Lambda_2(d_k)=\Gamma(\overline{\Lambda}_1(d_k))+W_{2k}$ (3-41)

La variable W_{2k} est faiblement corrélée avec x_k et y_{1k}, ceci est du à la présence de l'entrelacement entre les deux décodeurs DEC1 et DEC2, d'où l'information W_{2k} et les deux observations x_k et y_{1k} seront conjointement utilisées de nouveau dans le décodeur DEC1 pour le calcul de $\Lambda_1(d_k)$ [26].

Dans la figure (3-6), on présente le schéma du nouveau décodeur qui utilise l'information extrinsèque $z_k = W_{2k}$, générée par le décodeur DEC2 en boucle de contre-réaction.

Dans ce nouveau décodeur, DEC1 possède trois entrées (x_k, y_{1k}, z_k) et les probabilités $\alpha_{1k}^{i}(m)$ et $\beta_{1k}(m)$ seront calculées en substituant $R_k = (x_k, y_{1k})$ par $R_k = (x_k, y_{1k}, z_k)$ dans les relations (3-22) et (3-28).

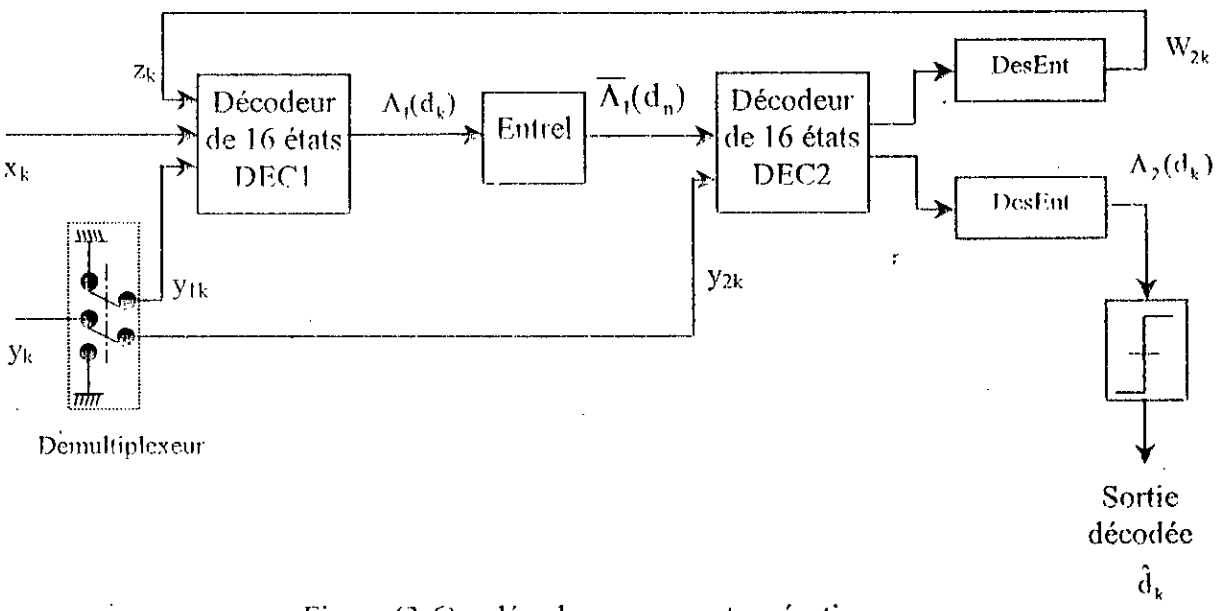


Figure (3-6) : décodeur avec contre-réaction

En tenant compte que z_k est faiblement corrélée avec x_k et y_{1k} et en supposant que z_k peut être approximée par une variable gaussienne de variance $\sigma_z^2 \neq \sigma^2$[26]. Alors, la probabilité de transition du canal gaussien discret et sans mémoire peut être maintenant factorisée en trois termes :

$$\begin{aligned} P\{R_k / d_k = i, S_k = m, S_{k-1} = m'\} = & P\{x_k / d_k = i, S_k = m, S_{k-1} = m'\} \\ & \times P\{y_k / d_k = i, S_k = m, S_{k-1} = m'\} \\ & \times P\{z_k / d_k = i, S_k = m, S_{k-1} = m'\} \end{aligned} \qquad (3\text{-}42)$$

Le logarithme du rapport de vraisemblance (LRV) $\Lambda_1(d_k)$ généré par le décodeur DEC1 s'écrit maintenant :

$$\Lambda_1(d_k) = \frac{2}{\sigma^2} x_k + \frac{2}{\sigma_z^2} z_k + W_{1k} \qquad (3\text{-}43)$$

L'information z_k n'est pas utilisée à l'entrée du décodeur DEC2 [26], les entrées du décodeur DEC2 sont les séquences $\bar{\Lambda}_1(d_n)$ et y_{2k} avec :

$$\bar{\Lambda}_1(d_n) = \Lambda_1(d_n)\big|_{z_n=0} \qquad (3\text{-}44)$$

Finalement, de la relation (3-41), l'information extrinsèque $z_k = W_{2k}$ après desentrelacement, peut être écrite par :

$$z_k = W_{2k} = \Lambda_2(d_k)\big|_{\bar{\Lambda}_1(d_k)=0} \qquad (3\text{-}45)$$

La décision sur la sortie du décodeur (DEC=DEC1+DEC2) s'écrit alors :

$$\hat{d}_k = \text{sign}\left[\Lambda_2(d_k)\right] \qquad (3\text{-}46)$$

Pour la construction du décodeur global, on fait appel au processus itératif présenté dans la figure ci-dessous, qui est composé de "q" décodeurs élémentaires identiques mis en cascade.

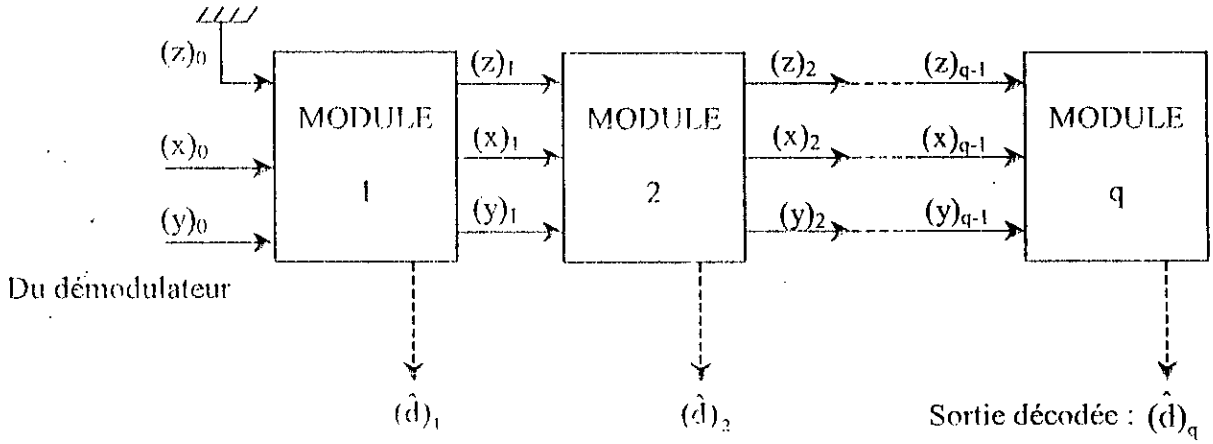


Figure (3-7) : Processus itératif du décodeur par contre-réaction

3.2 Structure des Turbo Codes

Les concaténations les plus célèbres sont la concaténation parallèle et la concaténation série de deux codes convolutifs. Il est possible de fabriquer des turbo codes hybrides en combinant le série et le parallèle[20].

3.2.1 Les Turbo Codes Parallèles

Un turbo code parallèle C_p est un code (J, N) linéaire. Le code C_p est construit en concaténant de manière parallèle selon le schéma de la figure (3-8) deux codes convolutifs C_1 et C_2.

Le treillis des deux codes démarre de l'état zéro et se termine après N/k_i+M_i branches dans l'état zéro, avec $R_i=k_i/n_i$ est le rendement de C_i et M_i est le nombre de mémoires du registre à décalage du code C_i. Chaque code convolutif C_i nécessite M_i branches dans la phase de terminaison du treillis.

En pratique, on prend souvent $C_1=C_2$ [20]. D'après la figure (3-8), les N bits d'information d sont codés par le codeur de C_1, entrelacés à travers Π et ensuite codés par C_2. Chaque code RSC C_i génère $(n_i-k_i)N/k_i$ bits de parité notés Y_i. Les bits d'information $d=X_1$ et X_2 étant les mêmes à une permutation près, nous ne transmettons pas l'entrée du code C_2 afin d'augmenter le rendement de C_p.

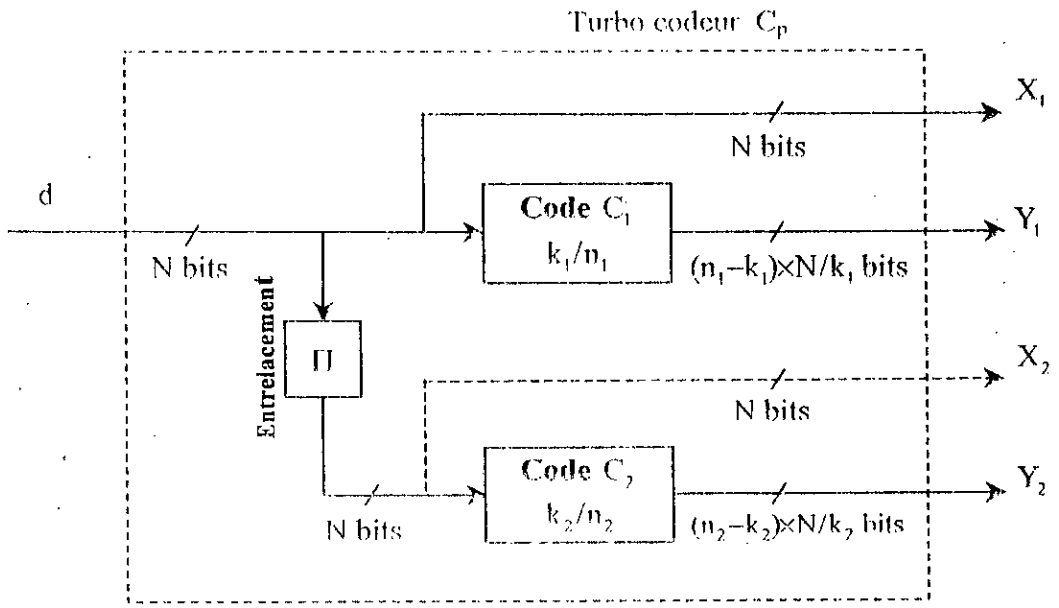


Figure (3-8) : Turbo codeur parallèle

En négligeant la fermeture des deux treillis (M_i : nombre de branches pour ramener l'état du codeur C_i à zéro), le rendement R de C_p s'obtient par l'expression simple

$$R = \frac{N}{J} = \frac{R_1 R_2}{R_1 + R_2 - R_1 R_2} = \frac{R_1}{2 - R_1} \text{ si } R_1 = R_2 \quad (3\text{-}47)$$

En tenant compte de la fermeture des deux treillis, le rendement du code C_p devient

$$R = \frac{N}{J + \frac{M_1}{R_1} + \frac{M_2}{R_2}} = \frac{R_1 R_2}{R_1 + R_2 - R_1 R_2 + \frac{R_2 M_1 + R_1 M_2}{N}}$$

$$R = \frac{R_1}{2 - R_1 + \frac{2M_1}{N}} \text{ si } R_1 = R_2 \quad (3\text{-}48)$$

Ainsi, le cas le plus fréquent correspond à $R = \frac{1}{3}$ avec $R_1 = \frac{1}{2}$ et en perforant un bit de parité sur deux ($R_1 = R_2 = \frac{2}{3}$ en perforant les bits de parité Y_1 aux instants pairs et les bits Y_2 aux instants impairs) nous avons $R = \frac{1}{2}$. Il est préférable de ne pas perforer les bits d'information afin de garantir une bonne convergence du décodage itératif [20].

3.2.2 Les Turbo Codes Séries

Un turbo code série C_s est un code en blocs (J, K) linéaire. Le code C_s est construit en concaténant de manière série selon le schéma de la figure (3-9) deux codes convolutifs C_1 et C_2. C_1 est dit code externe et C_2 code interne. Comme pour le code C_p, le treillis des deux codes constituants C_s démarre de l'état zéro et se termine à l'état zéro, après N/k_1+M_1 branches pour C_1 et après N/k_2+M_2 branches pour C_2.

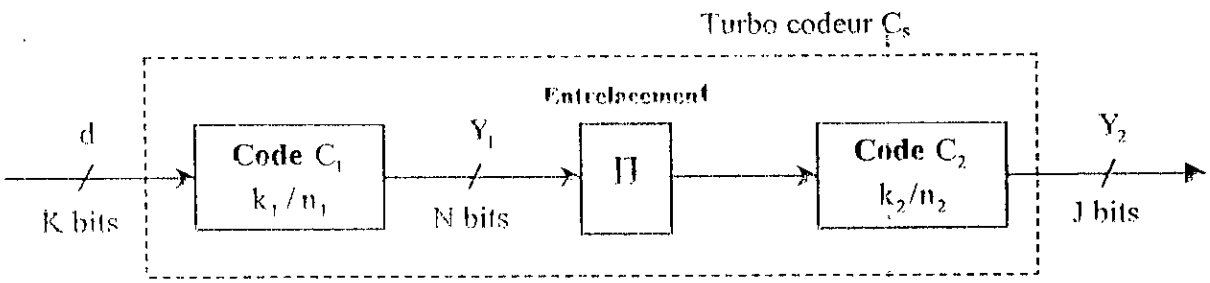


Figure (3-9): Turbo codeur série

En pratique, on prend un code externe non systématique et non récursif (NRNSN) ou un RSC, mais le code C_2 doit toujours être un RSC[7]. D'après la figure (3-9), les k bits d'information d sont codés par le codeur C_1, entrelacés à travers Π et ensuite recodés par C_2. Le code externe génère $N=K/R_1$ bits codés Y_1. Les bits Y_1 contiennent les bits d'information X_1 si C_1 est un RSC. Le code interne RSC génère $J=N/R_2$ bits codés Y_2 contenant ses bits d'information X_2. En négligeant la fermeture des deux treillis, le rendement de C_s s'obtient par un simple produit

$$R = \frac{K}{J} = R_1 \times R_2 \tag{3-49}$$

En tenant compte de la fermeture des deux treillis, le rendement du code C_s devient

$$R = \frac{K}{J + \frac{M_1}{R_1} + \frac{M_2}{R_2}} = \frac{R_1 R_2}{1 + \frac{R_2 M_1 + R_1 M_2}{K}} \tag{3-50}$$

Ainsi, le cas le plus fréquent correspond à $R = \frac{1}{2}$ avec $R_1 = \frac{2}{3}$ et $R_2 = \frac{3}{4}$. Nous remarquons déjà que les codes constituants d'un turbo code série ont un rendement plus élevés que les codes constituants d'un turbo code parallèle.

3.3 Le Décodage Itératif des Turbo Codes

Les turbo codes sont décodés par une technique itérative (dite turbo decoding) qui fait appel à un décodage MAP (maximum a posteriori) des codes constituants. Le décodage itératif fonctionne au niveau des bits d'information et non au niveau des mots code. Son but est l'estimation de l'APP (probabilité a posteriori) de chaque bit d'information.

3.3.1 Le Décodeur MAP

Dans le premier turbo code [26], on a montré que les informations extrinsèques W_{1k} et W_{2k} sont utilisées comme des variables aléatoires à l'entrée des deux décodeurs respectivement DEC2 et DEC1. Dans cette partie on donnera une autre forme de calcul des

APP en utilisant le logarithme du rapport de vraisemblance dit LLR [16][9], où les informations W_{1k} et W_{2k} seront utilisées comme des probabilités a priori, sur les bits d'information.

Le LLR est égal au rapport $\log\left(\frac{p(1)}{p(0)}\right)$ où p(b) est une observation, une APP ou une probabilité a priori associée au bit b.

Dans l'équation (3-36), nous avons la relation:

$$\Lambda(d_k)=\log\frac{\sum_m\sum_{m'}\sum_{j=0}^{1}\gamma_1(R_k,m',m)\alpha_{k-1}^{j}(m')\cdot\beta_k(m)}{\sum_m\sum_{m'}\sum_{j=0}^{1}\gamma_0(R_k,m',m)\alpha_{k-1}^{j}(m')\cdot\beta_k(m)}$$

Avec:

$$\gamma_i(R_k,m',m)=P\{R_k/d_k=i,S_k=m,S_{k-1}=m'\}\cdot P\{d_k=i,S_k=m/S_{k-1}=m'\}$$
$$=P\{R_k/d_k=i,S_k=m,S_{k-1}=m'\}\cdot P\{d_k=i/S_k=m,S_{k-1}=m'\}\cdot P\{S_k=m/S_{k-1}=m'\}$$

On a aussi :

$$P\{d_k=i,S_k=m/S_{k-1}=m'\}=P\{d_k=i/S_{k-1}=m'\}\cdot P\{S_k=m/d_k=i,S_{k-1}=m'\} \quad (3\text{-}51)$$

Et comme le bit d_k ne dépend pas de l'état S_{k-1} et

$$P\{S_k=m/d_k=i,S_{k-1}=m'\}=P\{d_k=i/S_k=m,S_{k-1}=m'\} \quad (3\text{-}52)$$

Alors, pour un code RSC, on a

$$\gamma_i(R_k,m',m)$$
$$=P\{x_k/d_k=i\}\cdot P\{y_k/d_k=i,S_k=m,S_{k-1}=m'\}\cdot P\{d_k=i\}\cdot P\{d_k=i/S_k=m,S_{k-1}=m'\} \quad (3\text{-}53)$$

Et pour un canal AWGN, on trouve

$$\Lambda(d_k)=\frac{2}{\sigma^2}x_k+\log\left(\frac{P\{d_k=1\}}{P\{d_k=0\}}\right)+\log\frac{\sum_m\sum_{m'}\sum_{j=0}^{1}\gamma_i^e(y_k,m',m)\alpha_{k-1}^{j}(m')\cdot\beta_k(m)}{\sum_m\sum_{m'}\sum_{j=0}^{1}\gamma_0^e(y_k,m',m)\alpha_{k-1}^{j}(m')\cdot\beta_k(m)} \quad (3\text{-}54)$$

Avec

$$\gamma_i^e(y_k,m',m)=P\{y_k/d_k=i,S_k=m,S_{k-1}=m'\}\cdot P\{d_k=i/S_k=m,S_{k-1}=m'\} \quad (3\text{-}55)$$

On pose $$L(d_k) = \log \frac{P\{d_k = 1\}}{P\{d_k = 0\}} \quad (3\text{-}56)$$

Ce qui donne

$$P(d_k = 1) = \frac{e^{L(d_k)}}{1+e^{L(d_k)}} = \frac{1}{1+e^{-L(d_k)}} \quad \text{et} \quad P(d_k = 0) = \frac{e^{-L(d_k)}}{1+e^{-L(d_k)}} \quad (3\text{-}57)$$

D'où $$P(d_k) = \frac{e^{-L(d_k)/2}}{1+e^{-L(d_k)}} e^{(2d_k - 1)\cdot L(d_k)/2}$$

$$= A_k \cdot e^{(2d_k - 1)\cdot L(d_k)/2} \quad (3\text{-}58)$$

On trouve aussi

$$P(x_k / d_k = i) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-1}{2\sigma^2}(x_k - (2d_k - 1))^2} = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-1}{2\sigma^2}(x_k^2 + 1)} e^{\frac{1}{\sigma^2}(2d_k - 1)x_k}$$

$$= B_k \cdot e^{(2d_k - 1)\cdot L_c x_k/2} \quad (3\text{-}59)$$

Et

$$P(y_k / d_k = i, S_k = m, S_{k-1} = m') = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-1}{2\sigma^2}(y_k - (2Y_k - 1))^2} = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-1}{2\sigma^2}(y_k^2 + 1)} e^{\frac{1}{\sigma^2}(2Y_k - 1)y_k}$$

$$= C_k \cdot e^{(2Y_k - 1)\cdot L_c y_k/2} \quad (3\text{-}60)$$

Avec $$L_c = \frac{2}{\sigma^2} \quad (3\text{-}61)$$

Si on pose $W_{2k} = La_{2k}$ (La_{2k} est l'information a priori sur le bit d'information d_k utilisée à l'entrée du premier décodeur) on aura

$$P(d_k) = \frac{e^{(2d_k - 1)\cdot La_{2k}}}{1+e^{(2d_k - 1)\cdot La_{2k}}} = \frac{e^{-La_{2k}/2}}{1+e^{-La_{2k}}} e^{(2d_k - 1)\cdot La_{2k}/2}$$

$$= D_k e^{(2d_k - 1)\cdot La_{2k}/2} \quad (3\text{-}62)$$

En substituant les expressions (3-59), (3-60) et (3-62) dans l'équation (3-53). Et pour une probabilité $P\{d_k = i / S_k = m, S_{k-1} = m'\}$ non nulle, on trouve

$$\gamma_i(R_k, m', m) = E_k e^{\{L_c x_k (2d_k - 1) + L_c y_k (2Y_k - 1) + La_{2k}(2d_k - 1)\}/2} \quad (3\text{-}63)$$

Avec $$E_k=B_kC_kD_k \tag{3-64}$$

Finalement, le LLR à la sortie du premier décodeur $\Lambda_1(d_k)$ peut être factorisé et exprimé par

$$\Lambda_1(d_k) = La_{1k}+L_cx_k+Le_{1k} \tag{3-65}$$

Où $$Le_{1k}=\log\frac{\sum_m\sum_{m'}\sum_{j=0}^{1}\gamma_1^e(y_k,m',m)\alpha_{k-1}^j(m')\beta_k(m)}{\sum_m\sum_{m'}\sum_{j=0}^{1}\gamma_0^e(y_k,m',m)\alpha_{k-1}^j(m')\beta_k(m)} \tag{3-66}$$

Où aussi $$Le_{1k}=\Lambda_1(d_k)-La_{2k}-L_cx_k \tag{3-67}$$

3.2.2 Application du Décodage MAP aux Turbo Codes

Les figures (3-10) et (3-11) illustrent la structure du décodeur itératif d'un turbo code parallèle et série.

Les deux décodeurs MAP de la figure (3-10) sont identiques. Chaque décodeur lit les observations communes sur les bits d'information et les observations de ses propres bits de parité. L'entrelaceur permute seulement les observations des bits d'information du DEC1 vers le deuxième décodeur DEC2. L'information extrinsèque (Le_{1k}) est calculée par soustraction de l'information a priori (La_{1k}) et de l'observation systématique (L_cx_k) du LLR ($\Lambda_1(d_k)$), ensuite injectée à l'entrée du DEC2 comme information a priori (La'_{1k}). Chaque décodeur élémentaire correspond à 1/2 itération. Une itération complète comprend le décodage de C_1 et C_2. Les observations ne changent pas en fonction des itérations, seules les informations extrinsèques et les probabilités a priori bougent d'une itération à l'autre. A l'itération 0, la première entrée a priori est initialisée à 0.

Les deux décodeurs MAP de la figure (3-11) sont différents. Chaque décodeur lit les observations de ses bits codés. Les observations des bits d'information de C_2 s'obtiennent par entrelacement de celles des bits codés de C_1, car C_2 est systématique. Les informations extrinsèques des bits codés calculées par le premier décodeur DEC1 sont utilisées comme probabilités a priori sur les bits d'information du deuxième décodeur DEC2. Les informations extrinsèques des bits d'information du deuxième DEC2 sont réinjectées comme des probabilités a priori sur les bits codés à l'entée du premier décodeur. Une itération complète

comprend le décodage de C_1 et C_2. Les observations ne changent pas en fonction des itérations, seules les informations extrinsèques et les probabilités a priori bougent d'une itération à l'autre. A l'itération 0, la première entrée a priori est initialisée à 0.

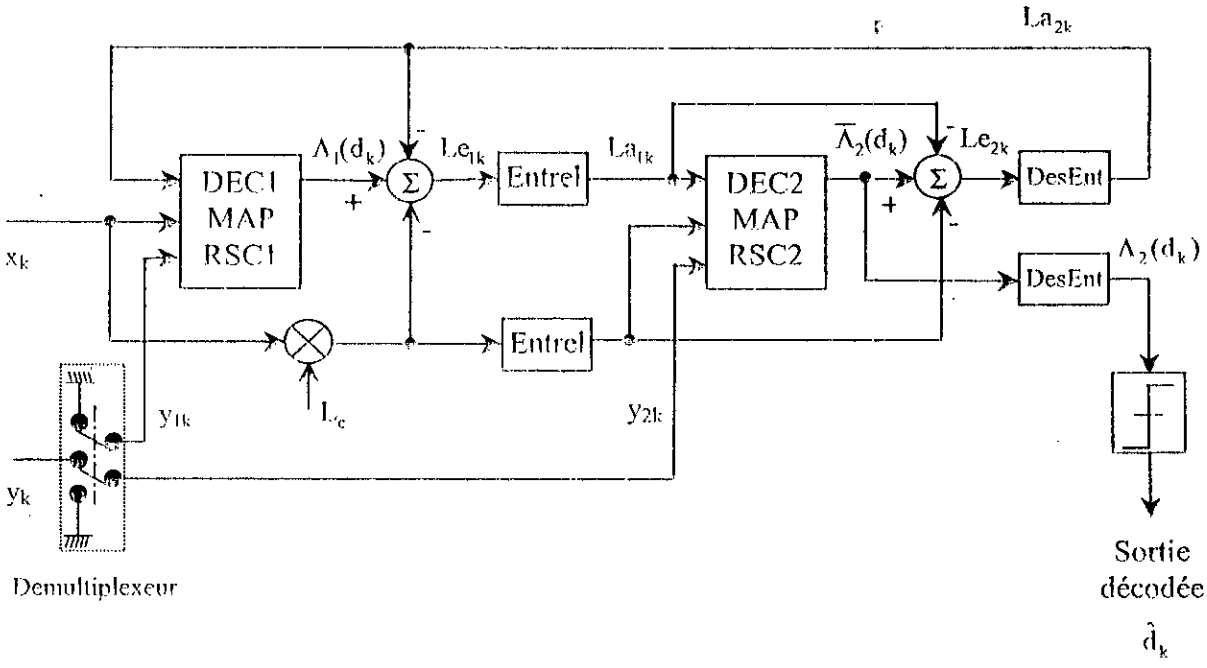


Figure (3-10) : Le décodage itératif d'un turbo code parallèle

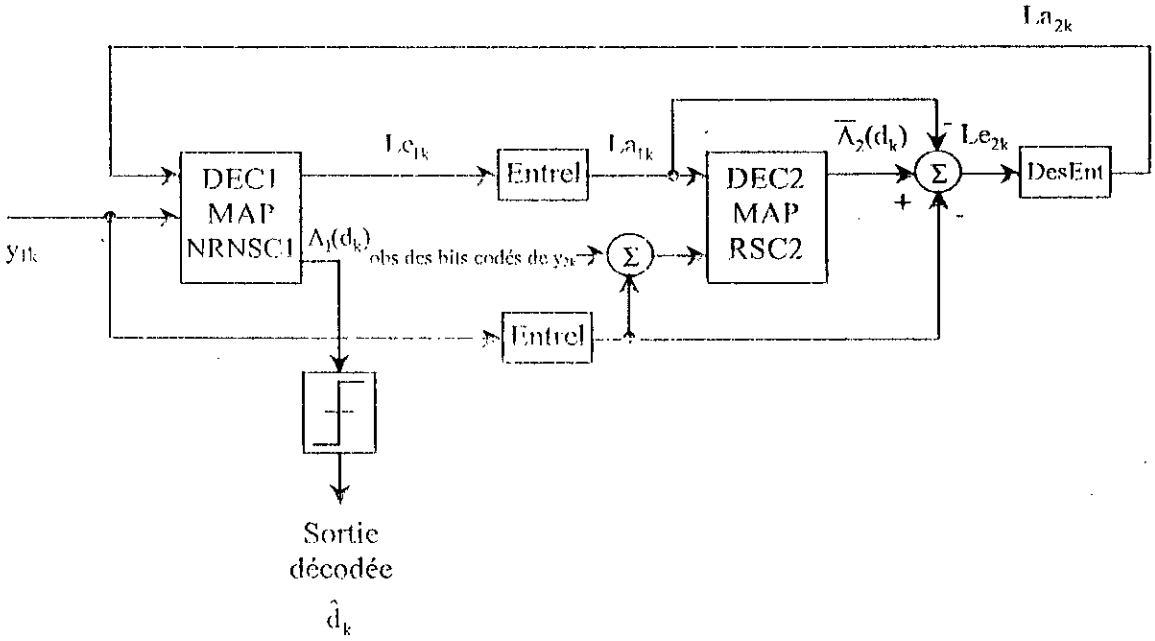


Figure (3-11): Le décodage itératif d'un turbo code série

Chapitre 4

Simulation et Résultats

Nous présentons dans ce chapitre les probabilités d'erreurs par bits BER (P_{ch}) en fonction du SNR ($SNR/bit = 10\log_{10}(E_b/N_0)$) pour des codes convolutifs et différents turbo codes parallèles et séries.

Les taux d'erreurs sont calculés par la méthode de Monte Carlo sur un canal à bruit additif blanc gaussien (canal AWGN).

Les J bits codés issus d'un turbo code sont émis sur un canal AWGN. Chaque bit codé est véhiculé par un symbole d'une modulation BPSK d'énergie moyenne $E_c = R \times E_b$. Nous considérons que les énergies moyennes par bit d'information E_b et par bit codé E_c sont associées au signal modulé sur fréquence porteuse.

Sauf indication, tous les entrelaceurs utilisés dans la concaténation sont pseudo-aléatoires et sont réalisés en précalculant une permutation Π aléatoire de taille N. La permutation sert à permuter N valeurs binaires, N observations réelles représentées par des probabilités conditionnelles, ou N probabilités a priori sur les bits entrelacés.

4.1 Codage convolutif

La figure (4-1-(a)) est le résultat d'un décodeur MAP utilisant un codeur convolutif récursif et systématique (RSC), par contre, la figure (4-1-(b)) montre la comparaison entre les deux codeurs (RSC) et (NRNSC), avec:

- Le rendement (taux) de codage: $R = \frac{1}{2}$.
- Les valeurs des fonctions génératrices: $G_1 = 37$ et $G_2 = 21$.

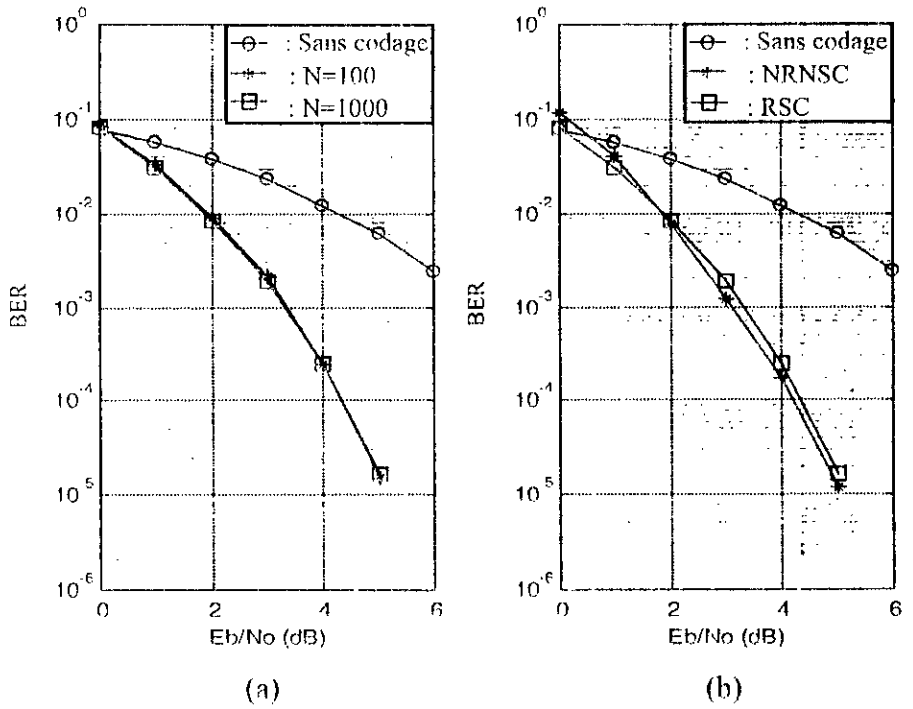


Figure (4-1) : (a) :Effet de la longueur de la trame N, (b) : effet de la récursivité et de la systématique sur le décodeur.

On remarque dans la figure (4-1-(a)) que la taille de la trame N n'a pas d'influence sur les performances du code convolutif. Nous remarquons aussi que le code RSC possède de meilleures performances que celui du code NRNSC pour un faible SNR (inférieur à 2 dB) et de moins performances pour un SNR important, ce qui justifie le choix d'un code RSC pour les turbo codes parallèles.

Les courbes obtenues dans la figure ci-dessous, sont le résultat d'un codage NRNSC dont ses paramètres sont les suivants:

- Le rendement (taux) de codage: $R = \frac{1}{2}$.
- La longueur de la trame : N=1000 bits.

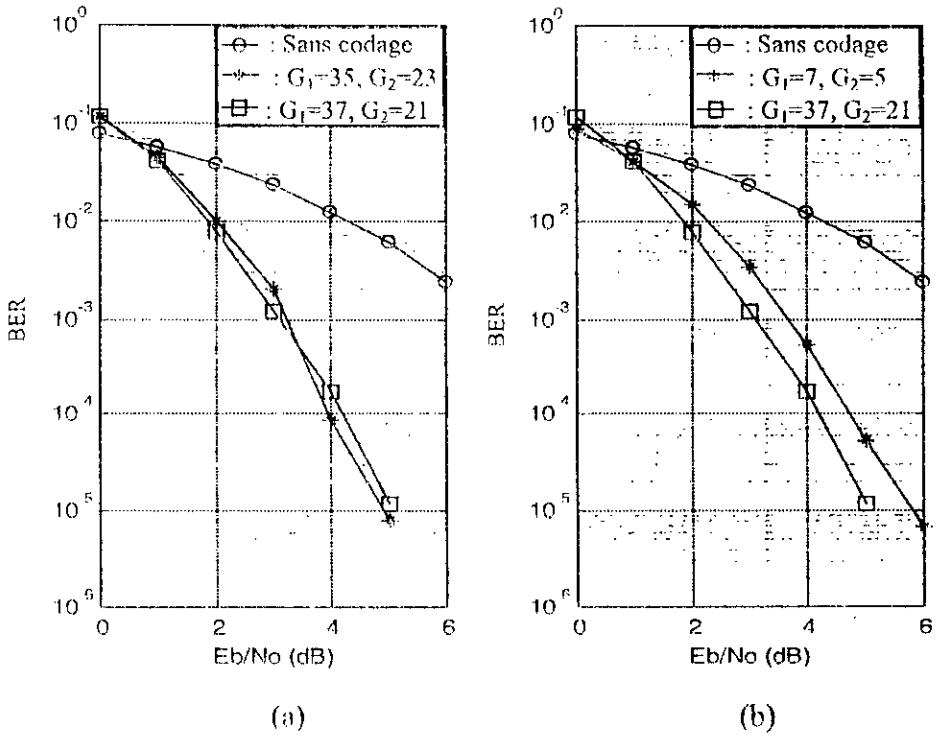


Figure (4-2) : (a) :Effet des coefficients G_1 et G_2, (b) : effet du nombre de mémoires (M) du registre à décalage.

Comme le montre la figure (4-2-(a)), les fonctions génératrices $G_1 = 37$ et $G_2 = 21$ donnent de meilleurs résultats pour un faible SNR. On remarque aussi dans la figure (4-2-(b)) que le BER est inversement proportionnel à la taille du registre à décalage (nombre de mémoires).

4.2 Turbo codes

4.2.1 Concaténation parallèle

- **L'effet du nombre d'itérations:**

Dans la figure ci-dessous, on montre l'effet du nombre d'itérations sur le décodage itératif dont les caractéristiques du codeur sont les suivantes:

- Le nombre de bascules du registre à décalage: M = 4.
- La longueur de la séquence du message: N = 65536 bits.
- La taille totale des bits transmis: L = 65536×100 bits.
- Le rendement (taux) de codage: $R = \frac{1}{2}$.
- Les valeurs des fonctions génératrices: $G_1 = 37$ et $G_2 = 21$.

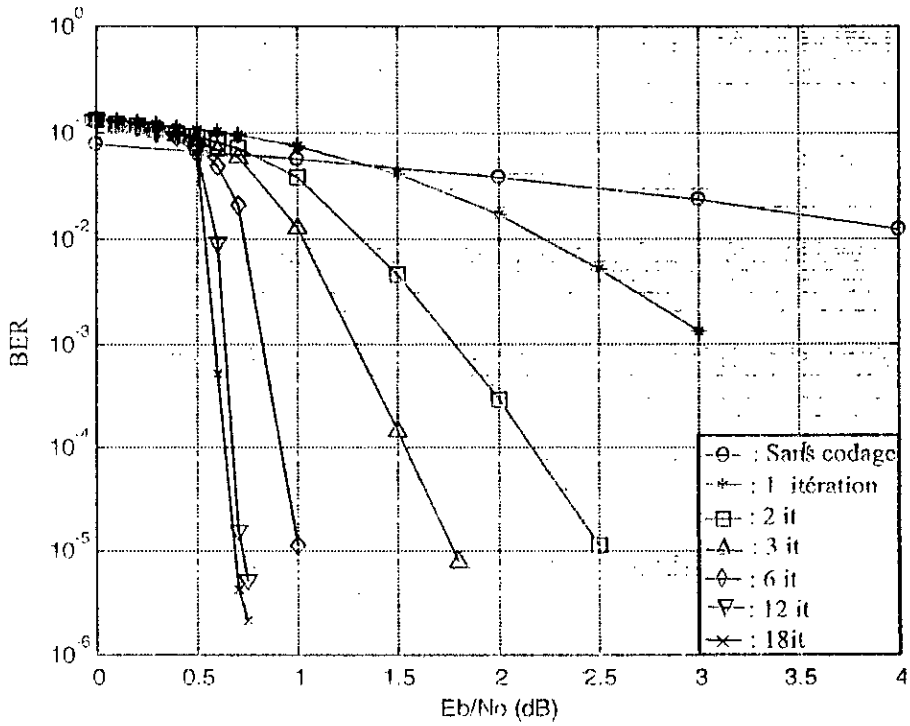


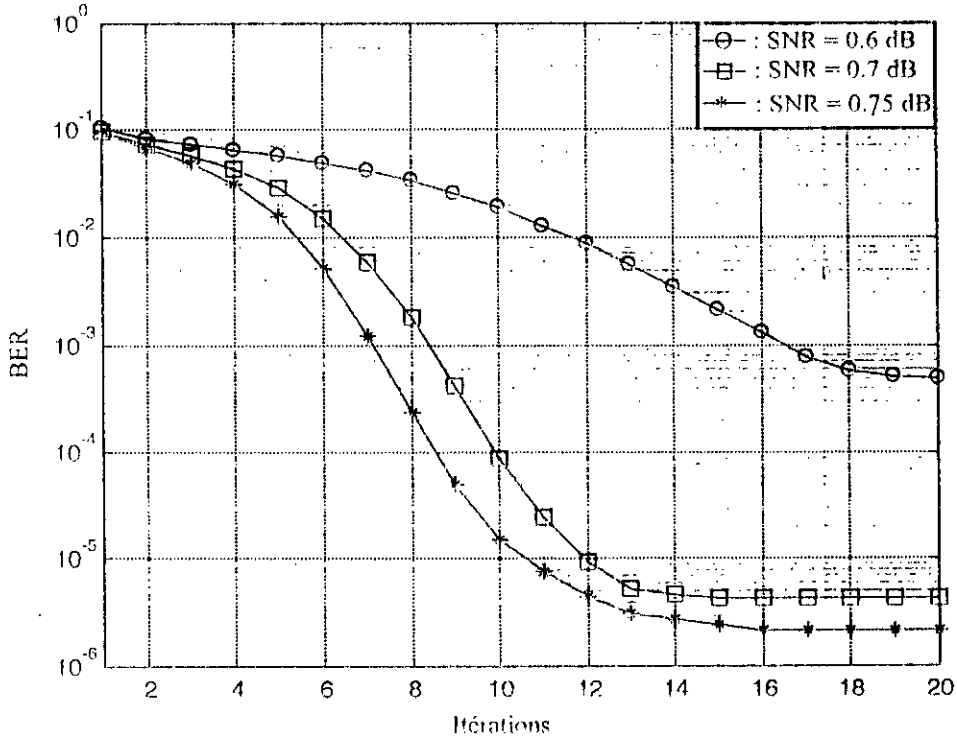
Figure (4-3): Taux d'erreurs par bit d'un turbo-décodeur pour différentes itérations

On voit bien dans cette figure l'effet du décodage itératif, on atteint un BER inférieur à 10^{-5} avec un SNR = 0.7 dB et avec 18 itérations. On remarque aussi qu'à partir du SNR > 0.5 dB, l'effet du décodage itératif commence à apparaître.

La saturation est remarquable à partir de la sixième itération; par exemple, si on cherche un Turbo-décodeur dont le SNR ≤ 1 dB et une probabilité d'erreur inférieure ou égale à 10^{-5}, on dira que sept (07) itérations réaliseront cette tache.

Dans la figure(4-4), on montre le phénomène de saturation et ceci est fait pour deux SNR différents avec les caractéristiques suivantes:

- Le nombre de bascules du registre à décalage : M=4.
- La longueur de la séquence du message : N = 65536 bits.
- La taille totale des bits transmis : L = 65536×100 bits.
- Le rendement (taux) de codage: R=0.5.
- Les valeurs des fonctions génératrices: $G_1 = 37$ et $G_2 = 21$.



Figure(4-4): Effet de la saturation

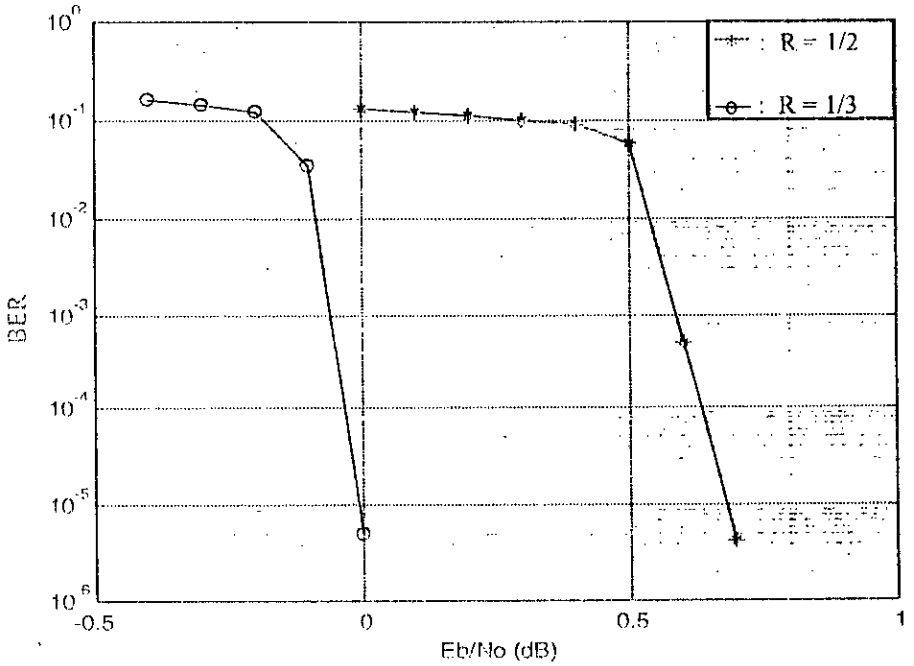
Les courbes ci-dessus ont été calculées pour trois SNR différents de la figure précédente, on montre ici que le Turbo-décodeur n'est pas efficace pour un SNR = 0.6 dB, déjà son BER

converge lentement vers le point de saturation. Pour les deux autres courbes, elles convergent rapidement vers leurs points de saturation.

- **L'effet du rendement de codage:**

La figure(4-5) montre l'influence du taux de codage sur les performances du turbo-décodeur, ceci est réalisé avec les paramètres suivants:

- Le nombre de bascules du registre à décalage: M = 4.
- La longueur de la séquence du message: N = 65536 bits.
- La taille totale des bits transmis: $L = 65536 \times 100$ bits.
- Le nombre d'itérations égale à 18.
- Les fonctions génératrices : $G_1 = 37$ et $G_2 = 21$.



Figure(4-5): Le BER pour deux rendements différents

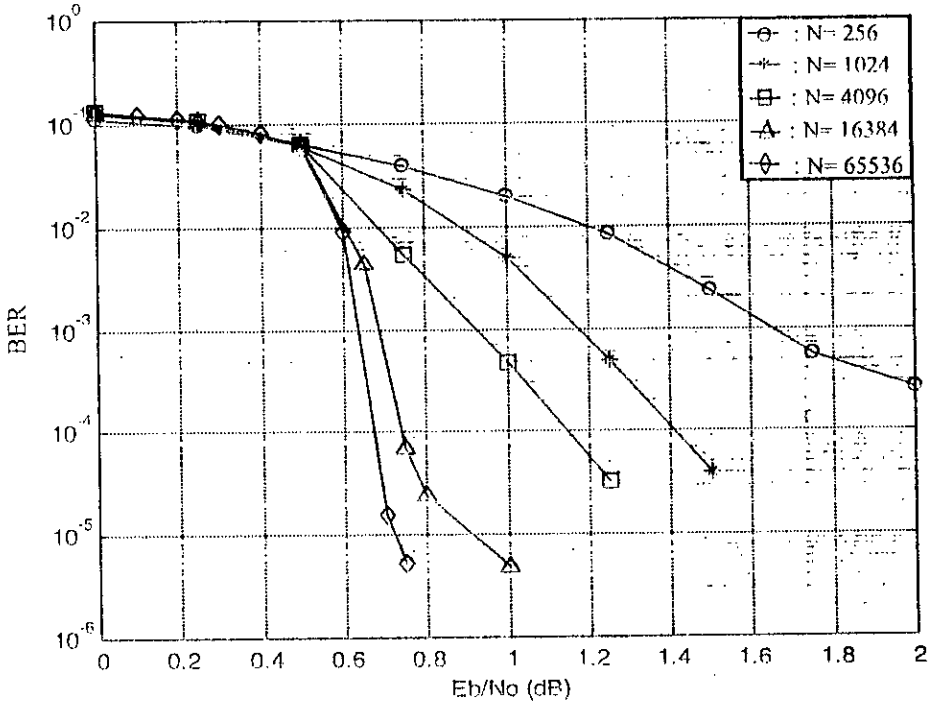
Ici, dans cette figure, on s'intéresse aux performances du décodage itératif en fonction du rendement de codage "R". On remarque que la courbe obtenue pour R=1/3 est presque un

simple décalage à gauche de celle obtenue pour R=1/2, ce décalage en dB est un gain en codage qui peut atteindre jusqu'à 0.7 dB.

- **L'effet de la longueur de la trame:**

Les courbes ci-dessous ont été calculées pour différentes longueurs de trames des bits transmis, et les paramètres du turbo-codeur sont comme suit:

- Le nombre de bascules du registre à décalage: M = 4.
- Le nombre d'itérations égale à 12.
- Les fonctions génératrices: $G_1 = 37$ et $G_2 = 21$.
- Le rendement (taux) de codage: $R = 0.5$.



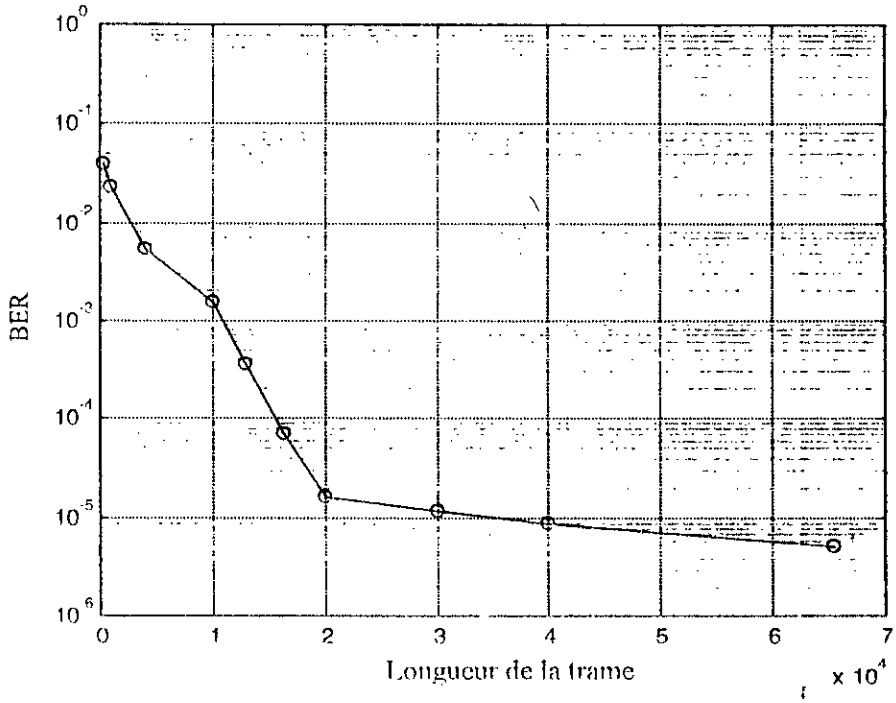
Figure(4-6): Influence de la longueur de la trame sur le turbo-décodeur

Dans le troisième chapitre, on a parlé de l'information extrinsèque générée par le deuxième décodeur utilisée en contre-réaction par le premier décodeur, qui est faiblement corrélée avec les entrées de ce dernier. La non-corrélation est réalisée par l'entrelaceur qui est mis entre les deux décodeurs, et comme la non-corrélation est proportionnelle à la distance $|n-k|$ (n est la

position du bit avant entrelacement et k est sa position après entrelacement) donc proportionnelle à la taille de l'entrelaceur qui est de N, ce qui justifie les résultats obtenus dans la figure (4-6).

Dans la figure(4-7), on montre l'effet de la longueur de la trame sur le Turbo-décodeur avec les paramètres suivants:

- Le nombre de bascules du registre à décalage : M = 4.
- Le nombre d'itérations égale à 12.
- Les fonctions génératrices: $G_1 = 37$ et $G_2 = 21$.
- Le rapport signal sur bruit : SNR = 0.75 dB.
- Le rendement (taux) de codage: R = 0.5.



Figure(4-7): Evolution du BER en fonction de la longueur de la trame

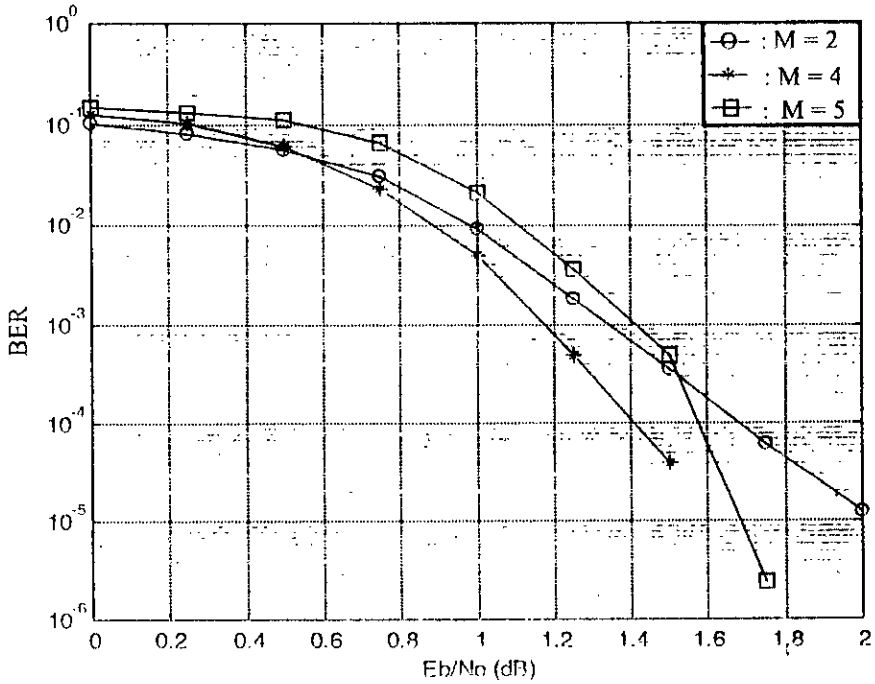
Cette courbe est calculée pour différentes valeurs de la taille de l'entrelaceur. On remarque que la faible corrélation entre l'information extrinsèque (information a priori) et les observations d'entrées du premier décodeur commence à être suffisante à partir de la taille N = 20000 bits.

- **L'effet du nombre de mémoires du registre à décalage:**

On montre l'influence du nombre de mémoires (bascules) du registre à décalage sur les performances du turbo-décodeur.

Les paramètres du turbo-codeur sont comme suit:

- La longueur de la séquence du message: N = 1024 bits.
- La taille totale des bits transmis: L=1024×500 bits.
- Le nombre d'itérations égale à 12.
- Le rendement (taux) de codage: $R = 0.5$.
- Les fonctions génératrices: $\begin{cases} \text{Pour } M=2: G_1=7, G_2=5; \\ \text{Pour } M=4: G_1=37, G_2=21; \text{ et} \\ \text{Pour } M=5: G_1=75, G_2=53. \end{cases}$



Figure(4-8): Comparaison des BERs pour différentes valeurs de M

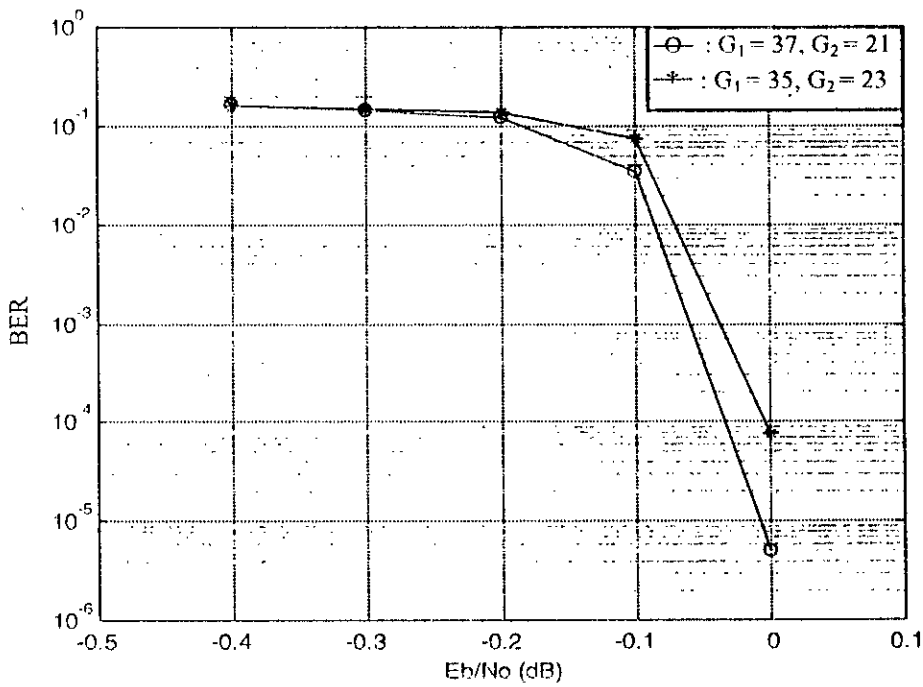
On remarque, dans cette figure, que les résultats obtenus pour $M = 2$ sont meilleurs pour un faible SNR ; par contre $M=5$, donne de meilleurs résultats pour un grand SNR.

- **L'effet des valeurs des fonctions génératrices G_1 et G_2, filtres (RIF)**

Dans la figure ci-dessous on montre l'effet de la réponse impulsionnelle du filtre (codeur). Le filtre est le registre à décalage et les coefficients de la réponse impulsionnelle sont les liaisons 1 ou 0 (g_{ji}) du registre à décalage avec la sortie du codeur.

Les paramètres de turbo-codeur sont les suivants:

- Le nombre de bascules du registre à décalage: $M = 4$.
- La longueur de la séquence du message: $N = 65536$ bits.
- La taille totale des bits transmis: $L = 65536 \times 100$ bits.
- Le rendement (taux) de codage : $R = 0.5$.
- Le nombre d'itérations égale à 18.



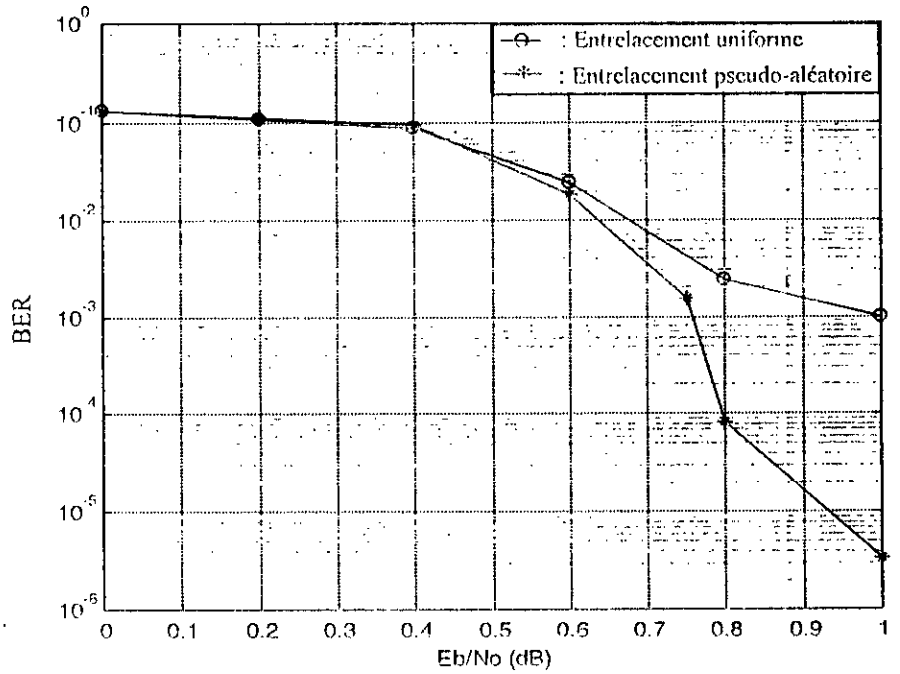
Figure(4-9): Comportement du turbo-décodeur en fonction de G_1 et G_2

On a déjà montré dans la figure (4-2-(a)) que les fonctions génératrices $G_1 = 37$ et $G_2 = 21$ donnent de meilleurs résultats avec un faible SNR pour les codeurs convolutifs, et comme les turbo codes fonctionnent pour de faibles énergies alors la figure (4-9) justifie les résultats obtenus.

- **L'effet du type d'entrelacement:**

On présente deux types d'entrelacement, l'un est un entrelacement pseudo-aléatoire et l'autre est un entrelacement uniforme avec les paramètres suivants du codeur.

- Le nombre de bascules du registre à décalage: $M=4$.
- La taille de la trame du message: $N = 10000$ bits.
- La taille totale des bits transmis: $L = 10000 \times 100$ bits.
- Le rendement (taux) de codage: $R = 0.5$.
- Le nombre d'itérations égale à 12.



Figure(4-10): L'effet du type d'entrelacement sur les performances du turbo-décodeur

On remarque dans la figure (4-10) que l'entrelacement pseudo-aléatoire donne de meilleurs résultats que celui d'uniforme, ceci est du au fait que le premier réalise la moindre corrélation entre l'information a priori et les observations des bits d'information et des bits de parité.

4.2.2 Concaténation Série

Pour la concaténation série, on a choisit l'exemple suivant :

- Le nombre de bascules du registre à décalage: M = 2.
- La longueur de la séquence du message: N = 10000 bits.
- La taille totale des bits transmis: L = 10000×300 bits.
- Le rendement (taux) de codage: $R = \frac{1}{3}$.
- Les valeurs des fonctions génératrices: $G_1 = 7$ et $G_2 = 5$.

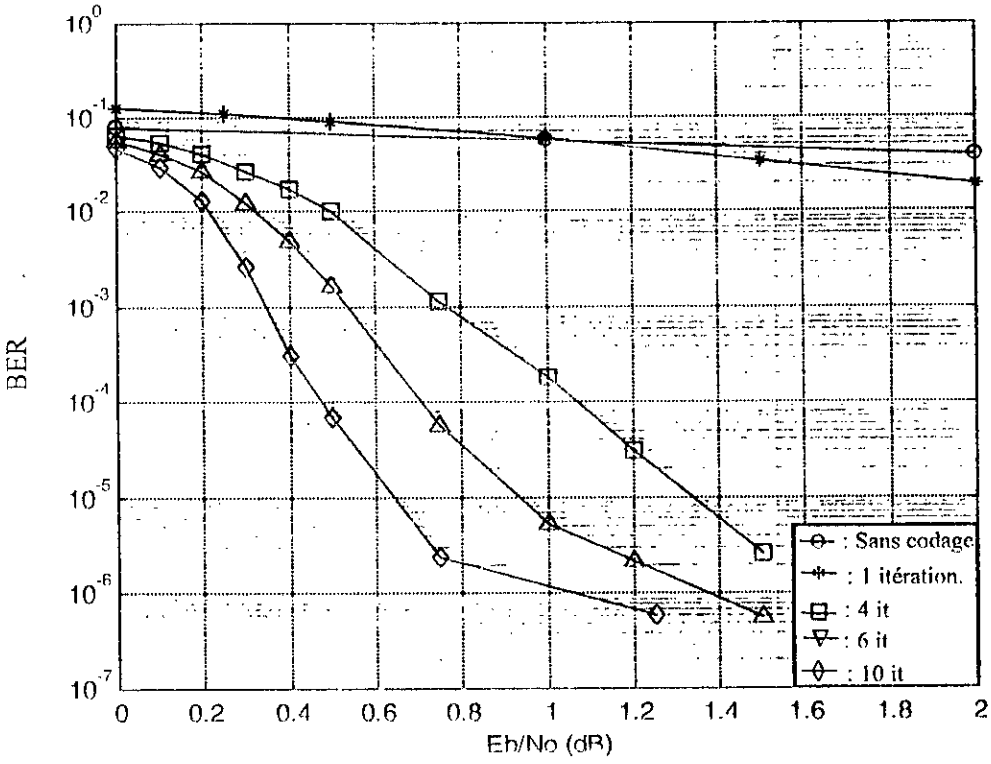


Figure (4-11): Les performances d'un turbo-décodeur SCCC

Les courbes présentées dans cette figure montrent l'effet du décodage itératif sur les performances du décodeur SCCC, les itérations ont le même effet que pour le turbo-décodeur parallèle.

Dans la figure (4-12), on montre la comparaison des performances entre les deux concaténations parallèle et série, avec :

- La longueur de la séquence du message: N = 10000 bits.
- La taille totale des bits transmis: $L = 10000 \times 300$ bits.
- Le rendement (taux) de codage: $R = \frac{1}{3}$.
- SPPP: $M = 2, G_1 = 7$ et $G_2 = 5$.
- CPPP: $M = 4, G_1 = 37$ et $G_2 = 21$.

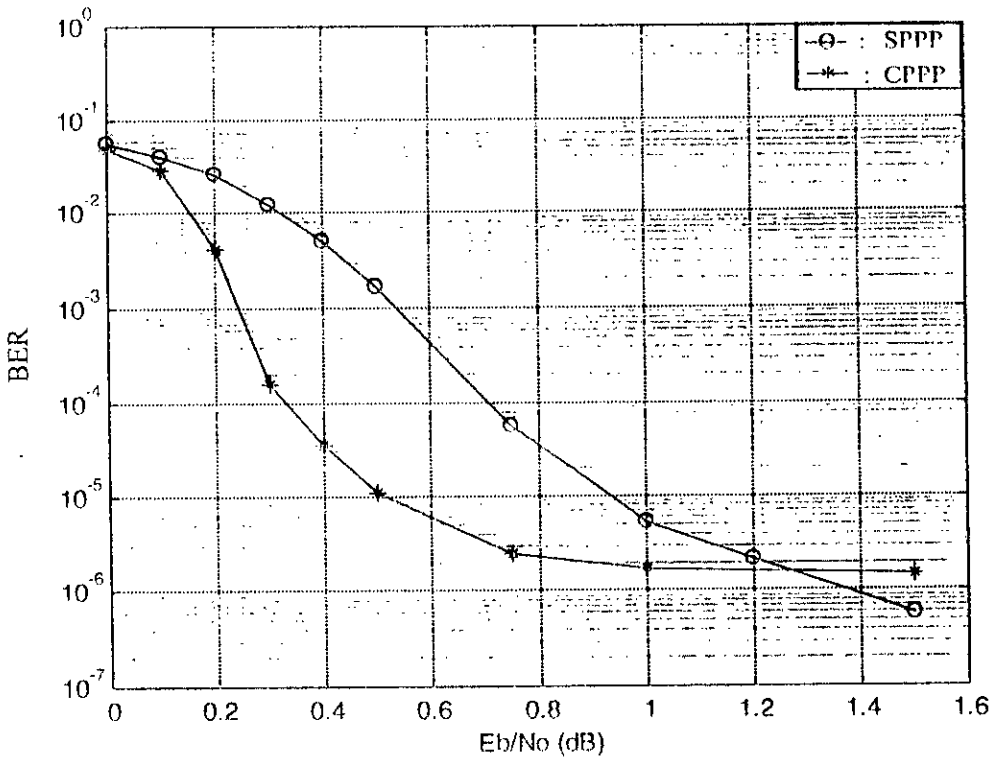


Figure (4-12): Comparaison de performances entre la PCCC et la SCCC

Les résultats obtenus dans cette figure avec les paramètres donnés ci-dessus montrent que la concaténation parallèle est meilleure pour de faibles SNR, par contre, la concaténation série donne de bonnes performances pour de grands SNR.

4.2.3 Application du Turbo Code à une Image non Compressée

On présente ici les résultats de simulation d'un turbo-décodeur MAP appliqué à une image couleur de taille 175×250 pixels, chaque pixel est codé sur 24 bits (16 millions de couleurs).

Les paramètres de cette simulation sont les suivants :

- Le codeur: concaténation parallèle de deux codeurs systématiques récursifs de mémoires $M = 4$ et de fonctions génératrices (37,21).
- La longueur de la trame : $N=10000$.
- Le rendement du codage : $R=\frac{1}{3}$.
- Le rapport signal sur bruit $SNR=10\log\left(\frac{E_b}{N_0}\right)=0.5$ dB.
- Le nombre total des bits transmis : $L=1.05$ Mbits.

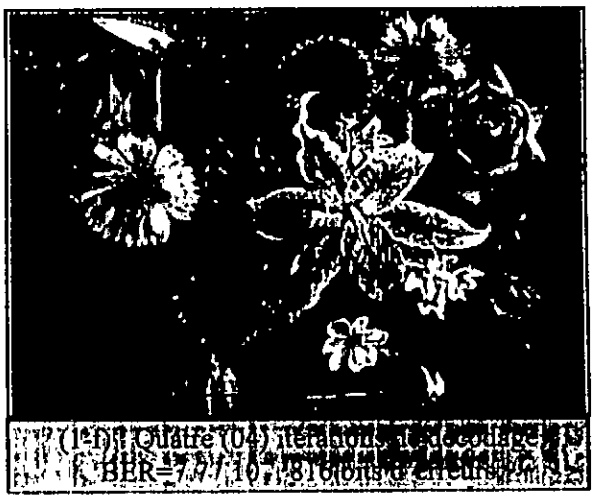
(1-a) : Image originale de taille 175x250 sur 24 bits

(1-b) : Image reçue sans codage
BER=0,0669, 70204 bits d'erreurs

(1-c) : Une itération de décodage
BER=0.1074, 112723 bits d'erreurs

(1-d) : Deux (02) itérations de décodage
BER=4.52 10^{-4}, 47475 bits d'erreurs

(1-e) : Trois (03) itérations de décodage ; BER=9.5 10^{-3} ; 9989 bits d'erreurs



(1-f) : Quatre (04) itérations de décodage ; BER=7.7 10^{-3} ; 8160 bits d'erreurs

(1-g) : Cinq (05) itérations de décodage ; BER=3.9 10^{-3} ; 41 bits d'erreurs

Ces images montrent l'efficacité du turbo code convolutif dans la correction des bits d'erreurs pour un faible SNR.

L'image originale (1-a) est la source d'information, elle est transmise (après codage ou sans codage) sur un canal à bruit additif blanc gaussien (AWGN). L'image (1-b) est l'image reçue de la transmission de l'image originale sans codage. Par contre, l'image (1-c) est l'image obtenue après une itération de décodage. On remarque qu'elle contient plus de pixels erronés (pixel≡24 bits) que celle de (1-b), ceci justifie les résultats obtenus dans la figure (4-3). Comme le montre les images (1-d), (1-e), (1-f) et (1-g), on obtient une nette amélioration lorsqu'on augmente le nombre d'itérations.

4.2.4 Application du Turbo Code Convolutif à une Image Compressée

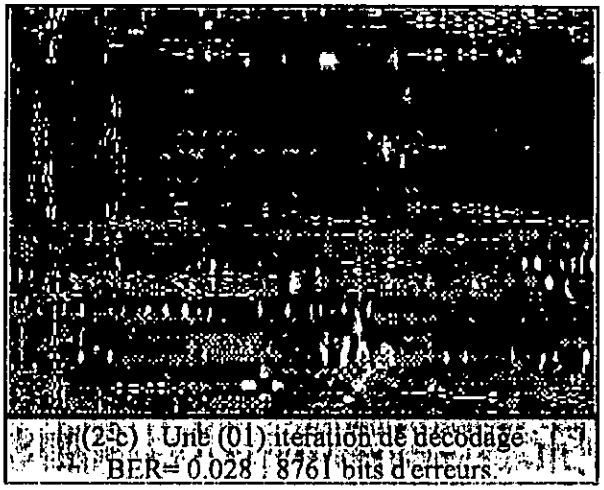
On présente ici les résultats de simulation d'un turbo-décodeur MAP appliqué à une image couleur compressée avec un taux de 70.28% (compression par la transformée d'ondelettes: Logiciel Viewer) de taille originale 175×250 pixels, chaque pixel est codé sur 24 bits (16 millions de couleurs).

Les paramètres de cette simulation sont les suivants :

- Le codeur : concaténation parallèle de deux codeurs systématiques récursifs de mémoires $M = 4$ et de fonctions génératrices (37,21).
- La longueur de la trame : $N = 1000$.
- Le taux du codage : $R = \frac{1}{3}$.
- Le rapport signal sur bruit $SNR = 10\log\left(\frac{E_b}{N_0}\right) = 1.5\,dB$.
- Le nombre total des bits transmis : $L=312$ Kbits.

(2-a) : Image originale compressée de taille 175x250 sur 24 bits.

(2-b) : Image obtenue sans codage : BER = $4.6.10^{-4}$: 14367 bits d'erreurs.



(2-c) : Une (01) itération de décodage
BER= 0.028 : 8761 bits d'erreurs

(2-d) : Deux (02) itérations de décodage
BER= 5.64.10⁻⁴ : 176 bits d'erreurs

(2-e) : Trois (03) itérations de décodage
BER= 1.92.10⁻⁵ : 6 bits d'erreurs

En comparant les images (1-f) et (2-d) qui possèdent le même ordre du BER, on remarque ce qui suit : l'image (1-f) est affectée par des centaines de bits d'erreurs (816 bits d'erreurs), néanmoins ces bits d'erreurs n'affectent pas l'image globale. Par contre, l'image (2-d) est affectée par une centaine de bits d'erreurs (176 bits d'erreurs) et elle est totalement sombre, parce que les bits d'erreurs n'affectent pas les pixels comme dans l'image (1-f), mais ils affectent les paramètres (coefficients) de la compression, ces derniers affectent l'image globale lorsqu'ils sont erronés. C'est pour cela, qu'en compression de données, un BER de 10^{-5} n'est pas suffisant, il est suggéré un BER inférieur à 10^{-7} [10].

Conclusion

Dans ce travail, nous avons étudié les principes de base des turbo codes. Une description du processus de décodage, l'algorithme MAP et le décodage itératif ont été aussi étudiés. En utilisant le domaine du logarithme de vraisemblance, nous avons montré que chaque décodeur utilise à son entrée des valeurs a priori et délivre la sortie décodée, les entrées a priori et la valeur extrinsèque. La valeur extrinsèque est utilisée comme une valeur a priori pour la prochaine itération.

Les systèmes de codage présentés dans cette thèse donnent de très hauts gains en codage. On atteint un $BER \leq 10^{-5}$ avec un $SNR = 0.7\,dB$, $N = 65536$ et $R = 0.5$ pour des codes convolutifs systématiques concaténés parallèlement.

Une propriété fondamentale des turbo codes qui a été illustrée par les exemples simples de ce travail, est que la performance du code s'améliore lorsque la longueur de la séquence du message et/ou le nombre de mémoires augmentent.

Les turbo codes convolutifs ont toujours besoin d'un ajout de M bits à chaque trame de taille N pour ramener l'état du codeur à zéro. Les performances de quatre, seize et trente deux

états, ainsi que les rendements1/2 et 1/3 des turbo codes et en utilisant l'entrelaceur pseudo-aléatoire ont été étudiés pour un canal AWGN pour de faibles rapports signal sur bruit.

En comparant les performances des turbo codes à quatre et à seize états, il est clair que pour de faibles E_b/N_0 le code à quatre états est plus performant. Mais si on augmente le SNR, le code à seize états est plus efficace que celui du code à quatre états. Le turbo code aux "nombres d'états hybrides" [20], à quatre et à seize états décode premièrement le plus puissant code à quatre états, ensuite il accomplit un "nettoyage" du bruit qui est équivalent à une augmentation du SNR avec le code à seize états.

Les résultats obtenus par les entrelaceurs de tailles importantes sont très attractifs. Par contre, les entrelaceurs de petites tailles offrent aux turbo codes des gains en codage acceptables qui leurs permettent de les utiliser dans les systèmes de communications mobiles par satellite[18].

La complexité du décodeur met une limite supérieure sur la longueur du registre à décalage M, pendant que le retard admissible maximal dans le système met une limite sur la longueur du bloc N.

Les turbo codes aux entrelaceurs pseudo-aléatoires peuvent être aussi utilisés dans le domaine de la cryptographie [20]. L'idée utilisée est semblable à celle de la technique du spectre étendu : noyer le signal dans un bruit et le récupérer à la réception. C'est possible avec les turbo codes puisqu'ils opèrent pour de faibles E_b/N_0. La supposition qui a été fait ici est que nous utilisons un très faible E_b/N_0 pour que chaque séquence de bits d'information ou de bits codés soit très difficile à extraire du bruit. Seulement, le haut gain du turbo décodeur peut extraire avec succès les données transmises qui sont noyées dans le bruit. Les sorties du codeur sont noyées dans le bruit dont la variance est variable dans chaque trame. L'entrelaceur pseudo aléatoire et la variance du bruit ajouté à l'émetteur sont les clefs du système de cryptage. Pour assurer la confidence de l'information les clés du système de cryptage sont connues à la réception. Supposons que le plan du codage est connu par un étranger, ce serait très difficile de trouver les variables manquantes indispensables au turbo décodeur. C'est un domaine intéressant comme perspectives.

Les codes convolutifs concaténés en cascade et décodés itérativement (SCCC) sont construits par les mêmes codes constituants et par le même entrelaceur, que ceux de PCCC,

mais ils sont concaténés en série plutôt que d'une manière parallèle. Encore, l'algorithme de décodage itératif MAP est utilisé pour atteindre des résultats près de l'optimum. Le SCCC atteint des performances comparables à celles de PCCC et il peut offrir dans certains cas des performances supérieures.

Les turbo codes PCCC ne sont pas convenables pour les systèmes de communication qui exigent des taux d'erreurs par bits inférieurs à 10^{-7} comme dans plusieurs applications, telle que la compression élevée de données vidéo et audio. Par conséquent, c'est nécessaire d'utiliser la SCCC.

En résumé, les performances des turbo codes convolutifs dépendent de quatre paramètres essentiels :

1. La taille du bloc N;
2. Le type d'entrelaceur ;
3. Le nombre de mémoires M ; et
4. Le rendement du codage R.

A partir de ces performances excellentes, il est attendu que les turbo codes deviennent la technique du codage standard de ce troisième millénaire.

Ce travail ouvre plusieurs perspectives d'application, à savoir :

- Implantation de l'algorithme MAP sur une carte DSP ou sur des circuits FPGA.
- Réalisation d'une chaîne de communication utilisant les turbo codes.

Bibliographie

[1] Adrian S. Barbulescu and J. Buetefuer, "Practical Delay Techniques in Serial and Parallel Concatenated Schemes (Turbo Codes)," Symposium on Information Theory and its Applications. Yuzawa, Japan, Vol.2, pp 599-602, Nov-Dec 1999.

[2] Paul K. Gray, "Serially Concatenated Trellis Coded Modulation," Ph.D. Dissertation, Uni. of South Australia, Mars. 1999.

[3] M. Bossert, "Channel Coding for Telecommunications," John Wiley and Sons Ltd, England.1999.

[4] Andy Bateman, "Digital Communications, Design for the world" Addison-Wesley, 1999.

[5] Matthew C. Valenti, "Turbo Codes and Iterative Processing," Proceedings of IEEE, New Zealand Wireless Communications Symposium. '98, Auckland New Zealand, Nov. 1998.

[6] Adrian S. Barbulescu and Steven S. Pietrobon, "TURBO CODES : a tutorial on a new class of powerful error correcting coding schemes, Part 1 : Code Structures and Interleaver Design" 26 October 1998.

[7] Adrian S. Barbulescu and Steven S. Pietrobon, "TURBO CODES: a tutorial on a new class of powerful error correcting coding schemes, Part II : Decoder Design and Performance," 26 October 1998.

[8] William E. Ryan, "A Turbo Code Tutorial. " New Mexico State University, Updated 29/06/ 98.

[9] J. Hagenauer, " The Turbo Principle: Tutorial Introduction and State of the Art," Proceedings of International Symposium on Turbo Codes, Brest-France, 1997.

[10] F. Burkert and J. Hagenauer, "A Serial Concatenated Coding Scheme with Iterative 'Turbo' and Feedback Decoding," Proceedings of International Symposium on Turbo Codes, Brest-France, 1997.

[11] D. Raphaeli and Y. Zarai, "Combined Turbo Equalisation and Turbo Decoding," Proceedings of International Symposium on Turbo Codes-Brest-France 1997.

[12] S. Benedetto, D. Divsalar, G. Montorsi, F. Pollara, "Serial concatenation of interleaved codes: performance analysis, design and iterative decoding," ISIT 1997, Uim, Germany, june 29 -july 4.

[13] Jung-Fu Cheng, "Iterative Decoding" Ph.D. Dissertation, California Institute of Technology, Mars, 1997.

[14] P. Hoeher, "New Iterative Turbo Decoding Algorithms," Institute for Communications Technology, (page 63-70), CROSSREF Turbo-97.

[15] Steven S. Pietrobon, and Mark C. Reed, "Turbo-code Termination Schemes and a Novel Alternative for Short Frames," Accepted for publication, PIMRC'96, Taipei, Taiwan, Oct. 1996.

[16] J. Hagenauer, E. Offer, and L. Papke, "Iterative Decoding of Binary Block and Convolutional Codes," IEEE Transactions on Information Theory, Mars. 1996.

[17] S. Benedetto and G. Montorsi, "Unveiling Turbo Codes : Some results on Parallel Concatenated Coding Schemes, " IEEE Transactions on information theory, Mars. 1996.

[18] Matthew C. Valenti, "An Introduction to Turbo Codes," Unpublished Report, May 1996 Bradley Dept. of Elect. Eng., Virginia Polytechnic Inst.

[19] Stephen G. Wilson, "Digital Modulation and Coding, " Prentice-Hall, 1996.

[20] Adrian S. Barbulescu, "Iterative Decoding of Turbo Codes and other Concatenated Codes," Ph.D. Dissertation, Uni. of South Australia, Feb. 1996.

[21] Steven S. Pietrobon, and Jean Y. Couleaud, "Serial Concatenated Systematic Convolutional Codes for Space Communications," Submitted to the IEEE Transactions on communications, 6 October 1995.

[22] Steven S. Pietrobon, "Implementation and Performance of Serial MAP Decoder for use in an Iterative Turbo Decoder," Proc. of 1995 IEEE International Symposium on Information Theory Whistler, British, Columbia, Canada, Sept 1995, p. 471.

[23] P. Robertson, E. Villebrun, and P. Hoeher, "A Comparison of Optimal and Sub-optimal MAP Decoding Algorithms Operating in the Log-Domain," Proceedings of IEEE International Communications Conference 1995.

[24] Steven S. Pietrobon, and Adrian S. Barbulescu, "A Simplification of the Modified Bahl Decoding Algorithm for Systematic Convolutional Codes," Proc. of ISITA '94, Sydney, Australia, Nov 94, pp. 875-880.

[25] P. Robertson, "Illuminating the Structure of Code and Decoder of Parallel and Concatenated Recursive Systematic (Turbo) Codes," Proceeding of IEEE Global Telecommunications Conference, 1994.

[26] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error Correcting Coding and Decoding: Turbo-codes," Proceedings of IEEE International Communications Conference 1993.

[27] M.-P. Béal, " Codage Symbolique," Masson, Paris, 1993.

[28] S. Haykin, "Digital Communications" John Wiley and Sons, Canada 1988.

[29] Alexandru Spataru " Fondements de la théorie de la transmission de l'information, " Presses Polytechniques Romandes,1987.

[30] K. Sam Shanmugam "Digital and Analog Communications Systems," University of Kansas, John Wiley and Sons, 1985.

[31] L.R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear codes for Minimising Symbol Error Rate," IEEE Transactions on Information Theory, Mars. 1974.