République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Département d'Electronique

Laboratoire Signal & Communications

Mémoire de Magister en Télécommunications

Présenté par :

MAKDOUR Mansour

Ingénieur d'Etat en Télécommunications ITO

Thème

Application du Turbo-Code au codage de canal

Président :	**- M. HADADI Mourad**	**Professeur à l'ENP**
Rapporteurs :	**-Mr : BERKANI Daoud**	**Professeur à l'ENP**
	-Melle. MERAZKA Fatiha	**MC à l'USTHB**
Examinateurs:	**-M. ZERGUI Rachid**	**CC à l'ENP**
	-M. BOUSSEKSOU Boualam	**CC à l'ENP**
	-M. TERRA Zidane	**CC à l'ENP**

2008

E.N.P. 10, Avenue Hassen Badi – El Harrache- Alger.

Liste des figures

Liste des tableaux

Sommaire

Introduction générale

La communication numérique est en forte expansion. Ainsi, le codage du canal est devenu un élément indispensable puisqu'il permet d'assurer la protection de l'information transmise en contrôlant les erreurs de transmission. Son principe est de rajouter au message à transmettre des informations supplémentaires, qui permettent de reconstituer le message au niveau du récepteur. Un code classique (code en bloc) est défini par les trois paramètres [n, k, d], où n est la taille du code, k sa dimension et d sa distance minimale. Son principe est découpé Le message à transmettre en blocs de k bits, qui sont alors traités séparément par le codeur. Les codes classiques ne permettent pas d'atteindre la limite de Shannon. On a donc développé d'autres systèmes de codages appelé codes convolutifs. Le principe des codes convolutifs, inventés par Peter Elias en 1954, est non plus de découper le message en blocs finis, mais de le considérer comme une séquence semi-infinie de symboles qui passe à travers une succession de registres à décalage, dont le nombre est appelé mémoire du code.

Les performances de la nouvelle classe de codes convolutionnels appelés codes turbo sont examinées. Ils sont formés à partir d'une concaténation en parallèle de codes convolutionnels récursifs et systématiques. Il a été montré que ces codes sont capables d'approcher la limite de Shannon dans le canal gaussien avec un BER de 10^{-5} pour Eb/N0 = 0.7. Cependant, leur comportement dans le canal radiomobile reste difficilement prévisible, du moins sous la forme initialement proposée. L'analyse de ces codes, essentiellement par simulation, tentera d'établir à partir d'un modèle de canal une structure d'encodage (longueur des blocs, format d'entrelacement) qui permet d'atteindre une performance optimale dans le cas d'un décodeur utilisant l'algorithme SOVA.

La structure d'encodage turbo repose sur la concaténation en parallèle et série de codes convolutionnels récursifs systématiques (RSC) de longueur de contrainte k et sur un entrelaceur pseudo-aléatoire prenant des blocs d'informations de longueur très grande. La taille des blocs rend ce format inadéquat aux communications audio en temps réel.

Le schéma de la figure1 illustre le principe d'encodage turbo: la séquence u de longueur L est présentée à l'entrée de RSC1 après qu'une séquence de k-1 bits lui soit ajoutée pour remettre les codeurs à l'état initial. Les autres encodeurs reçoivent une version entrelacée de la séquence d'information. La séquence $x_{\inf}$ est alors concaténée avec les sorties des M autres codeurs pour obtenir un taux de codage $R = 1/(M+1)$. Souvent, on a recours au poinçonnage pour avoir un taux $R = 1/2$.

Grâce à l'entrelacement pseudo-aléatoire, les turbo-codes paraissent aléatoires au canal, ce qui constitue une caractéristique dont le décodage peut tirer bénéfice. Le design de l'entrelaceur affecte les performances de ces codes dans le sens qu'il modifie leurs propriétés de distance.

Le décodage, basé sur un principe itératif, utilise l'algorithme de Bahl. Cet algorithme tente de minimiser le taux d'erreur binaire par l'estimation de la probabilité à priori (APP) de chaque bit du mot-code. Ceci constitue un réel avantage sur l'algorithme de Viterbi qui calcule le MLE du mot code. Le décodage, comme cela apparaît à la figure2 requiert deux décodeurs SOVA délivrant chacun un estimé des bits et une séquence d'informations extrinsèques Z_{ext} qui constitue une mesure de confiance de l'estimation. Les séquences extrinsèques sont utilisées de façon itérative par les deux décodeurs. Au fil des itérations, la sortie x de SOVA2 s'approche d'un estimé MAP alors que les informations extrinsèques deviennent de plus en plus corrélées. Les résultats déjà publiés montrent qu'au bout de 7 itérations on arrive à des performances satisfaisantes.

La performance de ces codes justifie les nombreuses études en cours pour des applications dans les modulations à haute efficacité spectrale, par exemple la télévision numérique, et aussi dans les systèmes CDMA en radiomobile.

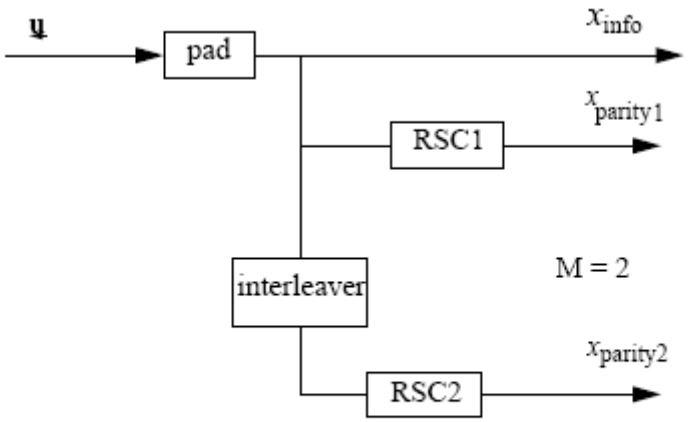


Figure 1 Schéma d'encodage

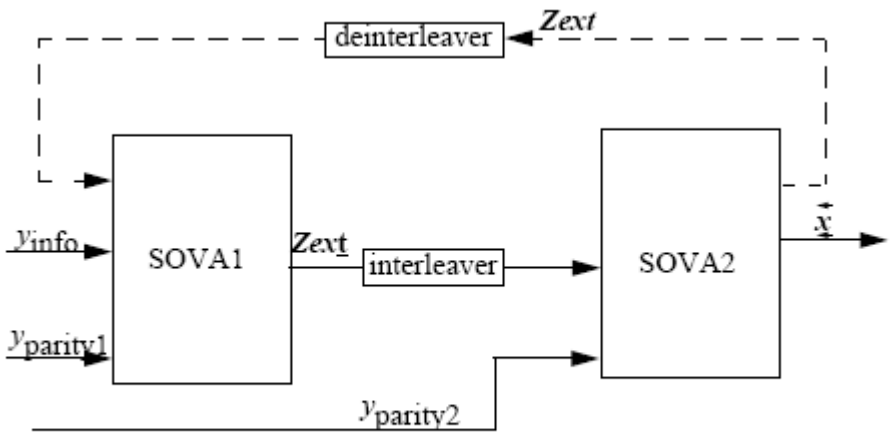


Figure 2 Principe du décodage

Donc, ce mémoire comporte 4 chapitres différents, le premier chapitre est une vue générale sur le codage de canal. Dans cet chapitre qui décrit l'architecture d'une chaîne de transmission, nous définissons ces défirent blocs, avec une étude théorique sur les codes en blocs linéaires. A partir d'un code en bloc on peut générer des codes en bloc systématique ou non systématique, sont capable de détecter et corriger les erreurs de transmission. Ceci est expliqué brièvement.

Le deuxième chapitre concernant les turbos-codeur, sont inventé pour améliorer les performances d'une communication numérique sur des canaux de transmission, ils sont inventés en 1990 par C. Berrou, A. Glavieux et P, peuvent être considérés comme un cas particulier des codes en bloc linéaires. Ces codes nécessite au moins une concaténation de deux code convolutifs préférable de type RSC et identiques séparé par un entrelaceur pour le but de la permutation des bites de la séquence du message d'entrer pour le deuxième RSC. On a expliqué dans ce chapitre la concaténation parallèle de deux codeurs convolutifs systématiques récursives, choisissant des entrelaceurs aléatoires (Section d'or, GRPI, GI, type canal).

On présente dans le troisième chapitre en détail le principe de turbo décodage itératif et l'introduction de la notion d'information intrinsèque et d'information extrinsèque, avec des différents algorithmes de décodage d'entrée soft et de sortie soft (SISO) à savoir le MAP, le Log-MAP et le SOVA.

Un quatrième chapitre contient simulations et analyses des résultats. Utilisant un simulateur d'un système de communication numérique programmé en Matlab. En distinguant

trois parties dans la simulation : le codeur, le canal et le décodeur. Le canal choisit est un canal gaussien a bruit blanc (AWGN).

Enfin, une conclusion générale sur ce travail.

I. Codage du canal

I.1. Introduction

Le codage de canal est l'une des deux parties importantes d'un système de communication numérique. Le but des ladite chaîne est de transmettre un message porteur d'information d'une source à un destinataire, éloigné soit géographiquement, soit temporellement, on parle plus souvent dans ce cas d'enregistrement, et de restitution du message).

I.2. Notion de message numérique

Un message numérique est défini comme une suite d'éléments pouvant prendre une parmi q valeurs possibles, on appelle « alphabet » l'ensemble de ces valeurs. Les éléments, qui peuvent être aussi considérés comme des variables aléatoires discrètes, sont dits q-aires. Dans le cas particulier et fréquent où l'alphabet est constitué uniquement de deux valeurs, notées traditionnellement 0 et 1, les éléments sont dits binaires. Tout élément d'un message q-aires peut être représenté par une suite d'éléments binaire, ce qui justifie l'importance du cas binaire. Un texte est un exemple de message numérique car il est composé caractères qui forment un alphabet fini. Ainsi une source discrète est obtenue telle que définie par la théorie de l'information**.** Le concept de message numérique étant précisé, nous pouvons maintenant aborder la présentation d'une chaîne de transmission numérique.

I.3. Chaîne de transmission numérique

Le schéma de principe d'une chaîne de transmission numérique est représenté sur la figureI.1, sans pour l'instant aucune justification de ses différents éléments. On peut distinguer : la source de message, le milieu de transmission et le destinataire qui sont des données du problème. Le codage et le décodage de source, le codage et le décodage de canal, l'émetteur et le récepteur représentent les degrés de liberté du concepteur pour réaliser le système de transmission. Nous allons maintenant décrire de façon succincte les différents éléments qui constituent une chaîne de transmission, en partant de la source de message vers le destinataire.

Le message porteur d'information émis par la source peut être de nature analogique (signal de parole, etc.). Dans ce cas, le signal doit être numérisé. La numérisation n'est pas étudiée ici. Rappelons tout de même brièvement qu'elle s'effectue en deux grandes étapes : échantillonnage (qui doit respecter le théorème d'échantillonnage pour garantir une

représentation fidèle du signal), puis quantification de ces échantillons et représentation binaire des valeurs quantifiées.

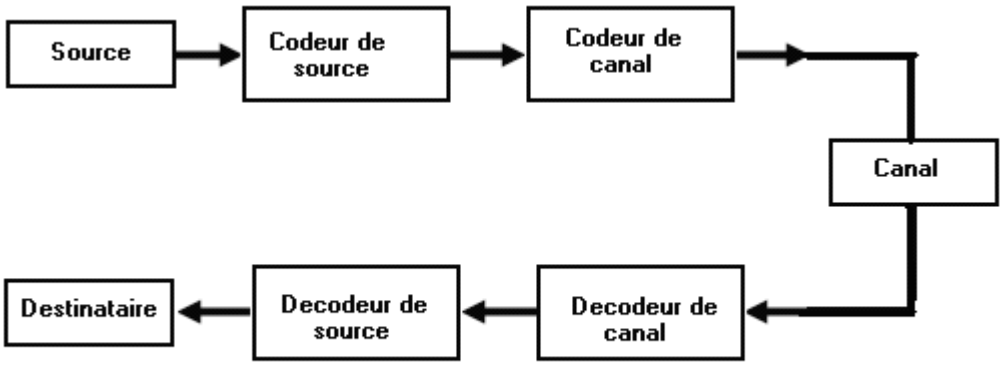


Figure I.1: Chaine de transmission numérique

I.3.1. Codeur de source

Le codage de source vise à la concision maximale du message, afin de minimiser les ressources nécessaires à la transmission (temps, puissance, bande passante, surface de stockage, etc.). Ce codage peut donc, pour diminuer le coût de la transmission, substituer un message aussi court que possible au message émis par la source, dans la mesure où cette substitution est réversible (i.e. que le message initial peut être exactement restitué). Le codage de source ne sera pas étudié. Indiquons simplement que ses limites théoriques sont fixées par la théorie de l'information (Premier théorème de Shannon).

I.3.2. Codeur de canal

Le codage de canal vise quant à lui à la protection du message contre les perturbations du canal de transmission. Si les perturbations engendrées induisent une qualité de restitution (souvent mesurée quantitativement par la probabilité d'erreur par bit sortant du codeur de source ou par message, trame, etc.) incompatible avec les spécifications fixées, le codage de canal se propose de transformer le message de manière à en augmenter la sûreté de transmission. Le « prix » qu'il en coûte est alors un accroissement de la taille du message.

Il y a donc antagonisme entre codage de source et codage de canal, l'objectif du premier étant de diminuer la redondance du message de source, celui du deuxième d'en ajouter dans un but de protection. Nous verrons que la théorie de l'information fixe les conditions pour lesquelles cet antagonisme n'est qu'apparent, et la division des tâches entre codeur de source et codeur de canal, pour l'instant arbitraire, justifiée.

I.3.3. Le canal

Le canal, au sens des communications numériques, et comme représenté à la FigureI.1, inclut le milieu de transmission (lien physique entre l'émetteur et le récepteur : câble, fibre, espace libre,…), le bruit (perturbation aléatoire issue du milieu, des équipements électroniques), et les interférences (provenant des autres utilisateurs du milieu de transmission, de brouilleurs intentionnels ou non) Par la suite, aussi bien dans la présentation des résultats de la théorie de l'information que dans cette introduction au codage de canal, nous utiliserons un modèle de canal plus global, incluant une partie de l'émetteur et du récepteur.

I.3.4. Le décodeur de canal

Plusieurs stratégies différentes peuvent être utilisées par le décodeur de canal.

- La première est la détection d'erreurs. Le décodeur observe la séquence reçue (ferme ou souple) et détecte la présence éventuelle d'erreur. Cette détection peut servir à contrôler le taux d'erreur (Error Monitoring) ou à mettre en oeuvre des techniques de retransmission (ARQ : Automatic Repeat Request) : le décodeur demande à l'émetteur de retransmettre la séquence dans laquelle une erreur a été détectée. Il est évident que ce type de procédé nécessite une voie de retour. Cette stratégie de détection est surtout utilisée par les couches transport et supérieures du modèle OSI. Les techniques employées ne seront que mentionnées, dans la mesure où il ne s'agit qu'un cas particulier et simple de la seconde stratégie.
- La deuxième est la correction d'erreurs (FEC : For Ward Error Correction). Elle nécessite des algorithmes beaucoup plus complexes que la simple détection, et plus de redondance dans la séquence émise. Toutefois, le milieu de transmission est utilisé de manière plus efficace.

I.4. Codes de blocs linéaires

Pour approcher au mieux la capacité C du canal si, conformément au théorème de Shannon, l'entropie de la source $H(X) < C$ nous pouvons ajouter de la redondance dans le codage de la source afin de diminuer les erreurs de transmission. C'est le problème du codage de canal. A côté des premiers codes empiriques (bit de parité, répétition,…) deux grandes

catégories de codes ont été développées et sont actuellement utilisées en faisant l'objet permanent de perfectionnements :

1. Les codes de blocs linéaires.
2. Les codes de convolution.

I.4.1. Définitions et propriétés des codes de blocs linéaires

I.4.1.1. Code binaire

Le codage de canal utilise les symboles d'une source pour construire des objets appelés mots de codes. Pour une source discrète binaire nous utiliserons deux symboles. Pour un alphabet non binaire de q symboles (source q-ary) si $q = 2^b$, un code de longueur N peut se ramener à l'étude d'un code binaire de longueur $b.N$. Nous pouvons aussi montrer que si $q \neq 2^b$ il est possible de se rapporter à une telle étude. Pour ces raisons, l'étude des codes binaires est une bonne introduction et nous nous contenterons de ce cas particulier.

I.4.1.2. Codes de blocs

L'information de la source est mise en trames de longueur fixe que nous devons transmettre c'est le message. Le codage de canal prend ce message pour en faire un mot de code :

$$\text{message} \rightarrow \text{codage de canal} \rightarrow \text{mot de code}$$

Le message est constitué de k caractères soit 2^k messages possibles. Le mot de code utilisé sera lui aussi de longueur fixe de n caractères soit 2^n mots de code possibles. Avec $n > k$ il y aura donc $n - k$ caractères du mot de code qui sont redondants et serviront à traiter les erreurs éventuelles. Par ailleurs, $2^n > 2^k$ donc un certain nombre de mots de code ne correspondent pas à un message mais seulement à des erreurs de transmission. On parle ainsi d'un code de bloc (n, k) et du rapport du code défini par :

$$R_c = k / n . \qquad \textbf{(I.1)}$$

Un cas particulier des codes de blocs est celui des codes où le message apparaît explicitement sur ses k caractères c'est à dire "en clair". A côté de ces k caractères seront donc ajoutés $n - k$ caractères redondants. Nous obtenons alors un code dit code systématique et c'est ce sous ensemble que nous étudierons.

I.4.1.3. Poids d'un code

Le poids d'un mot de code est par définition le nombre de caractères non nuls que contient ce mot.

11001000 est de poids 3 10011010 est de poids 4 00000000 est de poids nul

A l'ensemble des mots d'un code on associe l'ensemble des poids qui est la distribution de poids du code. Un code est dit de poids fixe (ou poids constant) si tous les mots du code ont le même poids.

I.4.1.4. Distance de Hamming

La distance de Hamming dij entre deux mots de code est le nombre de bits dont ils diffèrent.

11001000 et 10011010 ont une distance de 3.

La distance minimale du code est le minimum de l'ensemble des distances entre codes :

$$d_{\min} = \min\{d_{ij}\} \qquad \textbf{(I.2)}$$

I.4.1.5. Code linéaire

Un code est linéaire s'il répond au principe de superposition : $a_i C_i + a_j C_j = C_k$ Pour tout $a_i, a_j = \{0,1\}$ (en binaire), si Ci et Cj sont des mots de code, alors C_k est aussi un mot de code.

Conséquence :

$$\text{Si } a_i = a_j = 0 \Rightarrow C_k = 0$$

Un code linéaire contient le code "0".
Un code de poids fixe ne peut être un code linéaire.

I.4.1.6. Distance minimale d'un code linéaire

$$\begin{aligned} Cj &= Cc + Cm \\ Ci &= Cc + Cq \end{aligned} \qquad \textbf{(I.3)}$$

Avec Cc = code commun constitué des bits "1" identiques de Ci et Cj.

Cm et Cq n'ont par conséquence aucun bit commun.

C_i = 11001000

C_j = 10011010 $\Rightarrow$ Cc = 10001000, Cm = 01000000, Cq = 00010010

En appelant w les poids des codes : dij = $w_m + w_q$

I.5. Génération d'un code de blocs linéaires systématiques

A partir d'un message X_m tel que :

$$X_m = \left[\underbrace{m_0 m_1 m_2 \ldots\ldots\ldots m_{k-1}}_{k \text{ caractères du message}} \right] \quad \textbf{(I.4)}$$

Nous devons générer un mot de code de bloc linéaire systématique qui a la structure suivante :

$$C_m = \left[\underbrace{b_0 b_1 \ldots\ldots\ldots b_{n-k-1}}_{(n-k) \text{Caractères de contrôle}} \underbrace{m_0 m_1 \ldots\ldots\ldots m_{k-1}}_{k \text{ Caractères du message}} \right] \quad \textbf{(I.5)}$$

$$n \text{ Caractères du mot du code}$$

Soit $C_m = [B_m \quad X_m]$ où B_m contient les caractères de contrôle appelés aussi caractères de parité.

I.5.1. Génération du mot de code

I.5.1.1. Matrice de génération de code

Matrice notée G, elle sert à générer le mot de code C_m à partir du message X_m soit:

$$C_m = X_m . G \quad \textbf{(I.6)}$$

Nous nous intéressons à des codes systématiques, la matrice G est donc de la forme :

$$G = [P \quad I_K] \quad \textbf{(I.7)}$$

G étant une matrice (k,n), P une matrice $(k,n-k)$ et I_k la matrice identité (k,k).

I.5.1.2. Code non systématique

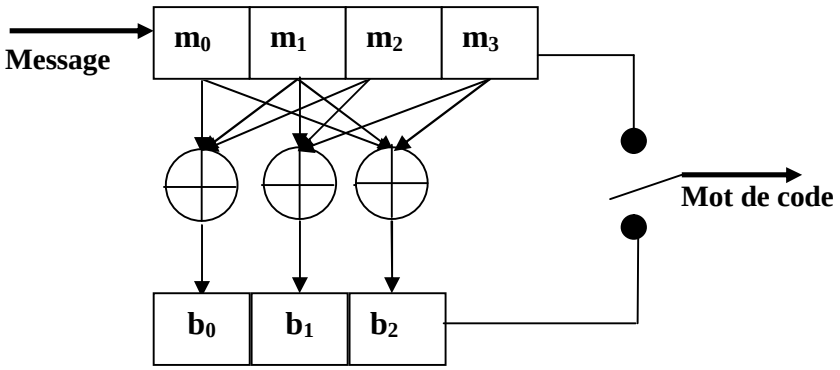
Un code non systématique serait tel que : $G = [P \quad M]$ où la matrice M est une matrice qui mélange les bits du message pour le crypter. Nous pouvons bien sûr nous ramener à un code systématique par combinaison des lignes entre elles pour faire intervenir la matrice I_k. Tout code de bloc peut ainsi se ramener à l'étude d'un code systématique équivalent. La partie essentielle de la matrice G est donc la matrice P appelée matrice de parité.

I.5.2. Principe de la réalisation du codeur

Chaque bit de parité s'exprime donc par :

$$b_i = \sum_{j=1}^{k} m_j p_{ji} \qquad \textbf{(I.8)}$$

En binaire les $p_{ji} \in \{0;1\} \Rightarrow$ les b_i sont obtenus par une simple addition binaire. Le principe de la réalisation technique en est donc simple, il suffit de disposer de deux registres à décalage et d'un inverseur (switch électronique bien sûr) ce qui donne le figureI.2.



FigureI.2 : Générateur du mot codes (codeur)

Cet exemple est celui d'un code (7, 4) tel que :

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{cases} b_0 = m_0 + m_1 + m_2 \\ b_1 = m_1 + m_2 + m_3 \\ b_2 = m_0 + m_1 + m_3 \end{cases} \qquad \textbf{(I.9)}$$

II. Turbo codeur

II.1 Introduction

Le second théorème de Shannon, publié en 1948 dans la référence [1], montre comment en principe le codage aléatoire permettrait d'atteindre la capacité d'un canal bruité, avec un taux d'erreurs arbitrairement faible. Cependant, bien que presque tous les codes aléatoires générés de cette façon sont en principe bons, il est nécessaire d'utiliser des longueurs de blocs assez importantes, ce qui conduit à des tables de code de taille élevé (grand) (la taille croît exponentiellement avec la longueur des blocs, et cela d'autant plus que le débit souhaité est grand). En conséquence, le décodage devient pratiquement irréalisable.

C'est la raison pour laquelle, depuis cette publication, de nombreux travaux ont porté sur la mise au point de codes de canal structurés (approche algébrique) de manière à en rendre le décodage faisable. Mais, le codage de canal n'a pas atteint ce niveau de maturité (expérience) à l'heure (moment) actuelle. Par exemple il n'existe pas actuellement de méthode universelle de codage de canal qui pourrait approcher asymptotiquement les limites de Shannon [2].

En effet, l'invention des turbos codes par C. Berrou et A. Glavieux et P. Thitimajishima en 1993, permis de disposer de codes de complexité raisonnable ayant des performances très proches de la limite de Shannon.

Une étude de la NASA montre que ce système de codage /décodage permet de réduire le taux d'erreur à 10^{-5} lorsque le rapport signal sur bruit est de l'ordre de 0 dB ou même lorsque celui-ci est négatif (Puissance de bruit plus forte que la puissance du signal).

Le terme turbo vient de la notion de bouclage semblable (similaire) à celle utilisée dans les moteurs turbo. L'idée est d'utiliser plusieurs séquences redondantes d'un même message. L'une étant calculée à partir du message original et l'autre à partir d'une version permutée de ce message. La structure des turbos codes résulte de la composition de deux ou plusieurs codeurs. Ces codeurs peuvent être de type convolutif systématique récursif (RSC), convolutif classique (CC) ou codes en bloc. Cette composition est appelée concaténation. Les deux plus connues sont concaténation en parallèle et en séries. Il est possible de combiner ces deux modes, on l'appelé concaténations hybrides. Mais cette dernière est décrite dans la littérature comme peu intéressante face au gain de performance déjà assuré par les modèles simples.

Dans tous les cas, les codeurs sont reliés par des entrelaceurs identiques entre eux. L'entrelacement est nécessaire lors du transfert de bits d'un codeur (ou décodeur) vers le suivant. Tout le principe des turbos codes est basé sur cette notion. L'entrelacement consiste

à permuter les données émises. Plusieurs méthodes de permutation sont possibles, cependant le choix de la structure de l'entrelaceur, est un facteur clé qui détermine les performances d'un code turbo [3].

II.2 Concaténation parallèle des codes convolutionels (PCCC)

Un codeur pour un turbo code classique consiste une concaténation parallèle de deux codeur RSC (codeur convolutionel récursif systématique) ou plus, généralement identiques, et un entrelaceur pseudo-aléatoire comme il est illustré en figure II.1. Cette structure de codeur est appelée concaténation parallèle parce que les deux codeurs opèrent sur le même bloc des bits d'entrée, par rapport à la concaténation série où le deuxième codeur (RSC2) opère sur les bits de sortie du premier codeur (RSC1). Dans tous qui suivent nous considérons seulement les turbo codes avec deux RSC, bien que cette méthode de codage puisse facilement être prolongée au cas où différents codeurs ou trois codeurs ou plus sont utilisés [4]. Cependant, les conclusions majeures et les résultats performants sont réalisables avec seulement deux codeurs RSCs.

L'entrelaceur est utilisé pour permuter les bits d'entrée tels que les deux codeurs fonctionnent sur le même bloc de bits d'entrée, mais dans un ordre différent. Le premier codeur reçoit directement le bit d'entrée u_k et produit la paire $(u_k, v_k^{(1)})$ tandis que le deuxième codeur reçoit le bit d'entrée u'_k et produit la paire $(u_k^{'}, v_k^{(2)})$, u'_k est jeté. Les séquences du bit d'entrée sont groupées dans les blocs de longueur finie égale à N (N la taille de l'entrelaceur). Puisque les deux codeurs sont systématiques et opèrent sur le même ensemble des bits d'entrée, il est seulement nécessaire de transmettre les bits d'entrée une fois, et le code global de taux $R = \frac{1}{3}$. Afin d'augmenter le taux du code à R=1/2, les deux séquences de parité, $v^{(1)}(D)$ et $v^{(2)}(D)$ peuvent être perforés, typiquement en supprimant alternativement $v_k^{(1)}$ et $v_k^{(2)}$. Nous nous référerons à un turbo code dont les RSCs ont les polynômes générateurs des bits de parité $h_0(D)$ et $h_1(D)$, et dont l'entrelaceur est de longueur N, par (h_0, h_1, N) turbo code, où h_0 et h_1 sont les représentations octales des polynômes.

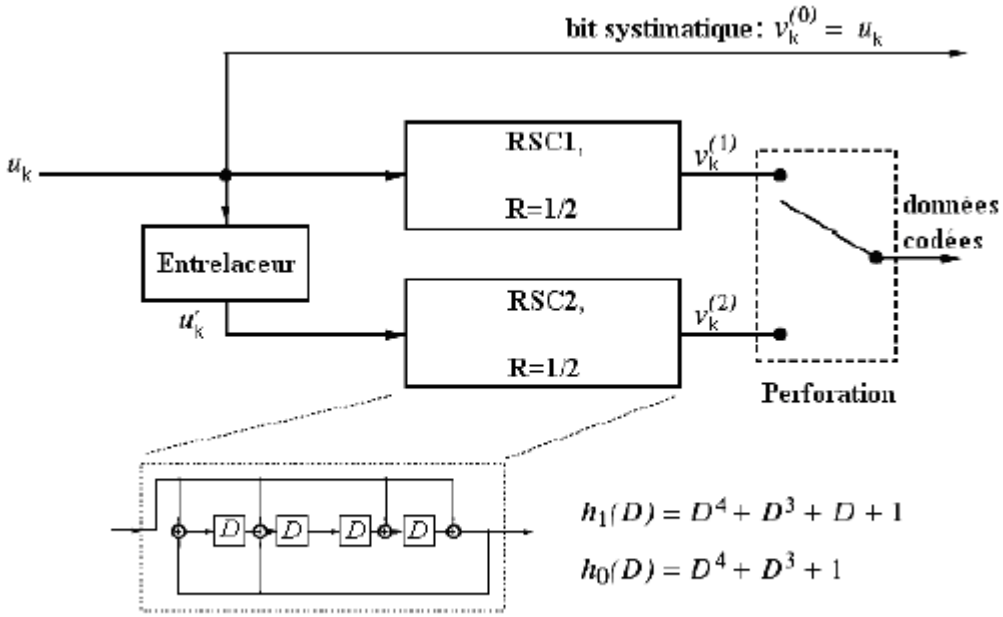


Figure II.1: Schéma d'un encodeur turbo

Afin d'illustrer les principes de base de la concaténation parallèle, on a considéré que le turbo codeur est constitué de deux RSCs identiques comme composants de codeur turbo, avec des polynômes générateurs des bits de parité $h_0(D) = D^2 + 1 = 5$ et $h_1(D) = D = 2$. Pour cette illustration on suppose un petit entrelaceur de taille N=16 qui produit le code turbo (5, 2, 16). L'entrelaceur est réalisé comme une matrice de 4×4 remplit séquentiellement ligne par ligne, avec u_k bites d'entrée. Une fois que l'entrelaceur a été rempli, les bits d'entrée au deuxième codeur u'_k sont obtenus en lisant l'entrelaceur d'une façon pseudo-aléatoire jusqu'à ce que chaque bit ait été lu une fois et seulement une fois. L'entrelaceur pseudo-aléatoire dans notre exemple est représenté par la permutation $\Pi_{16} = \{15,10,1,12,2,0,13,9,5,3,8,11,7,4,14,6\}$, ce qui implique $u'_0 = u_{15}, u'_1 = u_{10}$, et ainsi de suite.

Si $\{u_0 .....u_{15}\} = \{1,0,0,0,1,0,0,0,0,0,0,0,1,0,1,0\}$ est la séquence d'entrée, et l'entrelaceur est représenté par la permutation Π_{16}, alors la séquence $u' = \Pi_{16}(\{u_0 .....u_{15}\}) = \{0,0,0,1,0,1,0,0,0,0,0,0,0,1,1,0\}$ est l'entrée du second codeur. Les diagrammes de treillis pour les deux codeurs constitutifs avec ces entrées sont montrés dans la figure II.2. La séquence de parité correspond au premier codeur est $\{0,1,0,1,0,0,0,0,0,0,0,0,0,1,0,0\}$ et pour le deuxième codeur est $\{0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,1\}$. Sans perforation le mot code résultant à un poids de Hamming $d = w(u) + w(v^{(1)}) + w(v^{(2)}) = 4 + 3 + 3 = 10$. Si le code est perforé et que la perforation est débutée avec $v_0^{(1)}$, alors le mot code résultant à un poids d= 4+0+1=5.

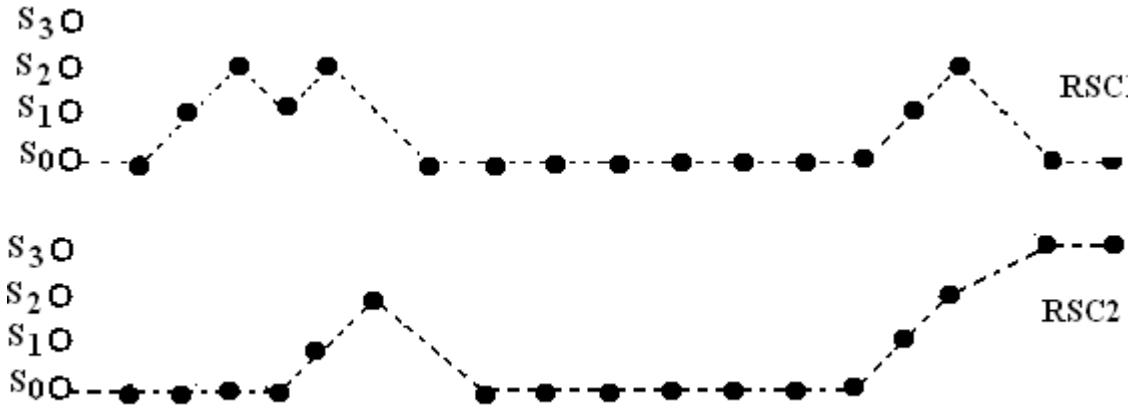


Figure II.2: Diagrammes de treillis des RSCs pour l'exemple de code turbo (5, 2, 16)

Ce simple exemple illustre plusieurs points saillants au sujet de la structure des mots code dans un turbo code. D'abord, l'entrelaceur pseudo-aléatoire permute les bits d'entrée, les deux séquences d'entrée u et u' sont presque toujours différents, cependant du même poids, et les deux codeurs veulent produire (avec la probabilité élevée) des séquences de parité de différents poids. En second lieu, on le voit facilement qu'un mot code peut se composer d'un certain nombre de détours distincts dans chaque codeur, comme illustré sur la figure II.2. Puisque les codeurs constitutifs (RSC1 et RSC2) sont réalisés sous la forme systématique récursive, un bit d'entrée non nul est exigé pour renvoyer le codeur à l'état zéro et tous les détours sont associés ainsi aux séquences d'information du poids 2 ou plus grand, au moins un pour le départ et un pour la fusion. Finalement, avec un entrelaceur pseudo-aléatoire il est fortement peu probable que les deux codeurs seront retournés à l'état zéro à la fin du mot code même lorsque le dernier (K-1) bits de la séquence d'entrée u est utilisé pour forcer le premier codeur de nouveau à l'état zéro.

Si ni l'un ni l'autre codeur n'est forcé à l'état zéro, ça veut dire qu'aucun bits de queue (tail) n'est subordonné pour terminer le premier codeur, donc la séquence de N-1 zéros suivis d'un '1' est une séquence d'entrée valide u au premier codeur. Pour quelques entrelaceurs, cette séquence sera permutée de telle sorte que $u' = u$. Dans ce cas, le poids maximum du mot code avec la perforation, et ainsi la distance libre du code, seront seulement deux. Pour cette raison, il est commun (fort probable) de terminer le premier codeur à l'état zéro. Une étude de l'ambiguïté de l'état final du deuxième codeur a été montrée par simulation dans les références [5, 6]. Des structures spéciales d'entrelaceur qui ont comme résultat des deux codeurs retournant à l'état zéro sont discutés dans les références [7], [8], et [9].

Chercher la distance libre et le spectre de distance d'un turbo code est compliqué par le fait que les codes convolutionels concaténés en parallèles sont en général des codes variables

dans le temps dus à l'entrelaceur. C'est-à-dire, pour la séquence décalée $Du(D)$ la première séquence de parité est $Dv^{(1)}(D)$, mais la séquence d'entrée au deuxième codeur (RSC2) n'est pas $Du'(D)$ (avec la probabilité élevée) dû à l'entrelaceur. Ainsi, la deuxième séquence de parité n'est pas $Dv^{(2)}(D)$.

Soit l'exemple précèdent avec une nouvelle séquence d'entrée, nous avons $\Pi_{16}(Du(D) = \{1,0,1,0,0,0,1,0,1,0,0,0,0,0,0,0\}$ et la seconde séquence de parité est donnée par $\{0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0\}$. Ainsi, le décalage temporaire des bits d'entrée résulte des mots code qui différent en position de bit et en poids total de Hamming.

II.2.1 Codeur convolutionel systématique récursif (RSC)

La figure II.3 montre un exemple simple d'un RSC. Cet codeur est dit systématique parce que l'information est émise, aussi cet codeur est dit récursif car il dispose de boucles de retour qui réinjectent le contenu de la mémoire à l'entrée du codeur.

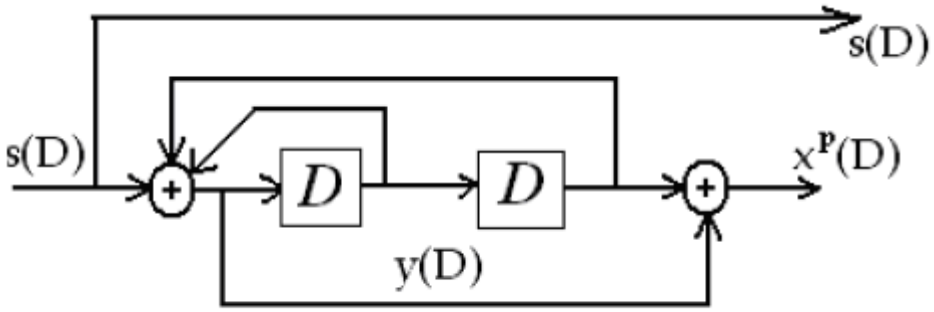


Figure II.3: Schéma d'un encodeur convolutionnel systématique récursif.

Ce type de dispositif a la particularité de disposer d'une réponse impulsionnelle $g(D)$ de durée infinie. En effet, supposons que la suite à l'entrée $s(D)$ soit une impulsion en $t = 0$, $s(D) = 1 + 0D + 0D^2 + 0D^3 + ... + 0D^k + ...$; on aura en sortie la suite $g(D)$ qu'on peut déterminer de la manière suivante :

On a

$y(D) = s(D) + Dy(D) + D^2 y(D)$ et donc $s(D) = (1 + D + D^2).y(D)$.

Par ailleurs, on a

$x^p(D) = y(D) + D^2 y(D) = (1 + D^2)y(D)$.

En éliminant $y(D)$ on trouve :

$$g(D) \stackrel{\Delta}{=} \frac{x^p(D)}{s(D)} = \frac{(1+D^2)}{(1+D+D^2)} = \frac{(1+D^3)}{(1+D^3)} \qquad \textbf{(II.1)}$$

On a

$$g(D)=\frac{(1+D)^3}{(1+D^3)}=\frac{1+D+D^2+D^3}{1+D^3}=(1+D+D^2+D^3)(1+D^3+D^6+D^9+..+D^{3k}+.. \quad \textbf{(II.2)}$$

Nous avons mis en évidence le caractère périodique de $g(D)$ (le second facteur est le développement en série de $(1+D^3)^{-1}$). Notons la similitude de ce type de machine avec un générateur de nombres aléatoires.

On peut déduire de la forme de $g(D)$ que seules les entrées qui sont des multiples de $(1+D+D^2)$ donneront une réponse de durée finie. Cela est gênant, dans la mesure ou cela signifie qu'un nombre non négligeable de messages source auront un vecteur de parité à poids faible et risquent de ce fait d'être confondus avec le message nul. (Nous savons que le poids minimal des mots de code non-nuls d'un code est égal à la distance minimale de ce code). La seule manière d'éviter ce problème serait d'augmenter la taille de la mémoire du codeur et nous savons que le prix à payer en termes de complexité de décodage est prohibitif. Cela explique la raison pour laquelle les codeurs récursifs n'ont pas intéressé beaucoup les chercheurs dans le domaine, jusqu'à la découverte par les auteurs des turbos codes d'un moyen subtil de restaurer une distance minimale importante et d'un algorithme de décodage efficace correspondant [2].

II.2.1.1 Choix d'un bon codeur (ou code)

Des recherches (ou bien des calculs) à l'aide d'ordinateur sont habituellement utilisées pour trouver les bons codes, c'est-à-dire trouver une distance libre (d_{free}) maximale avec une longueur de contrainte (K) minimale. De cette façon, les codes dans les tableaux II.1-II.8, ont été trouvés [10].

Tableau II.1 : Code convolutionnel avec R=1/2

$K-1$	$g^{(1)}$	$g^{(0)}$	d_{free}
2	5(1)	7	5
3	15	17	6
4	23	35	7
5	65	57	8
6	133	171	10
7	345	237	10
8	561	753	12
9	1161	1545	12
10	2335	3661	14
11	4335	5723	15
12	10533	17661	16
13	21675	27123	16

14	56721	61713	18
15	111653	145665	19
16	347241	246277	20

Tableau II.2 : Code convolutionnel avec R=1/3

$K-1$	$g^{(2)}$	$g^{(1)}$	$g^{(0)}$	d_{free}
2	5	7	7	8
3	13	15	17	10
4	25	33	37	12
5	47	53	75	13
6	133	145	175	15
7	225	331	367	16
8	557	663	711	18
9	1117	1365	1633	20
10	2353	2671	3175	22
11	4767	5723	6265	24
12	10533	10675	17661	24
13	21645	35661	37133	26

Tableau II.3 : Code convolutionnel avec R=2/3

$K-1$	$g_2^{(2)}, g_1^{(2)}$	$g_2^{(1)}, g_1^{(1)}$	$g_2^{(0)}, g_1^{(0)}$	d_{free}
2	3, 1	1, 2	3, 2	3
3	2, 1	1, 4	3, 7	4
4	7, 2	1, 5	4, 7	5
5	14, 3	6, 10	16, 17	6
6	15, 6	6, 15	15, 17	7
7	14, 3	7, 11	13, 17	8
8	32, 13	5, 33	25, 22	8
9	25, 5	3, 70	36, 53	9
10	63, 32	15, 65	46, 61	10

Tableau II.4 : Code convolutionnel avec R=1/4

$K-1$	$g^{(3)}$	$g^{(2)}$	$g^{(1)}$	$g^{(0)}$	d_{free}
2	5	7	7	7	10
3	13	15	15	17	13
4	25	27	33	37	16
5	53	67	71	75	18
6	135	135	147	163	20
7	235	275	313	357	22
8	463	535	733	745	24
9	1117	1365	1633	1653	27
10	2387	2353	2671	3175	29
11	4767	5723	6265	7455	32
12	11145	12477	15537	16727	33
13	21113	23175	35527	35537	36

Tableau II.5 : Code convolutionnel avec R=1/5

$K-1$	$g^{(4)}$	$g^{(3)}$	$g^{(2)}$	$g^{(1)}$	$g^{(0)}$	d_{free}
2	7	7	7	5	5	13
3	17	17	13	15	15	16
4	37	27	33	25	35	20

5	75	71	73	65	57	22
6	175	131	135	135	147	25
7	257	233	323	271	357	28

Tableau II.6 : Code convolutionnel avec R=1/6

$K-1$	$g^{(5)}$	$g^{(4)}$	$g^{(3)}$	$g^{(2)}$	$g^{(1)}$	$g^{(0)}$	d_{free}
2	7	7	7	7	5	5	16
3	17	17	13	13	15	15	20
4	37	35	27	33	25	35	24
5	73	75	55	65	47	57	27
6	173	151	135	135	163	137	30
7	253	375	331	235	313	357	34

Tableau II.7 : Code convolutionnel avec R=1/7

$K-1$	$g^{(6)}$	$g^{(5)}$	$g^{(4)}$	$g^{(3)}$	$g^{(2)}$	$g^{(1)}$	$g^{(0)}$	d_{free}
2	7	7	7	7	5	5	5	18
3	17	17	13	13	13	15	15	23
4	35	27	25	27	33	35	37	28
5	53	75	65	75	47	67	57	32
6	165	145	173	135	135	147	137	36
7	275	253	275	331	235	313	357	40

Tableau II.8 : Code convolutionnel avec R=1/8

$K-1$	$g^{(7)}$	$g^{(6)}$	$g^{(5)}$	$g^{(4)}$	$g^{(3)}$	$g^{(2)}$	$g^{(1)}$	$g^{(0)}$	d_{free}
2	7	7	5	5	5	7	7	7	21
3	17	17	13	13	13	15	15	17	26
4	37	33	25	25	35	33	27	37	32
5	57	73	51	65	75	47	67	57	36
6	153	111	165	173	135	135	147	137	40
7	275	275	253	371	331	235	313	357	45

II.2.2 Entrelacement

Un entrelaceur Π permet de modifier les indices (positions des bits en paquet d'information) des données par permutations, il représente un élément du groupe de permutation pour N éléments. Des groupes de permutation peuvent être décomposés en utilisant une seule méthode. Par exemple, $\Pi_{16} = \{15,10,1,12,2,0,13,9,5,3,8,11,7,4,14,6\}$ peut être écrit dans la décomposition de cycle de disjonction avec deux cycles :

$$\Pi_{16} = \{(0,15,6,13,4,2,1,10,8,5),(3,12,7,9)\} \quad \textbf{(II.3)}$$

Ce qui signifie simplement que les symboles sont pris par ordre comme suit : $0 \to 15 \to 6 \to 13 \to 4 \to 2 \to 1 \to 10 \to 8 \to 5 \to 0 \to 15$ dans un cycle de longueur 8, et un deuxième cycle de longueur 4. Alternativement, l'entrelaceur peut être exprimé par sa fonction d'indice.

$$\mathrm{p}(i) = j, 0 \le i, j < N \qquad \textbf{(II.4)}$$

Soit l'entrelaceur le plus commun, l'entrelaceur de bloc de taille $N = m \times n$. Ce type d'entrelaceur est basé sur l'écriture en ligne (entrée d'entrelaceur) et la lecture en colonne (sortie d'entrelaceur) d'une matrice d'information. L'entrelacement est accompli par la lecture de l'information par la colonne (sortie de l'entrelaceur). Cet entrelaceur a la fonction d'indice :

$$\mathrm{p}(i) = ni + \left[\frac{i}{m}\right] \mod N, \quad 0 \le i, j < N \qquad \textbf{(II.5)}$$

La fonction $\left(\left[\frac{i}{m}\right]\right)$ rend l'entrelaceur difficile à analyser, et un entrelaceur "linéarisé" est présenté dans la référence [11] avec la fonction d'indice :

$$\mathrm{p}(i) = ki + u \mod N, \quad 0 \le i, j < N \qquad \text{(II.6)}$$

Avec k et u sont des nombres entiers fixes.

Les entrelaceurs en bloc peuvent réaliser de bonnes valeurs de distance libre, mais ils échouent à produire un spectre des distances très étroit. Les coefficients angulaires d'approximativement N semblent fonctionner mieux, ayant pour résultat la bonne dispersion des points indexés [11].

On dispose aussi pour améliorer la performance du turbo code dans l'intervalle entre les valeurs moyennes et grandes du rapport signal sur bruit des entrelaceurs semi aléatoires [12], ce qui évite explicitement d'étaler des indices proches à l'entrée aux indices proches à la sortie. Ils sont produits de façon analogue aux entrelaceurs aléatoires, choisissant chaque incrément $\mathrm{p}(i)$ aléatoirement et testant si $|\mathrm{p}(i) - \mathrm{p}(l)| \ge T$ pour tout $i - S < l < i$, c'est-à-dire en contrôlant tous les choix précédents. Si un choix d'indice ne remplit pas cette condition, il est rejeté et un nouvel indice est choisi. Le processus continue jusqu'à ce que tous les N indices aient été choisis. Évidemment, le temps de recherche pour un entrelaceur approprié augmente avec S et T, et l'algorithme peut même ne pas se terminer avec succès. Cependant,

selon la référence [12], des valeurs de $S,T < \sqrt{N/2}$ finissent l'indexage dans un temps raisonnable.

Pour les entrelaceurs déterministes qui ont les propriétés statistiques des entrelaceurs aléatoires, Takeshita et Costello [11] ont conçu ce qu'ils ont appelé les entrelaceurs quadratiques, qui sont définis par la fonction quadratique d'indice suivante :

$$\mathrm{p}(i) = \frac{ki(i+1)}{2} \mod N \qquad \textbf{(II.7)}$$

Avec k est une constante impaire [10].

II.3 Nouveaux types d'entrelaceurs

Les entrelaceurs classiques sont habituellement conçus pour fournir une profondeur spécifique d'entrelacement. C'est efficace si les erreurs en rafale n'excèdent jamais la profondeur d'entrelaceur, mais il est inutile si l'entrelaceur est super-concu, c'est-à-dire que le nombre des erreurs produites en rafale sont beaucoup plus court que la profondeur d'entrelaceur. Par exemple, une simple matrice de 10×10 d'entrelaceur a une profondeur d'entrelacement de 10 éléments. Si une rafale de 10 erreurs se produit, le deentrelaceur écartera de façon optimale ces 10 erreurs dans tout le bloc de 100 éléments. Si l'erreur rafale est de 11 éléments, cependant, deux erreurs seront encore adjacentes. Si on a deux erreurs alors elles seront espacées de 10 éléments seulement après deentrelacement. Par contre, une matrice de 2×50 d'entrelaceur auraient espacé ces deux erreurs 50 éléments à part. Naturellement cet entrelaceur n'est pas bon pour des erreurs produites en rafales plus importantes. Dans la pratique, la plupart des canaux produisent habituellement des événements d'erreur de longueur aléatoire, et la longueur moyenne peut être variable dans le temps. Ceci rend très difficile de concevoir l'entrelaceur optimal en utilisant les approches classiques. Ce qui est recherché est une stratégie d'entrelacement qui est bonne pour un nombre d'erreurs quelconque [13].

II.3.1 Entrelacement par section d'or

II.3.1.1 Section d'or

La section d'or surgit dans beaucoup de problèmes mathématiques intéressants. La figure II.4 illustre le principe de section d'or en relation avec le problème d'entrelacement. Soit un segment de ligne de longueur 1, le problème est de diviser ce segment en longueur égal à g, et un segment plus court de longueur égal à $1-g$, tel que le rapport du segment le

plus long au segment entier est identique au rapport du segment le plus court au segment plus long.

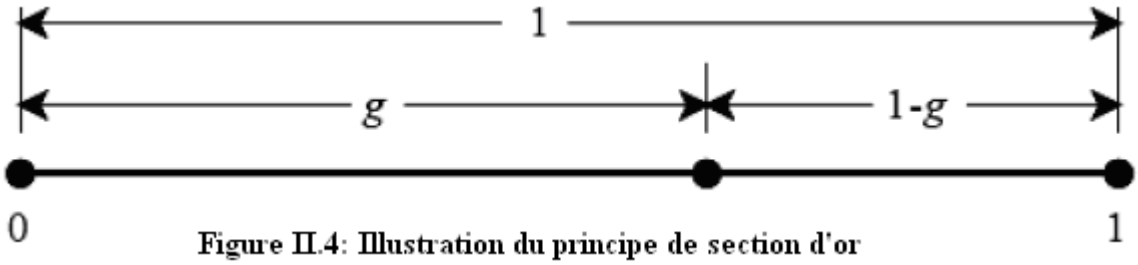


Figure II.4: Illustration du principe de section d'or

$$g = \frac{1-g}{g} \qquad \textbf{(II.8)}$$

La solution de cette équation quadratique simple pour g donne la valeur de section d'or

$$g = \frac{\sqrt{5}-1}{2} \approx 0.618 \qquad \textbf{(II.9)}$$

Considérant maintenant les points produits en commençant à 0 et en ajoutant des incréments de g, en utilisant l'arithmétique modulo 1. Après le premier incrément, on dispose de deux points à 0 et à g, en utilisant l'arithmétique modulo1. Des distances de modulo 1 sont utilisées pour tenir compte de l'option d'avoir le premier point à n'importe quelle position. De l'équation (II.8), la distance de 1-g est la même que g^2 . Après le deuxième incrément, le premier et troisième point déterminent la distance minimum et cette distance est g^3. Après le troisième incrément, le premier et le quatrième point déterminent la distance minimum et cette distance est g^4 . Le minimum de distance après le cinquième point est identique. La distance minimum après le sixième point est g^5. Les mêmes distances peuvent également être produites avec l'incrément de complément de $(1-g) = g^2 \approx 0.382$. Des puissances plus élevées peuvent également être utilisées pour la valeur d'incrément, mais les distances minimales initiales sont réduites à la valeur la plus petite d'incrément.

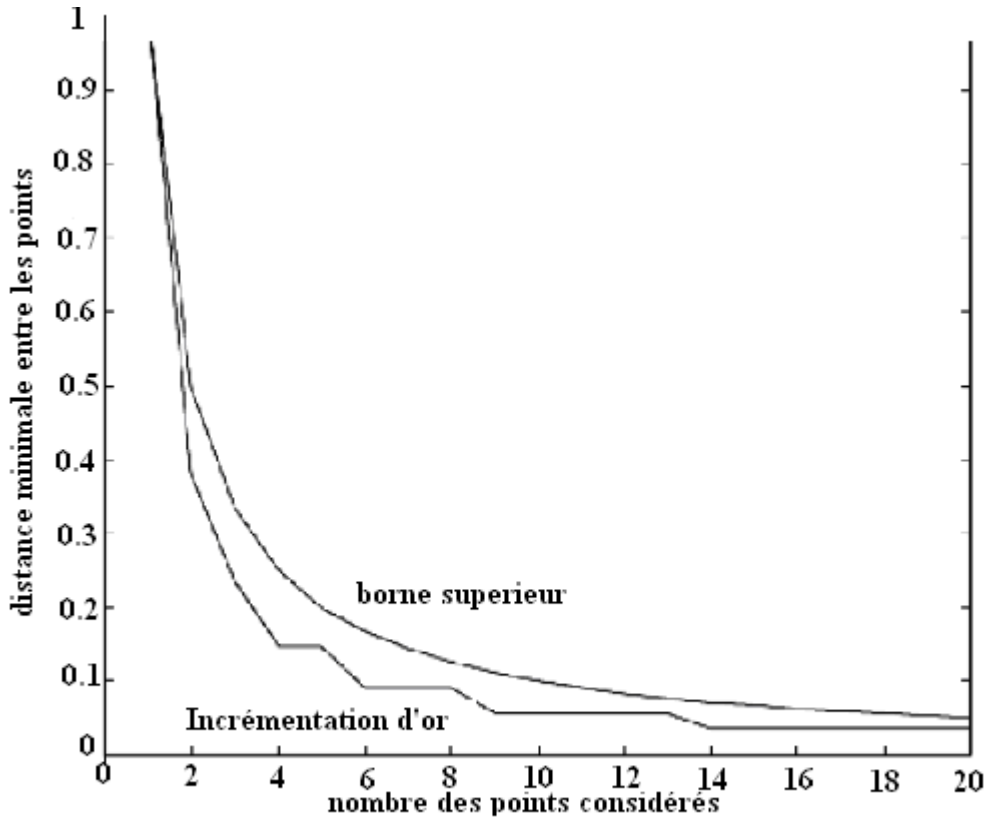


Figure II.5: Distance minimale entre les points en fonction du nombre de points avec un incrément d'or

La figure II.5 montre un tracé des distances minimales en fonction du nombre de points considérés, car des points sont ajoutés en utilisant un incrément de g avec l'arithmétique $\bmod 1$. La figure II.5 montre également une limite (borne) supérieure pour chaque nombre spécifique de points. C'est-à-dire, les points donnés de n, qui pourraient être uniformément espacés avec une distance minimum de$1/n$. Naturellement les résultats d'incrément de section d'or sont valides pour tous les nombres de points en même temps. La limite supérieure ne l'est pas. Néanmoins, les résultats d'incrément de section d'or sont montrés pour se rapprocher de la limite supérieure. Noter cela même lorsque la distance minimale chute, la plupart des points seront toujours à la même distance minimale précédente des points voisines, avec une distance moyenne entre les points égale à la limite supérieure.

Les propriétés de distance de l'incrémentation de sections d'or, illustrées sur la figure II.5, sont souhaitables pour les entrelaceurs en général, mais sont en particulier souhaitables pour les entrelaceurs de turbo code. On montre maintenant comment ces propriétés peuvent être utilisées en concevant un certain nombre d'entrelaceurs pratiques [13].

II.3.1.2 Entrelaceurs GRPI

Pour les entrelaceurs GRPI, la fonction d'indice d'entrelaceur est calculée comme suit :

$$\mathrm{p}(i) = s + ip \mod N, \quad i = 0,....,N-1 \qquad \textbf{(II.10).}$$

Où s est un nombre entier utilisé comme valeur initiale, p est un indice d'incrément entier, et N est la longueur d'entrelaceur. N et p doivent être des nombres relativement primaires pour s'assurer que chaque élément est lu une fois et seulement une fois. L'indice d'initialisation s est habituellement placé à 0, mais d'autres valeurs de nombre entier s peuvent être choisies. L'indice primaire relatif p est choisi "proche" d'une des valeurs du nombre non entier c.

$$c = N(g^m + j)/r \qquad \textbf{(II.11)}$$

Où g est la valeur de section d'or, m est un entier positif plus grand que zéro, r est l'indice d'espacement (distance) entre les éléments voisins à étaler au maximum, et j est un nombre entier $\mathrm{mod}\, r$. Comme il a été montré précédemment, les valeurs préférées pour m sont en général 1 ou 2. Dans une mise en place typique où des éléments adjacents doivent être au maximum étalés, j est placé à 0 et r à 1 [13].

II.3.1.3 Entrelaceurs GI

Les entrelaceurs GI (entrelaceur d'or) n'utilisent pas le nombre entier primaire et l'arithmétique de modulo du nombre entier, mais plutôt ils utilisent le tri des nombres à valeurs réelles dérivées de la section d'or. La première étape est de calculer la valeur de section d'or g. La deuxième étape est de calculer la valeur réelle c d'incrément définie par (II.11). La troisième étape est de produire le vecteur d'or v à valeurs réelles. Les éléments de v sont calculés comme suit:

$$v(u) = s + nc \mod N, \; n = 0,....,N-1 \qquad \textbf{(II.12)}$$

Où s est n'importe quelle valeur initiale réelle. L'étape suivante consiste à trier (ordre décroisant des valeurs) le vecteur d'or v et de trouver le vecteur d'indice z qui définit ce tri. C'est-à-dire, trouver le vecteur z de tri tels que $a(n) = v(z(n))$, $n = 0...N-1$ ou $a = tri(v)$. Les indices de l'entrelaceur d'or sont alors données par $\mathrm{p}(z(n)) = n, n = 0...N-1$. En fait, le vecteur z est l'entrelaceur inverse pour π.

La valeur initiale de s est habituellement placée à 0, mais d'autres valeurs réelles de s peuvent être choisies. Les valeurs préférées pour m sont en général 1 ou 2. Pour étaler au maximum les éléments adjacents, j est placé à 0 et r à 1 [13].

II.3.1.4 Entrelaceurs DGI

On a trouvé que pour les turbos codes, les entrelaceurs possédant un aspect aléatoire tendent à des performances meilleures que des entrelaceurs complètement structurés, particulièrement pour les blocs de grande taille de l'ordre de 1000 bits ou plus. Cependant, les propriétés de l'étalement de l'entrelaceur d'or sont toujours très souhaitables afin de maintenir une bonne distance minimum et assurer la convergence rapide en étalant efficacement l'erreur rafale dans tout le bloc. Ces deux dispositifs sont entourés dans l'entrelaceur d'or perturbé (dithered golden interleaver). La seule différence entre l'entrelaceur GI et l'entrelaceur DGI d'or perturbé est l'inclusion d'un vecteur d de perturbation, dans le vecteur d'or v. C'est-à-dire,

$$v(n) = s + nc + d(n) \quad \bmod N \quad n = 0,...,N-1 \qquad \textbf{(II.13)}$$

Où $d(n)$ est le n-ième composant de vecteur perturbation. La perturbation supplémentaire est uniformément distribuée entre 0 et ND, où D est la largeur normale de la distribution de perturbation. Le vecteur d'or perturbé v est trié (ordre décroisant des valeurs) et les indices d'entrelaceur sont produits d'une façon semblable à l'entrelaceur d'or (GI) décrit précédemment.

On a trouvé expérimentalement, pour les turbos codes et pour n'importe quelle longueur de bloc, que D est prise égale à 0.01.

On remarque que l'entrelaceur d'or (GI) est maintenant un cas particulier de l'entrelaceur d'or perturbé (DGI) avec D = 0 [13].

II.3.2 Entrelaceurs type canal

L'entrelaceur aléatoire utilisé habituellement en turbo code suit la loi de distribution uniforme. C'est-à-dire, pour une variable aléatoire X définie dans un intervalle $[a,\ b]$, sa fonction de densité de probabilité (PDF) illustrée dans la figure II.6 est donnée par:

$$f_x(x) = \begin{cases} \dfrac{1}{b-a} & a \le x \le b \\ 0 & ailleurs \end{cases} \qquad \textbf{(II.14)}$$

Remarque: *X* représente la fonction d'indice d'entrelaceur.

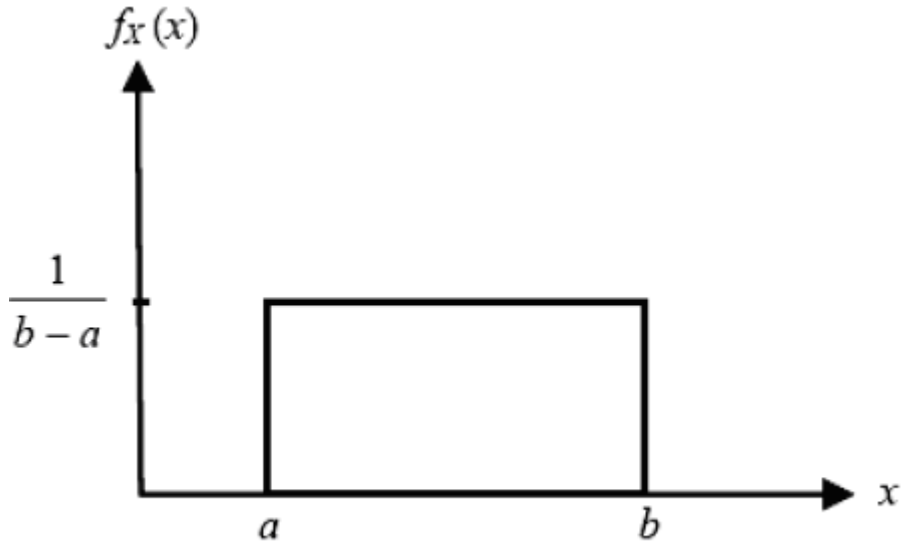


Figure II.6: Fonction de densité de probabilité uniforme

La fonction de distribution de X , montrée en figure II.7, est donnée par:

$$F_x(x) = P(X \le x) = \int_{-\infty}^{x} f_x(x)du = \begin{cases} 0 & x < a \\ \dfrac{x-a}{b-a} & a \le x \le b \\ 1 & x \ge b \end{cases} \qquad \textbf{(II.15)}$$

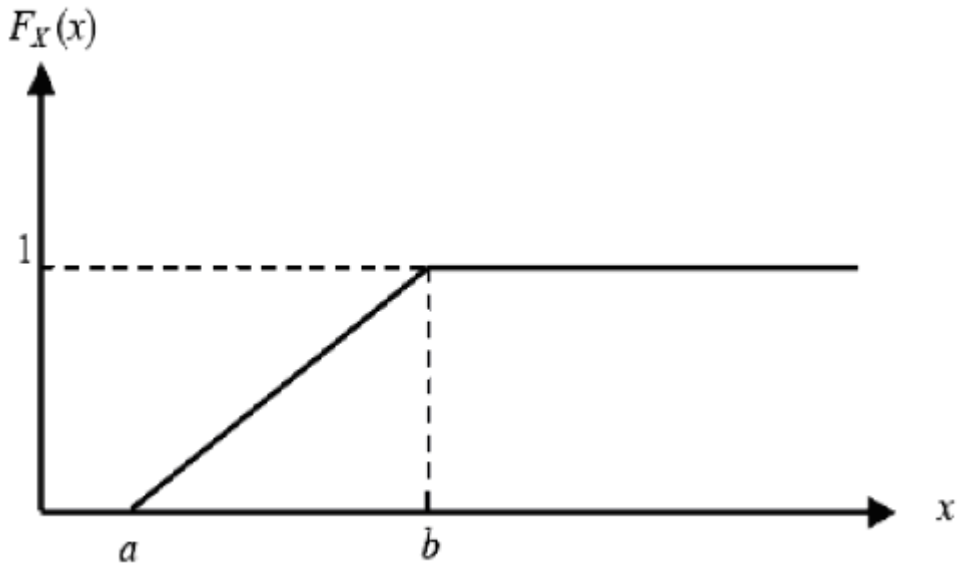


Figure II.7: Fonction de distribution de variable aléatoire uniforme

La moyenne et la variance de X sont respectivement,

$$E[X] = \mu = \frac{1}{2}(a+b) \quad \textbf{(II.16)}$$

$$\sigma_x^2 = \frac{1}{12}(b-a)^2 \quad \textbf{(II.17)}$$

Nous suggérons de faire un type d'entrelaceur dont sa fonction d'indice suit la loi de distribution du type de canal de transmission. C'est-à-dire, si le canal est Gaussien par exemple, on aura une fonction d'indice qui suit la loi de distribution Gaussienne, le même principe s'applique s'il s'agit du canal de Rayleigh.

II.4 Distribution normale (Gaussienne)

Sa fonction de densité de probabilité (PDF) montrée en figure II.8, est donnée par:

$$f_x(x) = \frac{}{\sigma.\sqrt{2\pi}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \ , \ \forall x \quad \textbf{(II.18)}$$

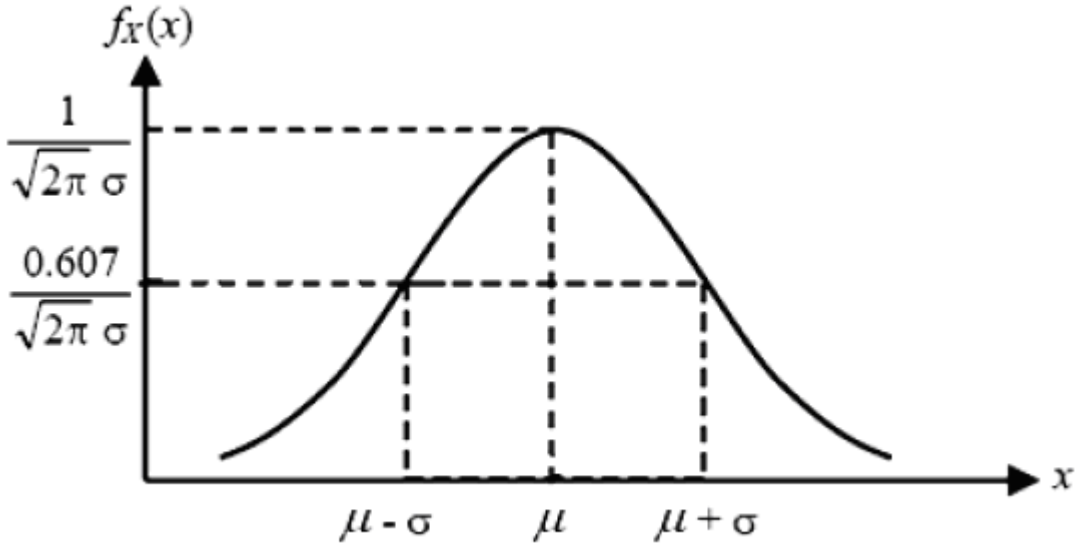


Figure II.8: Fonction de densité de probabilité Gaussienne

Où μ et σ sont respectivement, la moyenne et l'écart type (racine carré de la variance) de X et satisfont les conditions $-\infty < \mu < \infty$ et $\sigma > 0$. La fonction de distribution montrée dans la figure II.9, est donnée par:

$$F_X(x) = P(X \le x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left[-\frac{(u-\mu)^2}{2\sigma^2}\right] du \quad \textbf{(II.19)}$$

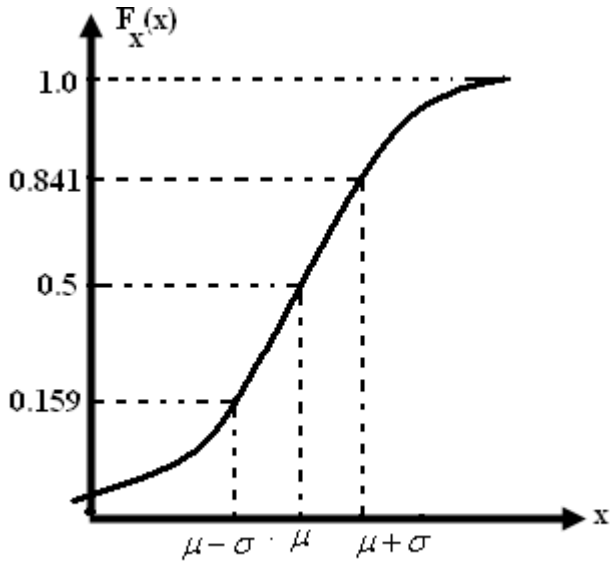


Figure II.9: Fonction de distribution Gaussienne

II.5 Distribution de Rayleigh

Soit $X = X_1^2 + X_2^2$, où X_1 et X_2 deux variables aléatoires Gaussiennes statistiquement indépendantes, avec $m = 0$ et une variance égale à s^2.

Soit $Y = \sqrt{X} = \sqrt{X_1^2 + X_2^2}$, la fonction de densité de la probabilité de Rayleigh montré dans la figure II.10, est donnée par:

$$f_y(y) = \begin{cases} \frac{y}{s^2}\exp(-\frac{y^2}{2s^2}) & y \geq 0 \\ 0 & \textit{ailleurs} \end{cases} \qquad \textbf{(II.20)}$$

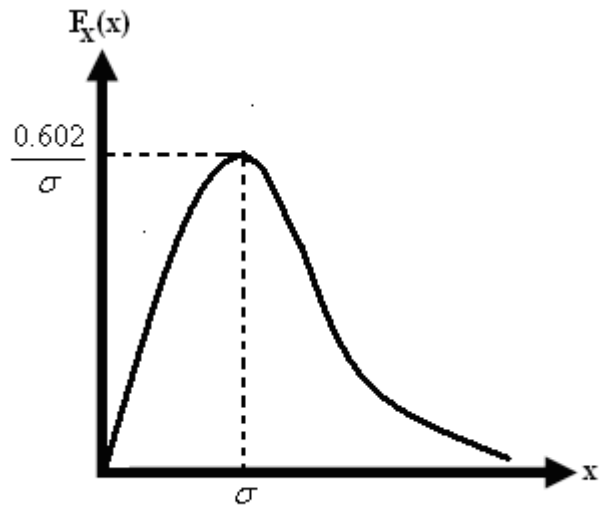


Figure II.10:Fonction de densité de probabilité de Rayleigh

La fonction de distribution, comme elle est montrée dans la figure II.11, est donnée par :

$$F_Y(y) = \begin{cases} 1 - \exp(-\frac{y^2}{2\sigma^2}) & y \geq 0 \\ 0 & \textit{ailleurs} \end{cases} \quad \textbf{(II.21)}$$

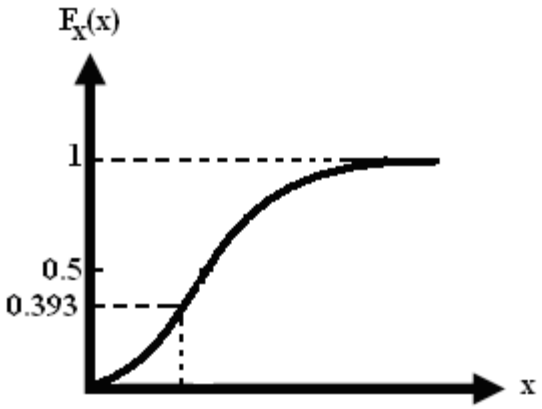


Figure II.11:Fonction de distribution de Rayleigh

Où la moyenne $\mu = E[Y] = \sigma\sqrt{\frac{\pi}{2}}$ et la variance $\sigma_y^2 = \text{var}(Y) = (2 - \frac{\pi}{2})\sigma^2$

II.6 Perforation

Différents débits de code sont réalisés en perforant les séquences x_{k1}^P et x_{k2}^P de bit de parité. La perforation de x_k^s, séquence de bits d'informations, mène à une dégradation des performances du turbo code. La figure II.12 illustre le processus de perforage. Un nombre '1' dans les tables représente un bit de code qui est inclus dans la séquence de bits transmis, et un nombre '0' représente un bit de code qui est exclu ou perforé. Du côté droit de chaque table est la liste des bits de code qui sont inclus dans la séquence transmise du code.

- En a), le code est non perforé et de taux de 1/3.
- En b), les bits de parité alternatifs de chaque encodeur sont perforés chaque fois à l'instant k. Le résultat est un taux de code égal à 1/2.
- En c), le code est perforé pour former un code de taux de 3/4 [14].

a) Taux 1/3 de Code Turbo

	K=0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**….**
x_K^S	1	1	1	1	1	1	1	1	
$x_{K_1}^P$	1	1	1	1	1	1	1	1	
$x_{K_2}^S$	1	1	1	1	1	1	1	1	

$\Rightarrow$

K=0			K=1			K=2	
x_0^S	$x_{0_1}^P$	$x_{0_2}^P$	x_1^S	$x_{1_1}^P$	$x_{1_2}^P$	x_2^S	$x_{2_1}^P$

b) Taux 1/2 de Code Turbo

	K=0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**…**
x_K^S	1	1	1	1	1	1	1	1	
$x_{K_1}^P$	1	0	1	0	1	0	1	0	
$x_{K_2}^S$	0	1	0	1	0	1	0	1	

⇒

K=0		K=1		K=2		K=3	
x_0^S	$x_{0_1}^P$	x_1^S	$x_{1_2}^P$	x_2^S	$x_{2_1}^P$	x_3^S	$x_{3_2}^P$

c) Taux 3/4 de Code Turbo

	K=0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**…**
x_K^S	1	1	1	1	1	1	1	1	
$x_{K_1}^P$	1	0	0	0	0	0	1	0	
$x_{K_2}^S$	0	0	0	1	0	0	0	0	

⇒

K=0		K=1	K=2	K=3		K=4	K=5
x_0^S	$x_{0_1}^P$	x_1^S	x_2^S	x_3^S	$x_{3_2}^P$	x_4^S	x_5^S

Figure II.12 : Principe de perforation

II.7 Terminaisons

Les turbos codes considérés sont des codes en blocs, de longueur N égale à la taille de l'entrelaceur. On a obligé de forcer les deux RSCs de retourner à l'état initial 0 par une séquence de bits appelés queue, ajoutée à la séquence de bits d'information.

Il est clair que le fait d'ajouter une séquence de queue force l'état final du premier codeur (RSC_1) à 0. Cependant, à cause d'entrelacement, l'état final du deuxième codeur (RSC_2) ne sera pas connu. Des études sur cet aspect sont présentées dans [8, 9]. Cependant, on a trouvé expérimentalement que le non prise en compte de l'état final du deuxième codeur mène aux différences non significatives en termes de performance du décodeur. Il suffit d'initialiser uniformément les probabilités des états finaux en treillis pour le deuxième décodeur (SISO2) dans l'exécution de l'algorithme du décodage [15].

III. Turbo décodeur

III.1 Introduction

La structure générale d'un turbo décodeur itératif est montrée dans la figure III.1.Deux décodeurs SISO sont liés par des entrelaceurs dans une structure semblable à celle du turbo codeur. Chaque décodeur reçoit trois entrées sont : les bits systématiquement codés, les bits de parité transmis du codeur associé et l'information de l'autre décodeur au sujet des valeurs probables des bits concernés. Cette information de l'autre décodeur est désignée sous le nom de l'information à priori. Les décodeurs doivent exploiter les deux ; les entrées soft par le canal et l'information à priori. Ils doivent également fournir des sorties soft pour les bits décodés. Les décodeurs doivent également donner les probabilités associées pour chaque bit qu'il a été correctement décodé. Les sorties soft sont typiquement représentées en termes appelés rapports de logarithme de vraisemblance (LLRs). Ces LLRs sont décrits dans la section III.2. Deux décodeurs appropriés sont l'algorithme de Viterbi de Sortie Soft (SOVA) [16] et l'algorithme de MAP [17], qui sont décrits dans les sections III.6 et III.3 respectivement.

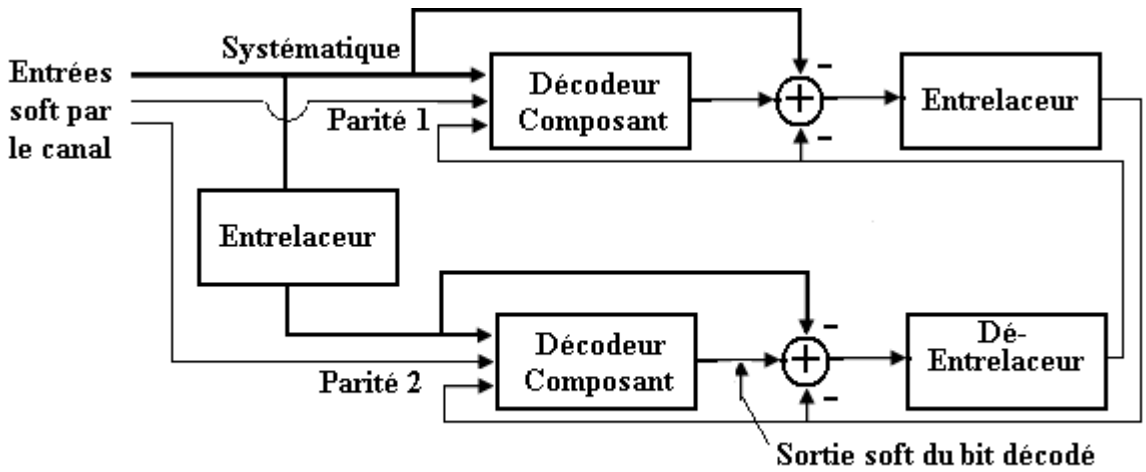


Figure III.1: Shéma d'un Turbo décodeur

Le décodeur de la figure III.1 fonctionne itérativement. Dans la première itération, le premier décodeur prend seulement les valeurs d'entrées soft par le canal (bits systématiques, bits de parité et les LLRs d'information à priori mis à zéro), et fournit une sortie soft exprimant une estimation des bits d'information. La sortie soft du premier décodeur est alors utilisée comme étant une information additionnelle pour le deuxième décodeur, ce dernier utilise cette information avec les bits reçus pour estimer les bits d'information.

Dans la deuxième itération, le premier décodeur décode les bits reçus avec des informations additionnelles sur les valeurs des bits d'entrée fournis par la sortie du deuxième décodeur dans la première itération. Cette information additionnelle permet au premier

décodeur d'obtenir une sortie soft plus précise, qui est alors utilisée par le deuxième décodeur comme étant information à priori. Ce cycle est répété, et on remarque qu'à chaque itération le taux d'erreur par bit (BER) s'améliore. Après un certain nombre d'itération le BER reste constant.

En raison de la nature itérative du décodage, le concept d'information extrinsèque et intrinsèque a été utilisé dans le travail original de Berrou décrivant le décodage itératif des turbos code [18].

Dans ce qui suit, les concepts et les algorithmes utilisés dans le décodage itératif des turbos code seront abordés [19].

III.2 Rapports de Logarithme de vraisemblance

Le concept des rapports de logarithme de vraisemblance (LLRs) a été montré par Robertson [6] pour simplifier le passage d'information du premier décodeur au deuxième décodeur dans l'opération du décodage itératif des turbos code. Le LLR d'un bit u_k des données est dénoté par $L(u_k)$, est défini pour être simplement le logarithme du rapport des probabilités du bit prenant ses deux valeurs possibles, on a alors :

$$L(u_k) \stackrel{\Delta}{=} \ln\left(\frac{P(u_k=+1)}{P(u_k=-1)}\right) \qquad \textbf{(III.1)}$$

Noter que les deux valeurs possibles pour le bit u_k sont prises pour être +1 et -1, plutôt que 1 et 0. Cette définition des deux valeurs d'une variable binaire ne fait aucune différence conceptuelle, mais elle simplifie légèrement les mathématiques dans les dérivations qui suivent.

Etant donné le LLR $L(u_k)$, il est possible de calculer la probabilité de $u_k=+1$ ou $u_k=-1$ comme suit. Rappelons que $P(u_k=-1)=1-P(u_k=+1)$ et en prenant l'exposant des deux côtés dans l'équation III.1, nous pouvons écrire :

$$e^{L(u_K)} = \frac{P(u_k=+1)}{1-P(u_k=+1)} \qquad \textbf{(III.2)}$$

Ainsi :

$$P(u_k=+1) = \frac{e^{L(u_K)}}{1+e^{L(u_K)}} = \frac{1}{1+e^{-L(u_K)}} \qquad \textbf{(III.3)}$$

De la même façon ;

$$P(u_K=-1) = \frac{1}{1+e^{+L(u_K)}} = \frac{e^{-L(u_K)}}{1+e^{-L(u_K)}} \qquad \textbf{(III.4)}$$

Et par conséquent, nous pouvons écrire :

$$P(u_K = \pm 1) = \left(\frac{e^{-L(u_K)/2}}{1 + e^{-L(u_K)}} \right) e^{\pm L(u_K)/2} \qquad \textbf{(III.5)}$$

Le terme entre parenthèse dans cette équation ne dépend pas de si nous sommes intéressés par la probabilité que $u_k = +1$ ou $u_k = -1$. Ainsi, ce terme peut être pris comme une constante, comme nous allons le voir dans la section III.3 où nous utilisons cette équation dans la dérivation de l'algorithme de MAP.

Comme le LLR $L(u_k)$ est basé sur les probabilités sans conditions $P(u_k = \pm 1)$, nous sommes également intéressés par le calcul des LLRs basé sur des probabilités conditionnelles. Par exemple, dans la théorie du codage de canal nous sommes intéressés par la probabilité de $u_k = \pm 1$ conditionnée par la séquence reçu $\underset{*}{y}$. Le LLR conditionnel $L(u_k / \underset{*}{y})$ est défini par:

$$L(u_k / \underset{*}{y}) \overset{\Delta}{=} \ln \left(\frac{P(u_k = +1 / \underset{*}{y})}{P(u_k = -1 / \underset{*}{y})} \right) \qquad \textbf{(III.6)}$$

Les probabilités conditionnelles $P(u_k = \pm 1 / \underset{*}{y})$ sont connues comme étant des probabilités à posteriori du bit décodé u_k. Ces probabilités sont fournies par les décodeurs d'entrée soft et de sortie soft (SISO) décrits dans les sections précédentes.

Indépendamment du LLR conditionnel $L(u_k / \underset{*}{y})$ basé sur les probabilités à posteriori $P(u_k = \pm 1 / \underset{*}{y})$, nous utilisons également le LLR conditionnel basé sur la probabilité pour que la sortie du canal serait y_k étant donné que le x_k transmis correspondant au bit +1 ou -1. Ce LLR conditionnel est noté par $L(y_k / x_k)$ et est défini par :

$$L(y_k / x_k) \overset{\Delta}{=} \ln \left(\frac{P(y_k / x_k = +1)}{P(y_k / x_k = -1)} \right) \qquad \textbf{(III.7)}$$

Noter la différence conceptuelle entre les définitions de $L(u_k / \underset{*}{y})$ dans l'équation III.6 et $L(y_k / x_k)$ dans l'équation III.7.

Si nous supposons que le bit transmis $x_k = \pm 1$ a été envoyé en utilisant un canal gaussien ou un canal d'évanouissement en utilisant la modulation BPSK, alors nous pouvons écrire la probabilité de y_k à la sortie du canal comme :

$$P(y_k / x_k = +1) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{E_b}{2\sigma^2}(y_k - a)^2 \right) \quad \textbf{(III.8)}$$

Où E_b est l'énergie transmise par bit, σ^2 est la variance de bruit et a est l'amplitude d'évanouissement (nous avons a=1 pour le canal AWGN). De même, nous avons :

$$P(y_k / x_k = -1) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{E_b}{2\sigma^2}(y_k + a)^2 \right) \quad \textbf{(III.9)}$$

Par conséquent, si nous utilisons la modulation BPSK dans un canal gaussien (probablement à évanouissement), nous pourrons réécrire l'équation III.7 comme suit:

$$\begin{aligned} L(y_k / x_k) &\stackrel{\Delta}{=} \ln\left(\frac{P(y_k / x_k = +1)}{P(y_k / x_k = -1)} \right) \\ &= \ln\left(\frac{\exp\left(-\frac{E_b}{2\sigma^2}(y_k - a)^2 \right)}{\exp\left(-\frac{E_b}{2\sigma^2}(y_k + a)^2 \right)} \right) \\ &= \frac{E_b}{2\sigma^2}.4a.y_k \\ &= L_c y_k \end{aligned} \quad \textbf{(III.10)}$$

Avec :

$$L_c = 4a\frac{E_b}{2\sigma^2} \quad \textbf{(III.11)}$$

L_c est défini comme étant la valeur de fiabilité du canal, et dépend seulement du SNR et de l'amplitude d'évanouissement du canal. Par conséquent, pour la BPSK utilisée dans un canal gaussien (probablement à évanouissement), le LLR conditionnel $L(y_k / x_k)$, qui est désigné sous le nom de sortie soft du canal, est simplement y_k à la sortie du canal multiplié par la valeur de fiabilité du canal L_c .

Après avoir donné l'expression de logarithme de vraisemblance, nous procédons maintenant à décrire l'algorithme à posteriori maximum, qui est l'un des algorithmes d'entrée soft et sortie soft (SISO: soft-input soft-output), ces derniers peuvent être utilisés dans le turbo décodage itératif [19].

III.3 Algorithme MAP

III.3.1 Introduction et préliminaires mathématiques

En 1974, Bahl, Cocke, Jelinek et Raviv ont proposé un algorithme, connu sous le nom d'algorithme à posteriori maximum (MAP), pour estimer les probabilités à posteriori des états et les transitions d'une source de Markov observée et soumise au bruit sans mémoire. Cet algorithme est également devenu l'algorithme de BCJR, (prend les premières lettres des noms de ses inventeurs). Pour les codes convolutionels, l'algorithme est optimal en terme de réduire au minimum le taux d'erreur par bit décodé, à la différence de l'algorithme de Viterbi [20] qui réduit au minimum la probabilité d'un chemin incorrect par le treillis choisi par le décodeur. Néanmoins, comme indiqué par Bahl [17], dans la plupart des applications les performances des deux algorithmes seraient presque identiques.

Le MAP fait la correction bit par bit. Il examine chaque chemin possible par treillis pour décider du chemin correct (mot de code correct), ce qui traduit par une augmentation en complexité. Pour cette raison et la raison des performances presque identique avec l'algorithme de Viterbi, le MAP était négligé pour longtemps et il n'était pas utilisé jusqu'à l'arrivé des turbos codes, car il s'adapte avec la philosophie des turbos codes et le décodage itératif. C'est-à-dire, il forme un SISO qui accepte une entrée soft et il fournit une sortie soft.

Nous appliquons la règle de Bayes à plusieurs corrections dans toute cette section. Cette règle donne la probabilité conjointe de a et de b, $P(a \wedge b)$, en termes de probabilité conditionnelle de a si b donnée, comme :

$$P(a \wedge b) = P(a/b).P(b) \qquad \textbf{(III.12)}$$

Une conséquence essentiel de règle de Bayes est que :

$$P[(a \wedge b)/c] = P[a/(b \wedge c)].P(b/c) \qquad \textbf{(III.13)}$$

Ce qui peut être dérivé de l'équation III.12 en considérant $x \equiv a \wedge b$ et $y \equiv b \wedge c$ comme suit. De l'équation III.12 nous pouvons écrire :

$$\begin{aligned} P[(a \wedge b)/c] = P(x/c) &= \frac{P(x \wedge c)}{P(c)} \\ &= \frac{P(a \wedge b \wedge c)}{P(c)} = \frac{P(a \wedge y)}{P(c)} \\ &= \frac{P(a/y).P(y)}{P(c)} = P[a/(b \wedge c)].\frac{P(b \wedge c)}{P(c)} \\ &= P[a/(b \wedge c)].P(b/c) \end{aligned} \qquad \textbf{(III.14)}$$

L'algorithme MAP donne, pour chaque bit décodé u_k, la probabilité que ce bit est +1 ou -1 si la séquence reçue $\underset{*}{y}$ est donné. Comme expliqué dans la section III.2, ceci est équivalent à trouver le LLR à posteriori $L(u_k / \underset{*}{y})$, où :

$$L(u_k / \underset{*}{y}) = \ln\left(\frac{P(u_k = +1/ \underset{*}{y})}{P(u_k = -1/ \underset{*}{y})}\right) \quad \textbf{(III.15)}$$

La règle de Bayes nous permet de réécrire cette équation comme suit :

$$L(u_k / \underset{*}{y}) = \ln\left(\frac{P(u_k = +1 \wedge \underset{*}{y})}{P(u_k = -1 \wedge \underset{*}{y})}\right) \quad \textbf{(III.16)}$$

Maintenant considérons la figure III.2 qui montre les transitions possibles de l'RSC $(K = 3, \quad h_0 = 7, \quad h_1 = 5)$, que nous avons utilisée comme composant du turbo codeur dans ce travail.

Pour $K = 3$, on a quatre états possibles. Etant donné un code binaire, on a pour chaque état deux transitions possibles, une si le bit d'entrée est -1 (montrée par une ligne continue) et l'autre si le bit de sortie est +1 (montrée par une ligne discontinue). On peut voir de la figure III.2 que si l'état précédent S_{k-1} et l'état présent S_k sont connus, alors la valeur de bit d'entrée u_k causée par la transition de S_{k-1} à S_k est aussi connue.

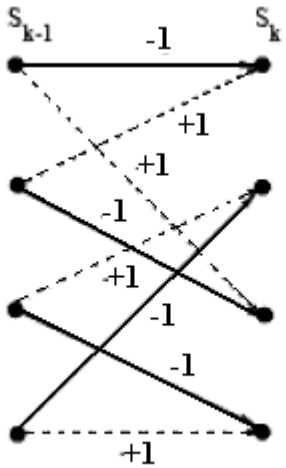


Figure III.2: Transition possibles pour un RSC de K=3

Les quatre transitions possibles de S_{k-1} à S_k quand $u_k = +1$ sont montrées avec des lignes discontinues. C'est-à-dire que ces transitions sont mutuellement exclusives (une seule peut être réalisée par le RSC). La probabilité pour que $u_k = +1$ est égale à la somme des probabilités des quatre transitions possibles causées par le bit d'entrée +1. Le même

raisonnement s'applique dans le cas de $u_k = -1$. Ainsi, on peut réécrire l'équation III.16 comme suit :

$$L(u_k / \underset{*}{y}) = \ln\left(\frac{\sum_{(S',S)_{S_K}=+1} P(S_{k-1} = s' \wedge S_k = s \wedge \underset{*}{y})}{\sum_{(S',S)_{S_K}=-1} P(S_{k-1} = s' \wedge S_k = s \wedge \underset{*}{y})}\right) \quad \textbf{(III.17)}$$

Où $(s',s)_{u_k=+1}$ est l'ensemble des transitions possibles de l'état précédent S_{k-1} à l'état présent S_k faites par l'entrée de bit +1 au RSC. Le même raisonnement s'applique pour $(s',s)_{u_k=-1}$. Pour réduire l'écriture on va remplacer l'écriture $P(S_{k-1} = s' \wedge S_k = s \wedge \underset{*}{y})$ par l'écriture $P(s' \wedge s \wedge \underset{*}{y})$.

On peut diviser la séquence $\underset{*}{y}$ en trois sections : $\underset{*}{y_k}$ le mot code reçu associé avec la transition présente, $\underset{*}{y_{j<k}}$ la séquence reçue avant cette transition et $\underset{*}{y_{j>k}}$ la séquence reçue après la transition. Cette division est montrée dans la figure III.3. Ainsi, on peut écrire la probabilité $P(s' \wedge s \wedge \underset{*}{y})$ comme suit :

$$P(s' \wedge s \wedge \underset{*}{y}) = P(s' \wedge s \wedge \underset{*}{y_{j<k}} \wedge \underset{*}{y_k} \wedge \underset{*}{y_{j>k}}) \quad \textbf{(III.18)}$$

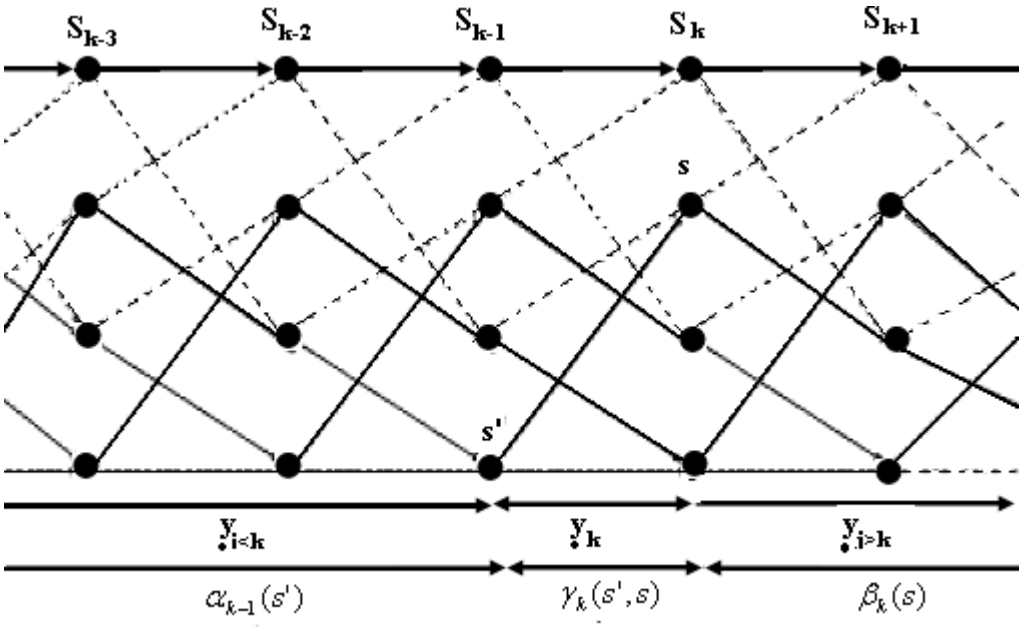


Figure III.3: Treillis d'un décodeur pour un RSC de K=3

Si on suppose que le canal est sans mémoire, alors $\underset{*}{y}_{j>k}$ ne dépend que de l'état présent s et non de s'. A l'aide de règle de Bayes $P(a \wedge b) = P(a/b).P(b)$, on peut écrire :

$$\begin{aligned} P(s' \wedge s \wedge \underset{*}{y}) &= P\left[\underset{*}{y}_{j>k} / (s' \wedge s \wedge \underset{*}{y}_{j<k} \wedge \underset{*}{y}_{k}) \right].P(s' \wedge s \wedge \underset{*}{y}_{j<k} \wedge \underset{*}{y}_{k}) \\ &= P(\underset{*}{y}_{j>k} / s).P(P(s' \wedge s \wedge \underset{*}{y}_{j<k} \wedge \underset{*}{y}_{k})) \end{aligned} \quad \textbf{(III.19)}$$

Encore avec la règle de Bayes l'équation III.19 peut se réduire à :

$$\begin{aligned} P(s' \wedge s \wedge \underset{*}{y}) &= P(\underset{*}{y}_{j>k} / s).P\left[(\underset{*}{y}_{k} \wedge s) / s' \right].P(s' \wedge \underset{*}{y}_{j<k}) \\ &= \beta_k(s).\gamma_k(s', s).\alpha_{k-1}(s') \end{aligned} \quad \textbf{(III.20)}$$

Où

$$\alpha_{k-1}(s') = P(S_{k-1} = s' \wedge \underset{*}{y}_{j<k}) \quad \textbf{(III.21)}$$

C'est la probabilité que le treillis est en état s' à l'instant $k-1$ et la séquence reçue par le canal à cet instant est $\underset{*}{y}_{j<k}$, comme il est montrée dans la figure III.3, et que :

$$\beta_k(s) = P(\underset{*}{y}_{j>k} / S_k = s) \quad \textbf{(III.22)}$$

C'est la probabilité que le treillis est en état s à l'instant k et la séquence future à recevoir par le canal est $\underset{*}{y}_{j>k}$. Finalement on a :

$$\gamma_k(s', s) = P\left[(\underset{*}{y}_{k} \wedge S_k = s) / (S_{k-1} = s') \right] \quad \textbf{(III.23)}$$

C'est la probabilité que le treillis était en état s' à l'instant $k-1$, déplaçant vers s à l'instant k et la séquence reçue par le canal pour cette transition est $\underset{*}{y}_{k}$.

L'équation III.20 montre qu'on peut diviser le calcul de probabilité $P(s' \wedge s \wedge \underset{*}{y})$ en trois calculs : α, β et γ. L'algorithme MAP permet le calcul de $\alpha_k(s)$ et $\beta_k(s)$ pour tous les états s dans tout le treillis, c'est-à-dire pour $K = 0,1,2,...,N-1,$ et $\gamma_k(s', s)$ pour toutes transitions possible de l'état $S_{k-1} = s'$ à l'état $S_k = s$, pour $K = 0,1,2,...,N-1,$. Ces valeurs sont utilisées pour trouver la probabilité $P\left(S_{k-1} = s' \wedge S_k = s \wedge \underset{*}{y} \right)$ de l'équation III.20, qui est utilisée dans l'équation III.17 pour donner $L(u_k / \underset{*}{y})$ pour chaque bit u_k. Dans ce qui suit, nous allons décrire comment le calcul des coefficients $\alpha_k(s), \beta_k(s)$ et $\gamma_k(s', s)$ se fait [19].

III.3.2 Calcul récursif en avant de $\alpha_k(s)$

De l'équation III.21 nous pouvons écrire :

$$\begin{aligned}\alpha_k(s) &= P(S_k = s \wedge \underset{*}{y}_{j<k+1}) \\ &= P(s \wedge \underset{*}{y}_{j<k} \wedge \underset{*}{y}_k) \\ &= \sum_{tous\ s'} P(s \wedge s' \wedge \underset{*}{y}_{j<k} \wedge \underset{*}{y}_k)\end{aligned} \qquad \textbf{(III.24)}$$

Utilisant la règle de Bayes et avec la supposition que le canal est sans mémoire on peut encore développer $\alpha_k(s)$ comme suit :

$$\begin{aligned}\alpha_k(s) &= \sum_{tous\ s'} P(s \wedge s' \wedge \underset{*}{y}_{j<k} \wedge \underset{*}{y}_k) \\ &= \sum_{tous\ s'} P\left[(s \wedge \underset{*}{y}_k)/(s' \wedge \underset{*}{y}_{j<k})\right].P(s' \wedge \underset{*}{y}_{j<k}) \\ &= \sum_{tous\ s'} P\left[(s \wedge \underset{*}{y}_k)/s'\right].P(s' \wedge \underset{*}{y}_{j<k}) \\ &= \sum_{tous\ s'} \gamma_k(s', s).\alpha_{k-1}(s')\end{aligned} \qquad \textbf{(III.25)}$$

Une fois que les valeurs de $\gamma_k(s', s)$ sont connu, les valeurs de $\alpha_k(s)$ seront calculées récursivement utilisant l'équation III.25. Supposant que le treillis à l'état initial $S_0 = 0$, les conditions initiales pour cette récursivité sont :

$$\begin{aligned}&\alpha_0(S_0 = 0) = 1 \\ &\alpha_0(S_0 = s) = 0 \qquad\qquad pour \quad tous\ s \neq 0\end{aligned} \qquad \textbf{(III.26)}$$

III.3.3 Calcul récursif en arrière de $\beta_k(s)$

De l'équation III.22 nous pouvons écrire :

$$\beta_{k-1}(s') = P(\underset{*}{y}_{j>k-1} / S_{k-1} = s') \qquad \textbf{(III.27)}$$

Utilisant la règle de Bayes sous la forme de l'équation III.13 et avec la supposition que le canal est sans mémoire on peut encore développer $\beta_{k-1}(s')$ comme suit :

$$
\begin{aligned}
\beta_{k-1}(s') &= P(\underset{*}{y}_{j>k-1}/s') \\
&= \sum_{tous\ s} P\left[(\underset{*}{y}_{j>k-1} \wedge s)/s'\right] \\
&= \sum_{tous\ s} P\left[(\underset{*}{y}_{k} \wedge \underset{*}{y}_{j>k} \wedge s)/s'\right] \\
&= \sum_{tous\ s} P\left[\underset{*}{y}_{j>k}/(s' \wedge s \wedge \underset{*}{y}_{k})\right].P\left[(\underset{*}{y}_{k} \wedge s)/s'\right] \\
&= \sum_{tous\ s} P(\underset{*}{y}_{j>k}/s).P\left[(\underset{*}{y}_{k} \wedge s)/s'\right] \\
&= \sum_{tous\ s} \beta_k(s).\gamma_k(s',s)
\end{aligned}
\qquad \textbf{(III.28)}
$$

Une fois que les valeurs de $\gamma_k(s',s)$ sont connus, on peut calculer $\beta_{k-1}(s')$ récursivement en arrière par les valeurs de $\beta_k(s)$ utilisant l'équation III.28. Si on suppose que le treillis se termine à l'état $S_0 = 0$, alors les valeurs initiales sont $\beta_N(0) = 1$ et $\beta_N(s) = 1$ pour tous les états $s \neq 0$, comme l'avez utilisé Berrou dans [18]. Par contre si on suppose que le treillis est sans terminaisons, c'est-à-dire que la probabilité que le treillis se termine à un état s, $\forall s$, est équiprobable. Alors les valeurs initiales sont $\beta_N(s) = 1/2^{k-1}$ [19].

III.3.4 Calcul de $\gamma_k(s', s)$

L'utilisation de la définition de $\gamma_k(s',s)$ de l'équation III.23 et la règle de Bayes sous la forme de l'équation III.13 nous donne :

$$
\begin{aligned}
\gamma_k(s',s) &= P\left[(\underset{*}{y}_{k} \wedge s)/s'\right] \\
&= P\left[\underset{*}{y}_{k}/(s' \wedge s)\right].P(s/s') \\
&= P\left[\underset{*}{y}_{k}/(s' \wedge s)\right].P(u_k)
\end{aligned}
\qquad \textbf{(III.29)}
$$

Où u_k est le bit d'entrée nécessaire pour faire la transition de $S_{k-1} = s'$ à $S_k = s$, et $P(u_k)$ est la probabilité à priori de ce bit. De l'équation III.5, ceci peut être écrit comme suit :

$$
\begin{aligned}
P(u_k) &= \left(\frac{e^{-L(u_k)/2}}{1+e^{-L(u_k)}}\right).e^{(u_k L(u_k)/2)} \\
&= C^{(1)}_{L(u_k)}.e^{(u_k L(u_k)/2)}
\end{aligned}
\qquad \textbf{(III.30)}
$$

Où

$$C_{L(u_k)}^{(1)} = \left(\frac{e^{-L(u_k)/2}}{1+e^{-L(u_k)}} \right) \quad \textbf{(III.31)}$$

Cette équation ne dépend que de $L(u_k)$ et non de la valeur de u_k (-1 ou +1).

$P\left[\underset{*}{y_k}/(s',s) \right]$ de l'équation III.29 est équivalente à $P(\underset{*}{y_k}/\underset{*}{x_k})$, où $\underset{*}{x_k}$ est le mot code transmis avec la transition de $S_{k-1} = s'$ à $S_k = s$. On suppose aussi que le canal est sans mémoire, alors :

$$P\left[\underset{*}{y_k}/(s',s) \right] = P(\underset{*}{y_k}/\underset{*}{x_k}) = \prod_{l=1}^{n} P(y_{kl}/x_{kl}) \quad \textbf{(III.32)}$$

Où x_{kl} et y_{kl} sont les bits qui composent $\underset{*}{x_k}$ et $\underset{*}{y_k}$, et n est le nombre de bits (longueur de mot code). Supposant que le bit transmis $x_{kl} = \pm 1$ a été transmis, en utilisant un canal gaussien ou un canal à évanouissement. En utilisant la modulation BPSK, nous pouvons écrire la probabilité $P(y_{kl}/x_{kl})$ comme suit :

$$P(y_{kl}/x_{kl}) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{E_b}{2\sigma^2}(y_{kl} - a x_{kl})^2 \right) \quad \textbf{(III.33)}$$

En remplaçant l'équation III.33 dans l'équation III.32 on aura :

$$\begin{aligned} P\left[\underset{*}{y_k}/(s' \wedge s) \right] &= \prod_{l=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{E_b}{2\sigma^2}(y_{kl} - a x_{kl})^2 \right) \\ &= \frac{1}{(\sigma\sqrt{2\pi})^n} \exp\left(-\frac{E_b}{2\sigma^2} \sum_{l=1}^{n} (y_{kl} - a x_{kl})^2 \right) \\ &= C_{\underset{*}{y_k}}^{(2)} . C_{\underset{*}{x_k}}^{(3)} . \exp\left(\frac{E_b}{2\sigma^2} 2a . \sum_{l=1}^{n} y_{kl} x_{kl} \right) \end{aligned} \quad \textbf{(III.34)}$$

Où

$$C_{\underset{*}{y_k}}^{(2)} = \frac{1}{(\sigma\sqrt{2\pi})^n} \exp\left(-\frac{E_b}{2\sigma^2} \sum_{l=1}^{n} y_{kl}^2 \right) \quad \textbf{(III.35)}$$

Le terme de gauche dans l'équation III.35 ne dépend que du SNR et de l'amplitude de $\underset{*}{y_k}$, et $C_{\underset{*}{x_k}}^{(3)}$ est défini comme étant :

$$C_{\underset{*}{x_k}}^{(3)} = \exp\left(-\frac{E_b}{2\sigma^2}a^2\sum_{l=1}^{n}x_{kl}^2\right)$$
$$= \exp\left(-\frac{E_b}{2\sigma^2}a^2 n\right) \quad \textbf{(III.36)}$$

Le terme $C_{\underset{*}{x_k}}^{(3)}$ ne dépend que du SNR et de l'amplitude a, donc on peut écrire $\gamma_k(s',s)$ comme suit :

$$\gamma_k(s',s) = P(u_k).P\left[\underset{*}{y_k}/(s'\wedge s)\right]$$
$$= C.e^{(u_k L(u_k)/2)}.\exp\left(\frac{E_b}{2\sigma^2}2a\sum_{l=1}^{n}y_{kl}x_{kl}\right)$$
$$= C.e^{(u_k L(u_k)/2}.\exp\left(\frac{L_c}{2}\sum_{l=1}^{n}y_{kl}x_{kl}\right) \quad \textbf{(III.37)}$$

Ainsi :

$$C = C_{L(u_k)}^{(1)}.C_{\underset{*}{y_k}}^{(2)}.C_{x_k}^{(3)} \quad \textbf{(III.38)}$$

Le terme C ne dépend pas de signe de u_k , ni de mot code transmis $\underset{*}{x_k}$. C est une constante dans la sommation en numérateur et dénominateur de l'équation III.17.

De l'équation III.17 et III.20 nous pouvons écrire :

$$L(u_k/\underset{*}{y}) = \ln\left(\frac{\sum_{(s',s)_{u_k=+1}}P(S_{k-1}=s'\wedge S_k=s\wedge \underset{*}{y})}{\sum_{(s',s)_{u_k=-1}}P(S_{k-1}=s'\wedge S_k=s\wedge \underset{*}{y})}\right)$$
$$= \ln\left(\frac{\sum_{(s',s)_{u_k=+1}}\beta_k(s).\gamma_k(s',s).\alpha_{k-1}(s')}{\sum_{(s',s)_{u_k=-1}}\beta_k(s).\gamma_k(s',s).\alpha_{k-1}(s')}\right) \quad \textbf{(III.39)}$$

Finalement, $L\left(u_k/\underset{*}{y}\right)$ est le LLR conditionnel délivré par le décodeur MAP [19].

III.3.5 Résumé des étapes pour la mise en œuvre de l'algorithme MAP

1. Initialisation des valeurs de $\alpha_0(s')$ comme il est montré dans l'équation III.26, et de $\beta_N(s)$ par $\beta_N(s) = 1/2^{k-1}\ \forall s$.
2. Calcul des valeurs de $\gamma_k(s',s)$ selon l'équation III.33,
3. Calcul des valeurs de $\alpha_{k-1}(s')$ selon l'équation III.25,

4. Calcul des valeurs de $\beta_k(s)$ selon l'équation III.28**,**

5. Finalement, le calcul des LLRs $L(u_k / \underset{*}{y})$ à posteriori délivrés par le décodeur MAP utilisant l'équation III.39.

III.4 Principes de turbo décodage itératif

III.4.1 Turbo décodage et préliminaires mathématiques

Dans cette section nous expliquons le concept de l'information extrinsèque et intrinsèque comme il a été utilisé par Berrou [18], et le décodage itératif pour les turbos code avec les algorithmes SISO (exp. MAP).

En considérant l'expression de $\gamma_k(s',s)$ dans l'équation III.37, on a :

$$\gamma_k(s',s) = C.e^{(u_k L(u_k)/2)}.\exp\left(\frac{L_c}{2}.\sum_{l=1}^{n} y_{kl}\,x_{kl}\right) \qquad \textbf{(III.40)}$$

Dans les codes dits systématiques, un des n bits transmis est le bit systématique u_k. Si nous supposons qu'il est le premier bit transmit, alors $x_{k1} = u_k$, et on peut écrire l'équation III.40 sous la forme :

$$\begin{aligned}\gamma_k(s',s) &= C.e^{(u_k L(u_k)/2)}.\exp\left(\frac{L_c}{2} y_{ks}.u_k\right).\exp\left(\frac{L_c}{2}\sum_{l=2}^{n} y_{kl}.x_{kl}\right)\\ &= C.e^{(u_k L(u_k)/2)}.\exp\left(\frac{L_c}{2} y_{ks}.u_k\right).X_k(s',s)\end{aligned} \qquad \textbf{(III.41)}$$

Où y_{ks} est la valeur réceptionnée de $x_{k1} = u_k$ et

$$X_k(s',s) = \exp\left(\frac{L_c}{2}.\sum_{l=2}^{n} y_{kl}.x_{kl}\right) \qquad \textbf{(III.42)}$$

En remplaçant l'équation III.41 dans l'équation III.39. On a :

$$
\begin{aligned}
L(u_k / \underset{*}{y}) &= \ln\left(\frac{\sum_{(s',s)_{u_k=+1}} \beta_k(s).\gamma_k(s',s).\alpha_{k-1}(s')}{\sum_{(s',s)_{u_k=-1}} \beta_k(s).\gamma_k(s',s).\alpha_{k-1}(s')}\right) \\
&= \ln\left(\frac{\sum_{(s',s)_{u_k=+1}} \beta_k(s).e^{(+L(u_k)/2)}.e^{(+\frac{L_c}{2}y_{ks})}.X_k(s',s).\alpha_{k-1}(s')}{\sum_{(s',s)_{u_k=-1}} \beta_k(s).e^{(-L(u_k)/2)}.e^{(-\frac{L_c}{2}y_{ks})}.X_k(s',s).\alpha_{k-1}(s')}\right) \\
&= L(u_k) + L_c y_{ks} + \ln\left(\frac{\sum_{(s',s)_{u_k=+1}} \beta_k(s).X_k(s',s).\alpha_{k-1}(s')}{\sum_{(s',s)_{u_k=-1}} \beta_k(s).X_k(s',s).\alpha_{k-1}(s')}\right) \\
&= L(u_k) + L_c y_{ks} + L_e(u_k)
\end{aligned}
\qquad \textbf{(III.43)}
$$

Donc :

$$
L_e(u_k) = \ln\left(\frac{\sum_{(s',s)_{u_k=+1}} \beta_k(s).X_k(s',s).\alpha_{k-1}(s')}{\sum_{(s',s)_{u_k=-1}} \beta_k(s).X_k(s',s).\alpha_{k-1}(s')}\right) \qquad \textbf{(III.44)}
$$

Ainsi nous pouvons voir que le LLR à posteriori $L(u_k / \underset{*}{y})$ calculé en utilisant l'algorithme MAP peut être considéré comme étant une somme de trois termes $L(u_k)$, $L_c y_{ks}$ et $L_e(u_k)$. Le LLR à priori $L(u_k)$ est dérivé de $P(u_k)$ dans l'expression de $\gamma_k(s',s)$ dans l'équation III.29. Cette probabilité est dérivée d'une source indépendante (émetteur) et est appelée la probabilité à priori de $k^{ième}$ bit étant +1 ou -1. Dans plusieurs cas on ne sait pas la valeur de cette probabilité, donc on prend le LLR $L(u_k)$ égale à zéro c'est-à-dire $P(u_k) = 0.5$. Cependant, en turbo décodage itératif, chaque décodeur (SISO) peut fournir à l'autre décodeur le LLR $L(u_k)$ estimé. Le second terme $L_c y_{ks}$ dans l'équation III.43 est la sortie soft du canal du bit systématique u_k, qui a été transmis directement à travers le canal et reçu comme étant y_{ks}. Le dernier terme dans l'équation III.43 $L_e(u_k)$ est extrait de la soustraction de $L(u_k)$ et $L_c y_{ks}$. C'est donc la décision propre de décodeur (algorithme) SISO. Alors $L_e(u_k)$ s'appelle le LLR extrinsèque.

L'équation III.43 montre que l'information extrinsèque $L_e(u_k)$ du décodeur MAP peut être obtenue par la soustraction de l'information à priori $L(u_k)$ et l'entrée systématique de canal $L_c y_{ks}$, de la sortie soft $L(u_k / \underset{*}{y})$ du décodeur SISO [19].

III.4.2 Turbo décodage itératif

Nous allons décrire maintenant comment le turbo décodage itératif fonctionne. La figure III.1 montre la structure de turbo décodeur itératif, elle se répète ici pour une raison conventionnelle comme étant la figure III.4.

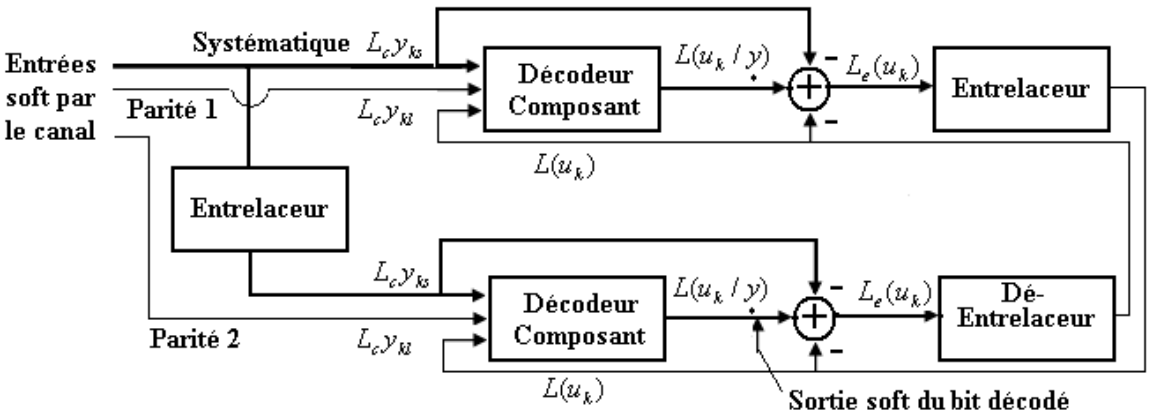


Figure III.4: Schéma d'un Turbo décodeur

Considérant premièrement le premier décodeur dans la première itération. Ce décodeur reçoit la séquence du canal $L_c \underset{*}{y}^{(1)}$ qui contienne $L_c y_{ks}$, la forme reçue des bits systématiques. $L_c y_{kl}$ sont les bits de parité du premier codeur (RSC1). Pour un taux de codage $R = 1/2$ on insert des zéros à la place des bits de parité perforés. Le premier composant décodeur (SISO1) peut traiter alors les entrées soft par canal et estimer les LLRs conditionnels $L_{11}(u_k / \underset{*}{y})$ des bits de donnée u_k, $K = 0,1,2,...,N$. L'indice 11 de $L_{11}(u_k / \underset{*}{y})$ indique qu'il est le LLR à posteriori de la première itération du premier décodeur. Notons qu'en première itération le premier décodeur n'aurait pas l'information à priori des bits, donc $L(u_k)$ dans l'équation III.37 qui donne $\mathfrak{g}_k(s', s)$ est nul. Ceci correspondant à la probabilité à priori égale à 0,5.

Deuxièmement, le second décodeur (SISO2) entre dans l'opération. Il reçoit la séquence du canal $L_c \underset{*}{y}^{(2)}$ qui contienne la forme reçue des bits systématiques entrelacés et les bits de parité du second codeur (RSC2). Si le code est perforé, on insert des zéros à la place des bits perforés. En plus de la séquence reçu du canal $L_c \underset{*}{y}^{(2)}$, le décodeur peut utiliser le LLR conditionnel $L_{11}(u_k / \underset{*}{y})$ fournit par le premier composant décodeur comme étant LLR à priori $L(u_k)$ dans le second décodeur. Après soustraction des $L_c y_{ks}$, $L(u_k)$ de SISO1 et

l'entrelacement, on obtient l'information extrinsèque $L_e(u_k)$. Ainsi le second décodeur utilise $L_c\, y_*^{(2)}$ et les LLRs à priori $L(u_k)$ (dérivé d'entrelacement de l'information extrinsèque $L_e(u_k)$ du premier décodeur) pour produire les LLRs à posteriori $L_{12}(u_k / y_*)$. Ceci est la fin de la première itération.

Pour la deuxième itération, le premier décodeur traite encore la séquence du canal reçue $L_c\, y_*^{(1)}$, avec la présence des LLRs à priori $L(u_k)$ fournis par l'information extrinsèque $L_e(u_k)$ des LLRs à posteriori $L_{12}(u_k / y_*)$ fournis par le second décodeur, d'où on peut produire un LLR à posteriori $L_{21}(u_k / y_*)$ amélioré. Le second décodeur utilisant le LLR à posteriori $L_{21}(u_k / y_*)$ amélioré du premier décodeur et délivre un LLR à priori $L(u_k)$ amélioré qui est utilisé avec la séquence reçue du canal $L_c\, y_*^{(2)}$ pour calculer $L_{22}(u_k / y_*)$. Ce processus itératif va continuer et à chaque nouvelle itération le BER moyen s'améliore [19].

L'algorithme MAP est extrêmement complexe à cause des multiplications nécessaires en équations III.25 et III.28. Dans ce qui suit, une description d'autres algorithmes SISO moins complexes que le MAP, pouvant le remplacé dans le turbo décodage itératif.

III.5 Modifications sur l'algorithme MAP

III.5.1 Introduction

L'algorithme MAP comme il est décrit en section III.3 est plus complexe que l'algorithme de Viterbi avec la décision hard (décodage après quantification). Les performances dans la plupart des cas sont identiques. Le MAP est ignoré pour plus de 20 ans. Cependant, son utilisation en turbo code a tiré l'attention à nouveau pour minimiser sa complexité. Initialement l'algorithme Max-Log-MAP est proposé par Koch et Baier [22] et Erfanian [23]. Cette technique transfère les récursivités en domaine logarithmique et invoque à une approximation qui réduit dramatiquement la complexité. A cause de cette approximation, les performances son moins bonnes par rapport aux performances de MAP. Cependant, Rebertson [24] en 1995 proposent l'algorithme Log-MAP qui donne les performances identiques à celle de l'algorithme MAP, mais avec une certaine complexité [29].

III.5.2 L'algorithme Max-Log-MAP

Le Max-Log-MAP simplifie le MAP par la transformation des ces équations en domaine logarithmique et l'utilisation de l'approximation :

$$\ln\left(\sum_i e^{x_i}\right) \approx \max_i(x_i) \qquad \textbf{(III.45)}$$

Où $\max_i(x_i)$ prend le maximum des x_i . $A_k(s)$, $B_k(s)$ et $\Gamma_k(s',s)$ sont définis comme suit :

$$A_k(s) \stackrel{\Delta}{=} \ln(\alpha_k(s)) \qquad \textbf{(III.46)}$$

$$B_k(s) \stackrel{\Delta}{=} \ln(\beta_k(s)) \qquad \textbf{(III.47)}$$

$$\Gamma_k(s',s) \stackrel{\Delta}{=} \ln(\gamma_k(s',s)) \qquad \textbf{(III.48)}$$

On peut écrire l'équation III.25 comme suit :

$$\begin{aligned} A_k(s) &\stackrel{\Delta}{=} \ln(\alpha_k(s)) \\ &= \ln\left(\sum_{tous\ s'} \gamma_k(s',s).\alpha_{k-1}(s')\right) \\ &= \ln\left(\sum_{tous\ s'} \exp[A_{k-1}(s') + \Gamma_k(s',s)]\right) \\ &\approx \max_{s'}\left(A_{k-1}(s') + \Gamma_k(s',s)\right) \end{aligned} \qquad \textbf{(III.49)}$$

Similairement, On peut écrire l'équation III.28 comme suit :

$$\begin{aligned} \beta_{k-1}(s') &\stackrel{\Delta}{=} \ln(\beta_{k-1}(s')) \\ &= \ln\left(\sum_{tous\ s} \beta_k(s).\gamma_k(s',s)\right) \\ &= \ln\left(\sum_{tous\ s} \exp[\beta_k(s) + \Gamma_k(s',s)]\right) \\ &\approx \max_{s}\left(B_k(s) + \Gamma_k(s',s)\right) \end{aligned} \qquad \textbf{(III.50)}$$

On remarque en équations III.49 et III.50 que le calcul de la probabilité d'état fait par la sélection de métrique maximale des chemins, et que la métrique du chemin est la somme de la métrique du chemin à l'instant $k-1$ plus la métrique de branche à l'instant k qui est équivalente à la récursivité de l'algorithme de Viterbi.

En utilisant l'équation III.37, la métrique de branche $\Gamma_k(s',s)$ peut être écrite comme suit :

$$\begin{aligned}\Gamma_k(s',s) &\stackrel{\Delta}{=} \ln\left(\mathfrak{g}_k(s',s)\right) \\ &= \ln\left(C.e^{(u_k L(u_k)/2)}.\exp\left(\frac{L_c}{2}\sum_{l=1}^{n} y_{kl}\,x_{kl}\right)\right) \\ &= \hat{C} + \frac{1}{2}u_k L(u_k) + \frac{L_c}{2}\sum_{l=1}^{n} y_{kl}\,x_{kl}\end{aligned} \qquad \textbf{(III.51)}$$

Où $\hat{C} = \ln(C)$ est une constante qui ne dépend pas de u_k ou bien de mot code transmit $\underset{*}{x}_k$. La métrique de branche est équivalente à celle utilisé dans l'algorithme de Viterbi, avec l'addition du LLR à priori $u_k L(u_k)$.

Finalement, de l'équation III.39 nous pouvons écrire les LLRs à posteriori $L(u_k / \underset{*}{y})$ qui sont calculés par le Max-Log-MAP comme suit :

$$\begin{aligned}L(u_k / \underset{*}{y}) &= \ln\left(\frac{\sum_{(s',s)_{u_k=+1}} \mathfrak{b}_k(s).\mathfrak{g}_k(s',s).\mathfrak{a}_{k-1}(s')}{\sum_{(s',s)_{u_k=-1}} \mathfrak{b}_k(s).\mathfrak{g}_k(s',s).\mathfrak{a}_{k-1}(s')}\right) \\ &= \ln\left(\frac{\sum_{(s',s)_{u_k=+1}} \exp\left(A_{k-1}(s') + \Gamma_k(s',s) + \mathfrak{b}_k(s)\right)}{\sum_{(s',s)_{u_k=-1}} \exp\left(A_{k-1}(s') + \Gamma_k(s',s) + \mathfrak{b}_k(s)\right)}\right) \\ &\approx \max_{(s',s)_{u_k=+1}} \left(\exp\left(A_{k-1}(s') + \Gamma_k(s',s) + \mathfrak{b}_k(s)\right)\right) \\ &\quad - \max_{(s',s)_{u_k=-1}} \left(\exp\left(A_{k-1}(s') + \Gamma_k(s',s) + \mathfrak{b}_k(s)\right)\right)\end{aligned} \qquad \textbf{(III.52)}$$

III.5.3 Algorithme Log-MAP (correction de l'approximation)

L'algorithme Max-Log-MAP donne une dégradation en performance comparée à l'algorithme MAP à cause de l'approximation de l'équation III.45. Quand elle est utilisée en turbo décodage itératif, Robertson [24] trouvait que cette dégradation est de 0.35 dB. Cependant, l'approximation de l'équation III.45 peut être rendu exacte par l'utilisation de logarithme Jacobien :

$$\begin{aligned}\ln\left(e^{x_1}+e^{x_2}\right) &= \max\left(x_1+x_2\right)+\ln(1+e^{|x_1-x_2|}) \\ &= \max\left(x_1+x_2\right)+ f_c\left(\left|x_1-x_2\right|\right) \\ &= g\left(x_1,x_2\right)\end{aligned} \quad \textbf{(III.53)}$$

Où $f_c(x)$ peut être considéré comme le terme de correction. C'est alors la base d'algorithme Log-MAP proposé par Robertson, Villerbrun et höher [24]**.** Similairement à l'algorithme Max-Log-MAP, les valeurs $A_k(s) \overset{\Delta}{=} \ln(\mathsf{a}_k(s))$ et $B_k(s) \overset{\Delta}{=} \ln(\mathsf{b}_k(s))$ sont calculées par l'utilisation d'une récursivité en avant et en arrière.

III.6 Algorithme SOVA (Soft Output Viterbi Algorithm)

III.6.1 Description de SOVA

Dans cette section nous décrivons une variante de l'algorithme de Viterbi, appelé l'algorithme de Viterbi à sortie soft (SOVA) [16, 25]. Cet algorithme a deux modifications qui permet d'être exploitable par le décodeur (SISO) du turbo décodeur. Premièrement la métrique du chemin est modifiée pour tenir compte de l'information à priori quand on sélectionne le chemin de maximum de vraisemblance (ML) à travers le treillis. Deuxièmement l'algorithme est modifié pour fournir des sorties soft sous la forme de LLR $L(u_k / \underset{*}{y})$ pour chaque bit décodé.

Pour la première modification, on considère la séquence d'états $\underset{*}{s_k^s}$ jusqu'à l'instant k dans le treillis. La probabilité que ce chemin (séquence) est correct à travers le treillis est donnée par :

$$P\left(\underset{*}{s_k^s} / \underset{*}{y_{j\le k}}\right) = \frac{P\left(\underset{*}{s_k^s} \wedge \underset{*}{y_{j\le k}}\right)}{P\left(\underset{*}{y_{j\le k}}\right)} \quad \textbf{(III.54)}$$

Parce que la probabilité de la séquence reçue $P(\underset{*}{y_{j\le k}})$ est constante pour tous les chemins à travers le treillis jusqu'à l'instant k, la probabilité que ce chemin est correct à travers le treillis est proportionnelle à $P(\underset{*}{s_k^s} \wedge \underset{*}{y_{j\le k}})$. Donc la métrique doit être définie par le fait que quand elle est maximisée, ceci implique la maximisation de $P(\underset{*}{s_k^s} \wedge \underset{*}{y_{j\le k}})$. On suppose que le canal est sans mémoire, on a :

$$P(\underset{*}{s_k^s} \wedge \underset{*}{y_{j\le k}}) = P(\underset{*}{s_{k-1}^{s'}} \wedge \underset{*}{y_{j\le k-1}}).P(S_k = s \wedge \underset{*}{y_k} / S_{k-1} = s') \quad \textbf{(III.55)}$$

Soit $M(\underset{*}{s_k^s})$ la métrique de chemin pour $\underset{*}{s_k^s}$, où :

$$\begin{aligned} M(\underset{*}{s_k^s}) &= \ln(P(\underset{*}{s_k^s} \wedge \underset{*}{y_{j\le k}})) \\ &= M(\underset{*}{s_{k-1}^{s'}}) + \ln(P(S_k = s \wedge \underset{*}{y_k} / S_{k-1} = s')) \end{aligned} \quad \textbf{(III.56)}$$

En utilisant l'équation III.23 nous avons alors :

$$M(\underset{*}{s_k^s}) = M(\underset{*}{s_{k-1}^{s'}}) + \ln(\gamma_k(s', s)) \quad \textbf{(III.57)}$$

Où $\gamma_k(s', s)$ est la probabilité de transition pour le chemin de $S_{k-1} = s'$ à $S_k = s$. De l'équation III.51 nous pouvons écrire :

$$\ln(\gamma_k(s', s)) = \Gamma_k(s', s) = \hat{C} + \frac{1}{2} u_k L(u_k) + \frac{L_c}{2} \sum_{l=1}^{n} y_{kl} x_{kl} \quad \textbf{(III.58)}$$

Puisque le terme $\hat{C}$ est constant, on peut le négligé et nous pouvons réécrire l'équation III.57 comme suit :

$$M(\underset{*}{s_k^s}) = M(\underset{*}{s_{k-1}^{s'}}) + \frac{1}{2} u_k L(u_k) + \frac{L_c}{2} \sum_{l=1}^{n} y_{kl} x_{kl} \quad \textbf{(III.59)}$$

La métrique dans l'algorithme SOVA est mise en forme comme dans le cas de l'algorithme de Viterbi, avec le terme additionnel $u_k L(u_k)$ qui est l'information à priori prise en compte. Notons que c'est équivalant à la récursivité en avant dans l'équation III.49 utilisée pour calculer $A_k(s)$ dans l'algorithme Max-Log-MAP.

On étudie maintenant la deuxième modification, c'est-à-dire la donnée issue de la sortie soft de l'algorithme de Viterbi. Dans un treillis binaire, on a deux chemins pour atteindre l'état à l'instant k. On calcule la métrique de ces deux chemins selon l'équation III.59. Si $\underset{*}{s_k^s}$ et $\underset{*}{\hat{s}_k^s}$ sont deux chemins qui peuvent atteindre l'état $S_k = s$ à l'instant k, et qui ont les deux métrique de chemins respectivement $M(\underset{*}{s_k^s})$ et $M(\underset{*}{\hat{s}_k^s})$, pour déterminer le chemin le plus correct on définit la notion de la différence de métrique Δ_k^s comme suit :

$$\Delta_k^s = M(\underset{*}{s_k^s}) - M(\underset{*}{\hat{s}_k^s}) \quad \textbf{(III.60)}$$

Alors, si $\Delta_k^s > 0$ donc le chemin $\underset{*}{s_k^s}$ est le plus probable d'être correct, et si $\Delta_k^s < 0$, alors le chemin $\underset{*}{\hat{s}_k^s}$ est le plus probable d'être correct. Δ_k^s est le LLR de bit u_k en instant k. On peut généralisé ce résultat pour avoir les LLRs $L(u_k / \underset{*}{y})$ pour tous les états comme suit :

$$L(u_k / \underset{*}{y}) = \left|\Delta_k^s\right|_{\max(M(\underset{*}{s_k^s}))} \qquad pour\ s = 0,1,2,...,2^{K-1} \qquad \textbf{(III.61)}$$

$\left|\Delta_k^s\right|_{\max(M(\underset{*}{s_k^s}))}$ est la valeur absolue de Δ_k^s pour le chemin $\underset{*}{s_k^s}$ pour obtenir $M(\underset{*}{s_k^s})$ maximum dans le treillis à l'instant k.

L'algorithme considéré dans cette section est le moins complexe des algorithmes de décodage SISO discutés dans ce chapitre. Dans [24], l'algorithme SOVA est moins complexe que l'algorithme Max-Log-MAP. Cependant, l'algorithme SOVA est aussi mois précis que les autres algorithmes décrits dans ce chapitre, et lorsqu'il est utilisé en turbo décodage itératif, les performances sont moindre de 0.6 dB [24] que le décodage qui utilise l'algorithme MAP.

III.6.2 Résumé des étapes de la mise en œuvre de l'algorithme SOVA

1. Initialisation des valeurs de $M(\underset{*}{s_0^s})\ \forall s$,
2. Calcul de $M(\underset{*}{s_k^s})$ et $M(\underset{*}{\hat{s}_k^s})\ \forall s$ en utilisant l'équation III.59,
3. Calcul de Δ_k^s en utilisant l'équation III.60;
4. Calcul des LLRs $L(u_k / \underset{*}{y})$ à posteriori délivrés par le décodeur SOVA en utilisant l'équation III.61.

III.7 Conclusions

Dans ce chapitre, nous avons décrit les techniques utilisées en turbo décodage itératif. On s'est intéressé surtout à l'algorithme de décodage, ou on a décrit quatre algorithmes : le MAP, Log-MAP, Max-Log-MAP et le SOVA. L'algorithme MAP semble meilleur, mais il est extrêmement complexe à cause du nombre de multiplications présentes dans les équations III.25, III.28 et III.37. L'algorithme Log-MAP réduit la complexité de l'algorithme MAP par sa transformation en domaine logarithmique (réduit l'effet des multiplications) en gardant les mêmes performances. L'algorithme Max-Log-MAP simplifie la complexité de MAP par l'approximation de l'équation III.45, mais ceci dégrade les performances de 0.35 dB [24] par rapport à l'algorithme MAP. Le dernier algorithme SOVA simplifie lui aussi la complexité de l'algorithme MAP, mais il présente une dégradation en performances de 0.6 dB [24] par rapport à l'algorithme MAP.

IV. Simulation et Résultats

IV.1.Conditions et paramètres de la simulation

Nous avons fait nos simulations dans les conditions suivantes :

- Un simulateur d'un système de communication numérique programmé en Matlab a été utilisé comme base de nôtres simulations sur les turbos codes. Une partie des programmes de ce simulateur est inspirée des programmes de Yufei [26]. Ce simulateur a été programmé en Matlab dans le but de tester rapidement est efficacement les changements en cours.
- On peut distinguer trois parties dans la simulation : le codeur, le canal et le décodeur.

La simulation du turbo (codeur/décodeur) est basée sur le schéma suivant :

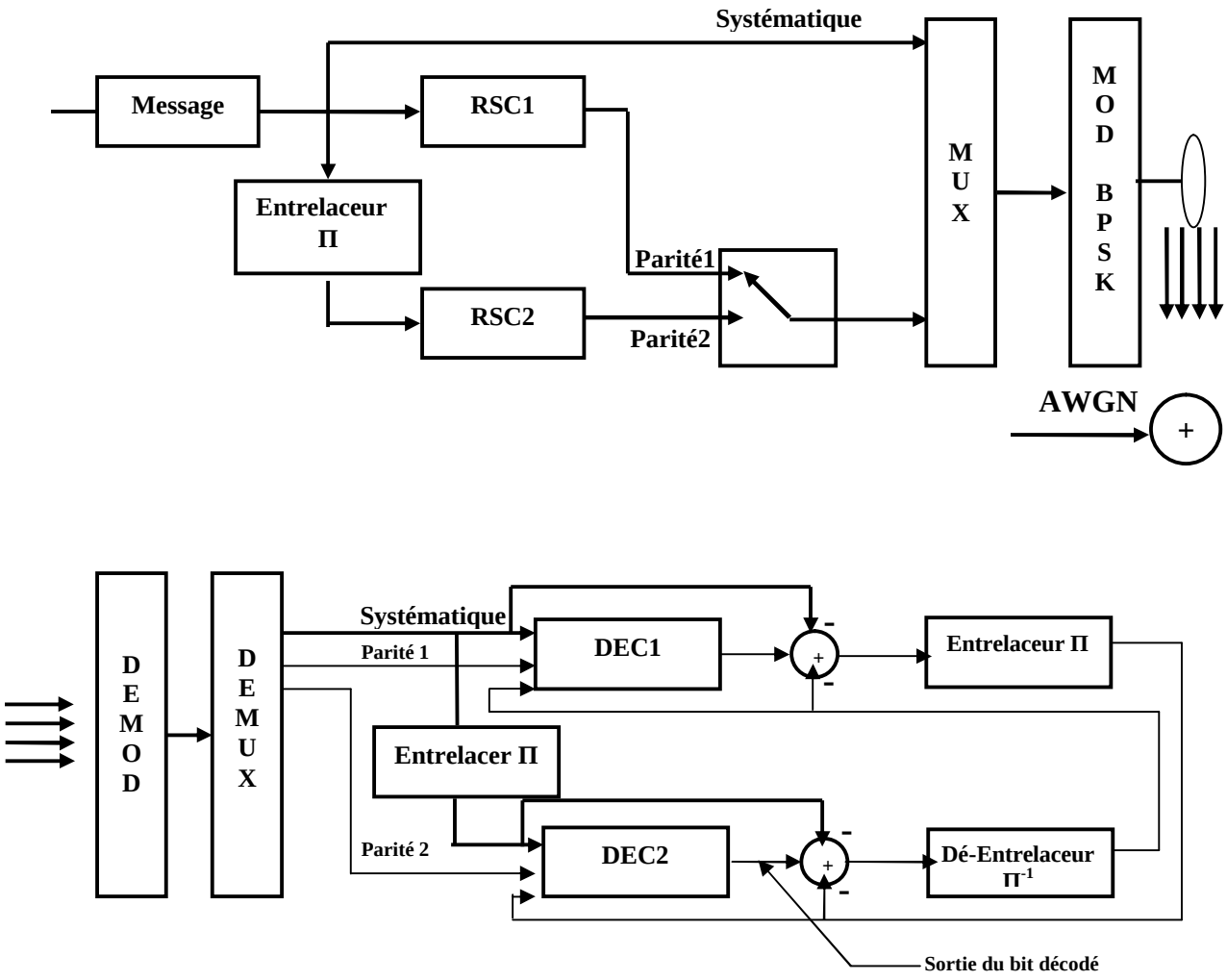


Figure IV.1 : Turbo codec appliqué au canal gaussien.

- Le codeur simulé est composé de deux RSC identiques concaténés en parallèle et séparés par un entrelaceur.
- On a utilisé des blocs de longueurs variables *N* (appelée aussi taille d'entrelaceur). On a pris *N=169* bits, *N=1024* bits (paquets courts) et *N=3600* bits (paquets longs). Cependant, un paquet de 169 bits est souhaitable pour la transmission de parole et 1000 bits est souhaitables pour la transmission vidéo selon la référence [19]. On a pris *N=1024* au lieu de 1000 bits. On a pris aussi *N=3600* comme exemple de transmission de données qui n'exige pas le temps réel.
- L'ordre des polynômes générateurs de codes convolutifs n'a pas d'importance en terme de performances des codes, c'est-à-dire prendre (g_0, g_1) est équivalent à celui de (g_1, g_0). Par ailleurs, cet ordre est très important dans le cas des turbos codes en terme de dégradation des performances démontrées dans la référence [19]. Pour cela on a utilisé les polynômes générateurs du tableau II.1 pour *K=3*. D'après ce tableau (section II.2.1.1 du chapitre II), le choix est optimal en considérant la distance libre minimale optimale.
- On a utilisé l'entrelaceur aléatoire utilisé par Berrou [18],
- On a utilisé deux valeurs de taux du code *R=1/3*(sans perforation) et *R=1/2* (avec perforation) pour voir l'effet de la variation de *R* sur les performances des turbos codes.
- Il y a différentes méthodes de terminaison pour forcer les deux RSCs de retourner à l'état initial 0. On a utilisé la méthode ajoutant des séquences de bits comme étant une queue au premier RSC seulement, pour les raisons citées dans la section II.7 du chapitre II.
- Dans notre simulation, on a utilisé le canal AWGN. Ce dernier est un bon modèle pour représenter les différents types de systèmes de communication surtout où la liaison radioélectrique est directe (canaux satellitaires par exemple).
- La simulation de turbo décodeur est basée sur la description évoquée dans le chapitre III. Le décodeur simulé est composé de deux SISOs utilisant le même algorithme de décodage, concaténés en parallèle et séparés par un de-entrelaceur des bits. On a utilisé deux types d'algorithmes de décodage : le Log-MAP qui est une variante de l'algorithme MAP utilisé habituellement dans le turbo décodage itératif, et le second algorithme SOVA qui est une modification de l'algorithme de Viterbi pour avoir des décisions soft (sorties soft) afin de l'adapter au turbo décodage itératif.
- Notons aussi que dans nos calculs du BER en fonction de E_b/N_0, on a fixé le nombre de blocs erronés à 15 pour arrêter la simulation.

IV.2. Simulation du système de communication dans un canal AWGN

IV.2.1. Effet de nombre d'itération sur les performances des turbos codes

Dans la figure IV.2 (a, b) on a choisi N=1024(longueur du bloc), entrelaceur aléatoire, décodeur Log-MAP et 15 blocs erronés.

Comme il est mentionné dans la section III.4, le nombre d'itérations est un facteur clé dans la détermination des performances des turbos codes. Il permet à partir du processus de décodage itératif, l'amélioration continue de l'information à priori $L(u_k)$ des bits d'information. Il est clair d'après la figureIV.2(a, b) que l'augmentation du nombre d'itérations est suivie d'une amélioration considérable du taux d'erreurs binaire (BER) du turbo code considéré de l'ordre de 10^{-2} dans la première itération, pour atteindre l'ordre de 10^{-5} dans la cinquième itération pour un $Eb/N0 = 2.0dB$. Et de l'ordre de 10^{-6} pour $Eb/N0 = 3.0dB$ (voire figureIV.2 (a, b)).

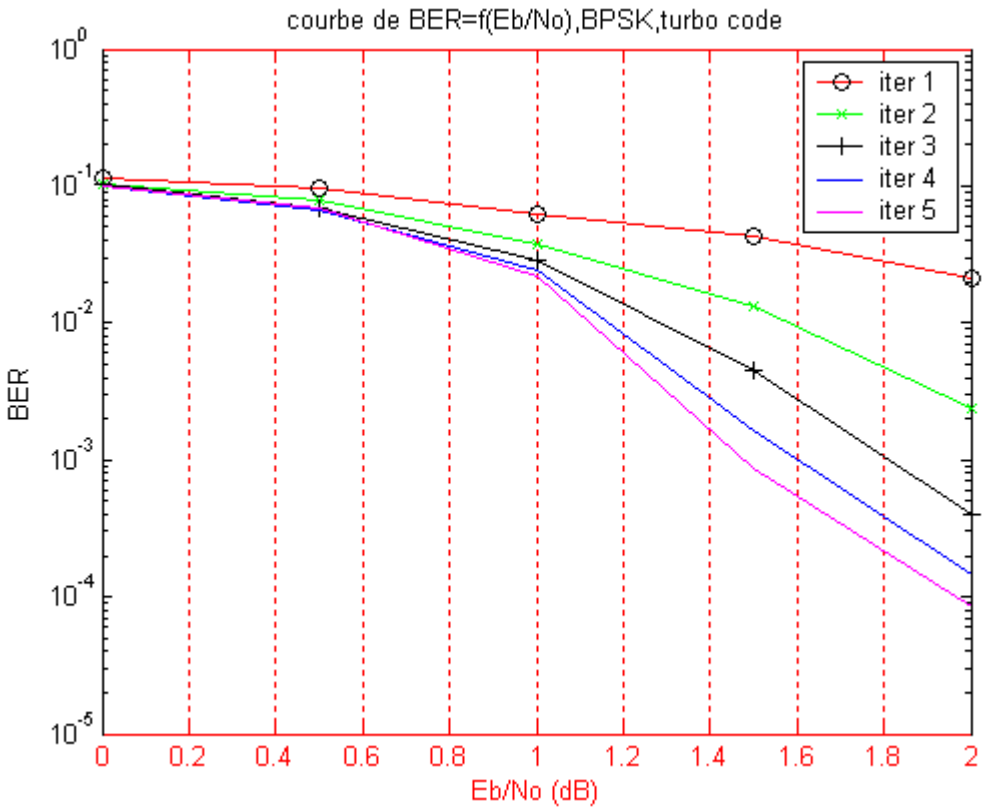


Figure IV.2.a : détermination du BER en fonction de Eb/N0=2dB pour 5 itérations en utilisant l'algorithme de décodage Log-MAP, N=1024.

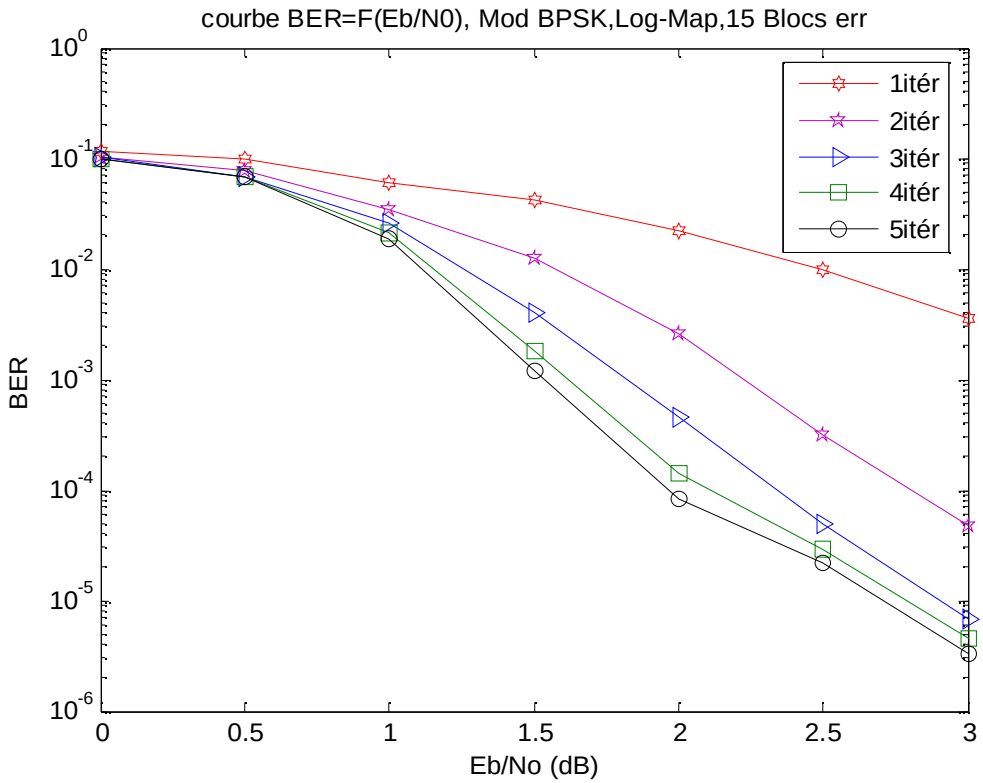


Figure IV.2.b : détermination du BER en fonction de Eb/N0=3dB pour 5 itérations en utilisant l'algorithme de décodage Log-MAP, N=1024.

Le même phénomène se reproduit lorsqu'on remplace l'algorithme Log-MAP par celui de SOVA (figure IV.3).l'effet des itérations reste le même.

Cependant, arriver à un certain nombre d'itérations, le BER produit ne porte pas une amélioration significative par rapport à l'itération précédente (voir la Figure IV.2.b « 4^{ième} et 5^{ième} itérations »).

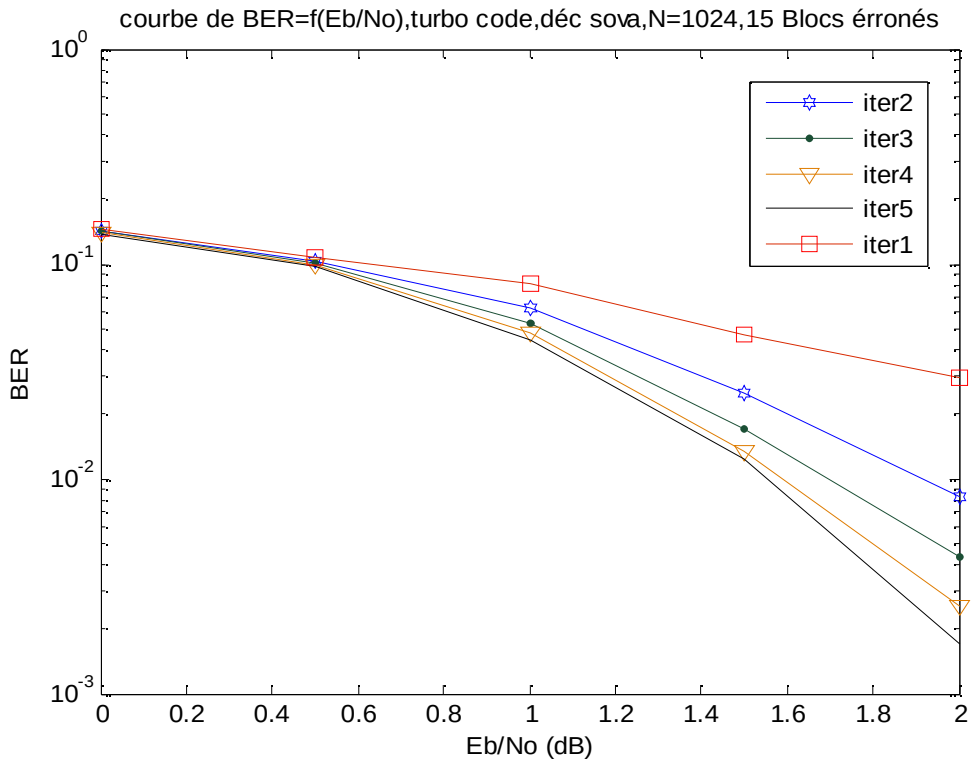


Figure IV.3 : détermination du BER en fonction de Eb/N0=2dB pour un certain nombre d'itérations en utilisant l'algorithme de décodage SOVA, N=1024.

IV.2.2. Effet de perforation sur les performances des codes turbo

Comme il est illustré dans la section II.2 du chapitre II, on a utilisé dans notre conception deux RSCs concaténés en parallèle pour générer des bits de parités lesquels vont être ajoutés aux bits systématiques d'information.

Si on transmet le code produit sans perforation on aura un taux de code R=1/3. Pour voir l'effet de la perforation sur les performances en BER des turbos codes, on a perforé les séquences de parités de chaque RSC pour avoir à la fin un taux de code R=1/2, cette conception est utilisé par Berrou [18].

Il est clair d'après la figure IV.3 que les performances de BER avec R=1/3 (sans perforation) sont meilleures que les performances avec R=1/2 (avec perforation) pour un bloc de longueur égale à 1024 bits. Si on fixe le BER à 10^{-4} , on aura un gain de code égale à 0.5 dB, ce qui donne une différence importante en puissance de transmission.

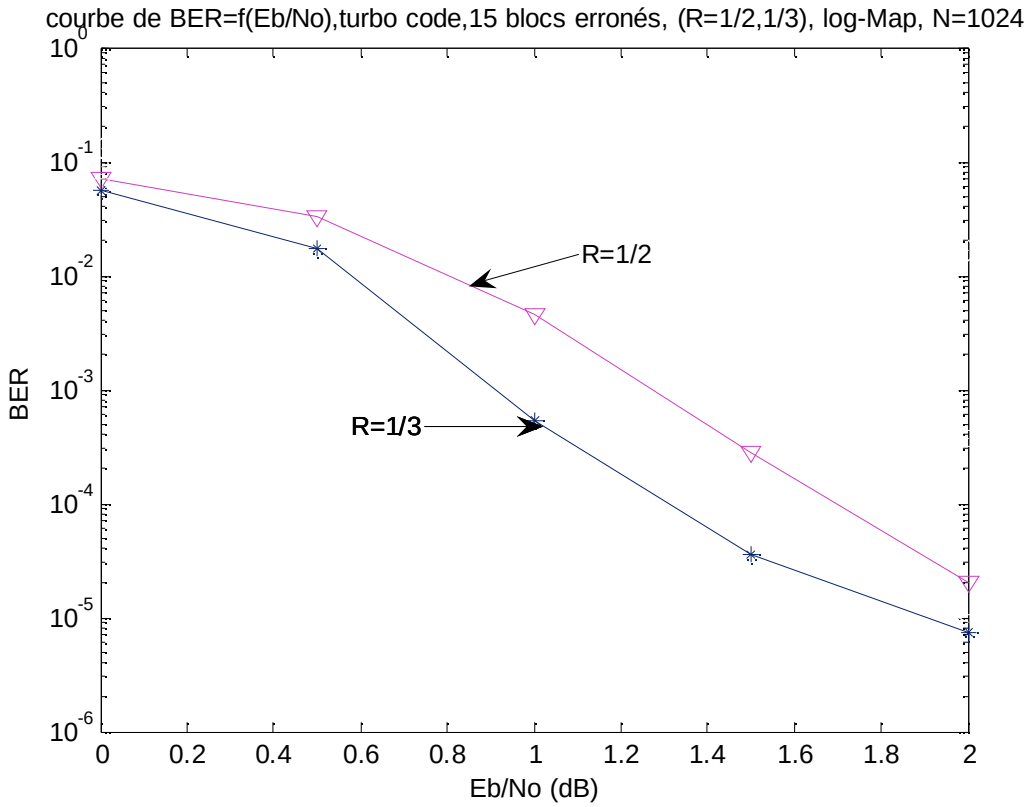


Figure IV.4 : Comparaison du BER pour les cas R=1/2 et R=1/3

IV.2.3. Effet de la longueur N du bloc sur les performances des codes turbo

Si on prend la cinquième itération pour les figures **(**IV.5. (a, b, c)) , on peut illustrer parfaitement le rôle dominant de la longueur N du bloc d'information dans le calcul du BER dans le cas des turbos codes. On déduit que le BER croit avec la longueur (taille) du bloc d'information à partir des valeurs moyennes de E_b / N_0 Si on transmit à un E_b / N_0 de 2dB par exemple, on a alors pour *N=169* bits un BER de l'ordre de10^{-2}, pour *N=1024* bits un BER de l'ordre de10^{-3}, et pour *N=3600* bits on a un BER de l'ordre de 10^{-4}. Donc on peut dire que l'augmentation de longueur de bloc d'information donne une amélioration considérable du taux d'erreurs binaire.

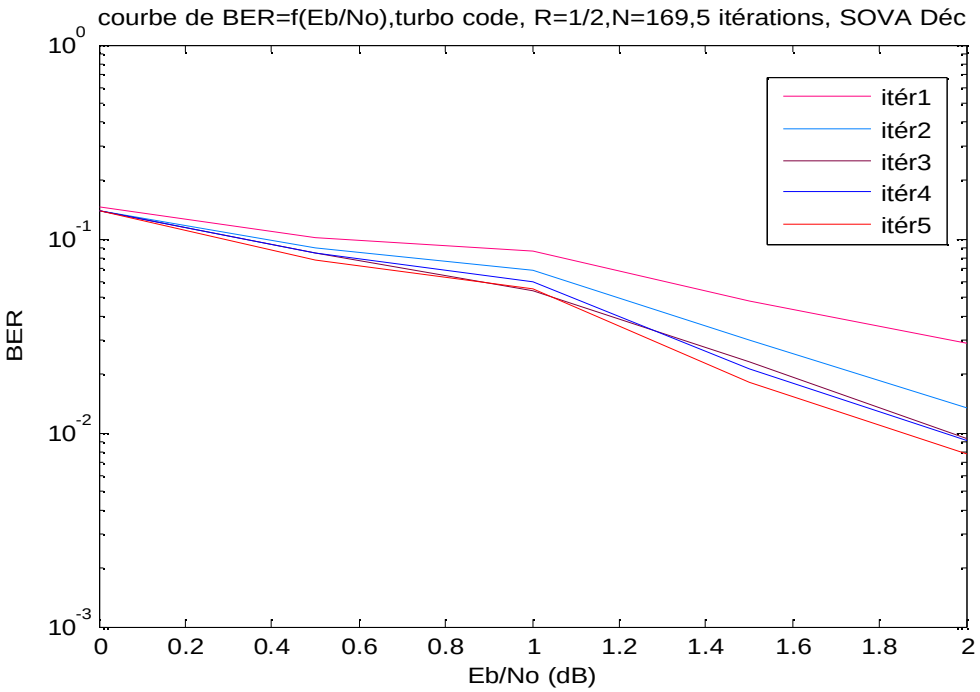


Figure IV.5.a Calcul du BER pour différentes valeurs de N=169 bits.

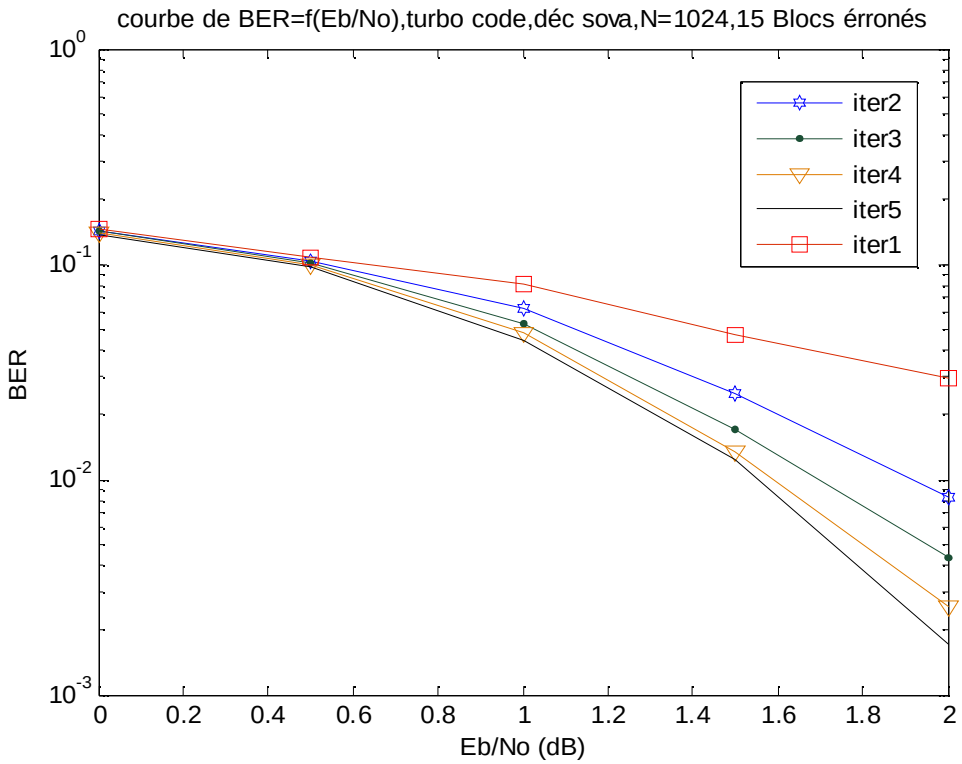


Figure IV.5.b Calcul du BER pour différentes valeurs de N=1024 bits

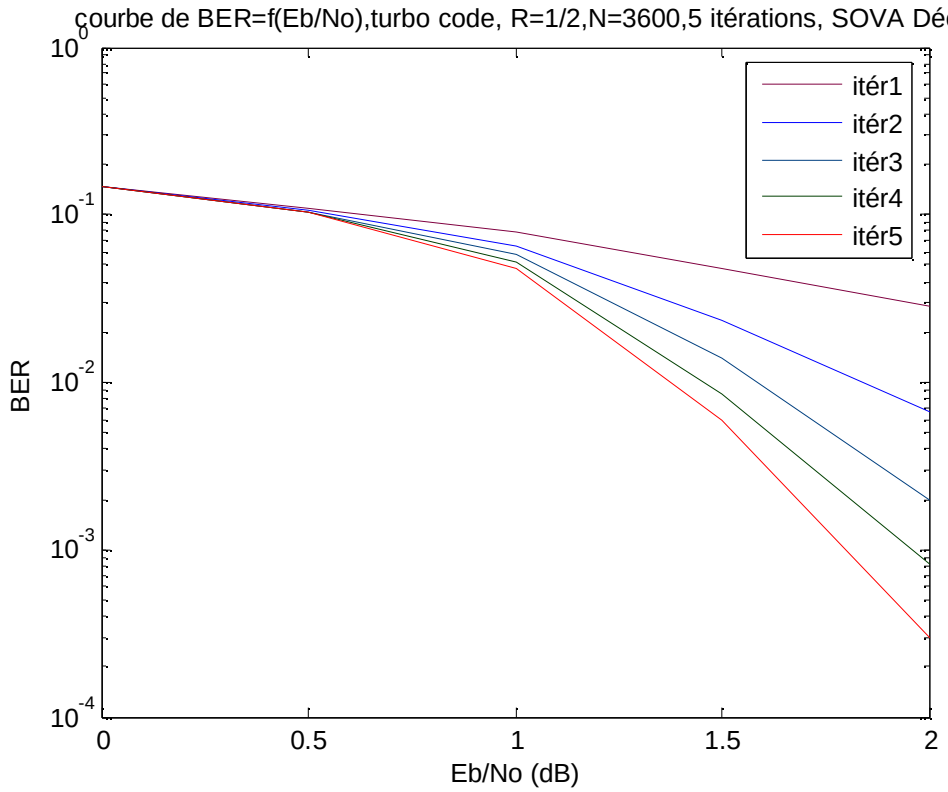


Figure IV.5.c Calcul du BER pour différentes valeurs de N=3600 bits

Conclusion générale

Ce travail concerne une initiation d'application d'une méthode de codage du canal appelé Turbo-Codes. L'utilisation des Turbo-Codes permet d'améliorer les performances d'un système de communication par rapport aux autres codes classiques. En fait, les performances en terme de BER = f(Eb/N0) sont meilleures à chaque augmentation du nombre d'itérations. Les turbo-codes, comparés aux codes conventionnels, présentent des résultats qui se rapprochent mieux des limites théoriques optimales de Shannon. En fonction des contraintes du système, ce type de codage pourrait être utilisé pour augmenter le débit de transmission et réduire la puissance d'émission.

Pour les entrelaceurs, partie intégrante des turbo-codes, nous avons présenté dans nôtres simulations avec MATLB , un entrelaceur aléatoire qui fait la permutation des bits d'entrées d'une façon aléatoires.

L'algorithme Log-MAP et l'algorithme SOVA sont utilisés dans le procédé du turbo décodage itératif. L'échange de l'information soft entre les décodeurs SISOs permet d'améliorer les performances des turbo-codes en combinant entre les paramètres du codeur/décodeur, à savoir : le nombre d'itérations, le type de RSC, le type d'entrelaceur, la longueur de contrainte *K*, l'algorithme de décodage, le taux de codage.

Enfin, on peut noter qu'il y a deux axes de recherche d'actualité sur les turbos codes. Ces axes concernent :

- les entrelaceurs dans la partie codage et
- les algorithmes de décodage dans la partie décodage.

Liste d'abréviation

BER Bit Error Rate (taux d'erreur par bit)
SOVA. Soft Output Viterbi Algorithm (algorithme de Viterbi à sortie soft)
AWGN Additif White Gaussien Noise (bruit additif blanc Gaussien)
RSC Recursive Systematic Coder (codeur convolutif systématique récursif)
M Nombre de mémoire ou registre

R le taux de codage

MAP Maximum A Posteriori
SISO Soft-In Soft-Out (entrée soft - sortie soft)
ARQ Automatic Repeat Request

FEC For Ward Error Correction

C Capacité d'un canal

CC classic code (code classique)

PCCC Parallel Concatenated Convolutive Code (Concaténation parallèle des codes convolutionels)
N la taille de l'entrelaceur

GRPI Golden Relative Prime Interleaver (entrelaceur d'or avec un nombre relativement primaire)
GI Golden Interleaver (entrelaceur d'or)
DGI Dithered Golden Interleaver (entrelaceur d'or perturbé)
LLR Log Likelihood Ratio (rapport de Logarithme de vraisemblance)
SNR Signal to Noise Ratio (rapport signal sur bruit)
ML Maximum Likelihood (maximum de vraisemblance)
ARQ Automatic Repeat Request

FEC For Ward Error Correction

Références

[1] C. E. Shannon, "*A mathematical theory of communications*" Bell Syst. Tech. J., July 1948.

[2] Louis Wehenkel, polycopie de cours "*théorie de l'information et du codage*", Université de Liége, faculté des sciences appliquées, Octobre 2001.

[3] Cédric Duchene, "*Réalisation de turbo codes. Analyse de décodage itératif par EXIT CHARTS*", projet de DEA, laboratoire ETIS de l'ENSEA.

[4] D. Divsalar and F. Pollara, "*Turbo codes for deep-space communications*", JPL TDA Progress Report 42-120, Feb. 1995.

[5] D. Arnold and G. Meyerhans, "*The Realization of the Turbo-Coding System*", Swiss Federal Institute of Technology, Zurich, Switzerland, July 1995.

[6] P. Robertson, "*Illuminating the Structure of Parallel Concatenated Recursive Systematic (TURBO) Codes*" Proceedings, GLOBECOM '94, San Francisco, Nov. 1994.

[7] A. S. Barbulescu and S. S. Pietrobon, "*Interleaver design for turbo codes*" Electron. Lett, Dec. 8, 1994.

[8] A. Barbnlescu and S. Pietrobon, "*Terminating the Trellis of Turbo-Codes in the Same State*", Electron. Lett, 1995.

[9] O. Joerssen and H. Meyr, "*Terminating the trellis of turbo-codes*," Electron. Lett., Aug. 4, 1994.

[10] Christian B. Schlegel and Lance C. Pérez, "*trellis and turbo coding*", Wiley-IEEE Press, 2005.

[11] O. Y. Takeshita and D. J. Costello, Jr. "*New deterministic interleaver designs for turbo codes*" IEEE Trans. Inform. Theory, 1988–2006, Sept. 2000.

[12] S. Dolinar and D. Divsalar, "*Weight distributions for turbo codes using random and nonrandom permutations*" JPL TDA Progress Report 42-122, Aug. 1995.

[13] S. Crosier, J. Lodge, P. Guinand, and A. Hunt, "*Performance of turbo-codes with relative prime and golden interleaving strategies*." Communication Research Centre, Station H, Ottawa, Canada, 1999.

[14] George White, "*Optimised Turbo Codes for Wireless Channels*" thesis of Ph. D., Communications Research Group, Department of Electronics, University of York, UK, October 2001.

[15] Todd K. Moon, "*Error correction coding: Mathematical methods and algorithms*" John Wiley & Sons, Inc., Publication, 2005.

[16] J. Hagenauer and P. Hoeher, "*A viterbi algorithm with soft-decision outputs and its applications*" in IEEE Globecom, , 1989.

[17] L.R. Bahl and J. Cocke and F. Jelinek and J. Raviv, "*Optimal Decoding of Linear Codes for Minimising Symbol Error Rate*", IEEE transactions on information theory, march 1974.

[18] C. Berrou and A. Glavieux and P. Thitimajshima, "*Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes*" in Proceedings of the International Conference on Communications, (Geneva, Switzerland), May 1993.

[19] L. Hanzo, T.H. Liew, B.L. Yeap, chapter 4 of "Turbo Coding, Turbo Equalisation and Space-Time Coding", Wiley-IEEE Press, July 2002.

[20] G. Forney, "The Viterbi algorithm" Proceedings of the IEEE, March 1973.

[21] M. Breiling, "Turbo coding simulation results" tech. Rep. University of Karlsruhe, Germany and Southampton University, UK, 1997.

[22] W. Koch and A. Baier, "Optimum and sub-optimum detection of coded data disturbed by time-varying intersymbol interference" IEEE Globecom, December 1990.

[23] J. Erfanian, S. Pasupathy, and G. Gulak, "Reduced complexity symbol dectectors with parallel structures for ISI channels" IEEE Transactions on Communications, 1994.

[24] P. Robertson and E. Villebrun and P. Höher, "A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain" in Proceedings of the International Conference on Communications, (Seattle, United States), June 1995.

[25] C. Berrou, P. Adde, E. Angui, and S. Faudeil, "A low complexity soft-output Viterbi decoder architecture" in Proceedings of the International Conference on Communications, May 1993.

[26] Yufei Wu, " turbo code demo page, site web: www.ee.vt.edu/~yufei/turbo/".

[27] Gérard Batail, " théorie de l'information: application aux techniques de communication", Masson, 1997.

الملخص

د لعمل يوصف تقنية جديد لاكتشا لترميز قنا لاتصا مستعملا لتسلسل بالتو لرمزتين لتفافيتين منهجيتين ترجعيتين مفصولتين بمحو مبد للتفر . د لأخير تسمى بالرمو لعنيفة كما يقا باللاتينية (تربو كو). حيث نها قا على لوصو لى حتما صغير للوقو في لخطأ على لبيت 10^{-5} ك بحاصل SNR يعا 0.7 سبا فقط , نها تستعمل في كل نظمة لاتصالا في لفضا , لاتصالا لمتنقلة بوسطة لقمر لصناعي, ... لخ

ستعملنا في د لعمل قنا غو (AWGN) د من جانب لترميز, ما بالنسبة لحل لتفر تعتبر حسن طريقة في د لمجا حلا لرمو لمكر (MAP) بلستعما مجا للوغا يتم لمشاهد. كل مز تستعمل قيم مسبقة تحر لقيم لمرمز , لقيم لظا رية, لقيم لمسبقة. لقيمة لظا رية تستعمل كقيمة مسبقة لإعا لمقبلة, لنتائج توضح لتبا لا لتكر تساعد لرمز على تحسين قد تهن على تصحيح لأخطا .

كلما مفتاحية : تشفير لقنا , لتشفير لعنيف,فك لتشفير لعنيف, خو ميا فك لتشفير.

Résumé

Ce travail présente une application d'une nouvelle technique de codage du canal de communication. Cette méthode est la concaténation parallèle de deux codes convolutifs systématiques récursives séparés par un entrelaceur des bits d'entrées appelé Turbo-Codes. Ces codes 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, par exemple les communications mobiles par satellite.

Les Turbo-Codes 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 et les itérations aident les décodeurs pour améliorer leur capacité de correction.

Mots clés : Codage de canal ; Turbo codes ; Turbo décodage ; Algorithme de décodage

Abstract

This work describes the application of a new coding technique using to coding the communication channel. This method is the parallel concatenation of two Recursive Systematic Convolutional Codes separated by the interleaver called Turbo-Codes. These codes capable to achieve an arbitrarily small Bit Error Rate (BER) of 10^{-5} with an SNR of just 0.7 dB. The Turbo-Codes can be used in any communication systems for example space communication mobile satellite.

This application using the 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 values. The extrinsic value is used as an a priori value for the next iteration. Results of simulation show how interleavers and the number of iterations help the decoders to improve their capability of correction.

Key words: Channel coding; Turbo coding; Turbo decoding; decoding algorithms.