République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université Echahid Hamma Lakhdar d'El-Oued

Faculté des Sciences et technologies
Département d'électronique
Spécialité : Télécommunications

Mémoire de Fin d'Etude

En vue de l'obtention du diplôme de Master

Présenté par : Khaldi kamel et El Bar Razika

Thème

Etude d'un canal AWGGN avec un décodeur itératif SOVA

Soutenu le : Juin 2018
Devant le jury composé de :

Nom & Prénom		
Mr. M. Hettiri	M.C.B	Président
Mr. S. Ghendir	M.C.B	Examinateur
Mr. R. Ajgou	M.C.A	Examinateur
Mr. A. Chemsa	M.C.A	Promoteur

2017/2018

Dédicaces

Nous dédie ce modeste travail :

A nos chers pères,

A nos frères,

A nos sœurs,

A toutes nos familles,

A tous nos enseignants durant nos études,

A tous ceux qui nous ont aidés et soutenu,…

Khaldi Kamel et Razika Barr

REMERCIEMENTS

En premier lieu, nous remercions Dieu qui nous a procuré ce succès.

Nous tenons à remercier vivement mon promoteur Mr Chemsa Ali pour ses conseils précieux et pour toutes les commodités et aisances qu'il nous a apportées durant mon étude et réalisation de ce projet.

Nos remerciements les plus vifs s'adressent aussi aux messieurs le président et les membres de jury d'avoir accepté d'examiner et d'évaluer notre travail.

Nous exprimons également notre gratitude à tous les professeurs et enseignants qui ont collaboré à notre formation depuis notre premier cycle d'étude jusqu'à la fin de notre master.

Sans omettre bien sûr de remercier profondément tous ceux qui ont contribué de prés ou de loin à la réalisation du présent travail. Et enfin, que nos chers parents et familles respectifs, et bien avant tout, trouvent ici l'expression de nos remerciements les plus sincères et les plus profonds en reconnaissance de leurs sacrifices, aides, soutien et encouragement afin de nous assurer cette formation de master dans les meilleures conditions.

Kamel et Razika

ملخص

ركزنا في هذا العمل على تشفير القناة بهدف التقليل من أخطاء الاتصال الرقمي الناجم عن التشويش AWGGN الخاضع لقانون غوص المعمم GGD. وبشكل خاص التشفير النفاث الملتف الذي يمثل آخر النتائج المتحصل عليها في هذا المجال، والذي يسمح ببلوغ الحد النظري **لشانون** المعلن عنه في 1948. هدف هذا العمل هو دراسة هذا النوع من التشفير مع الموائمة الثنائية ذات الإزاحة الطورية BPSK، وذلك قصد تحسين نسبة الخطأ الثنائي (BER) بحضور التشويش AWGGN باستعمال خوارزمية فيتاربي SOVA.

المصطلحات الأساسية :

التشفير النفاث الملتف CTC ، التشويش AWGGN، الموائمة الثنائية ذات الإزاحة الطورية BPSK، خوارزمية SOVA، نسبة الخطأ الثنائي BER.

Résumé

Dans ce travail, nous nous intéressons au problème du codage de canal, dont le but est de corriger les erreurs dues au canal à bruit AWGGN lors d'une transmission numérique. En particulier, les turbocodes convolutifs constituent la dernière avancée dans ce domaine et atteignent la borne prédite par **Shannon** en 1948. L'objectif de ce travail est d'étudier et de tester ces codes avec la modulation BPSK, pour améliorer le taux d'erreurs binaire (BER) en présence du bruit blanc additif à GGD abrégé AWGGN, et en utilisant l'algorithme SOVA.

Mots clefs

Turbocode convolutif CTC, Bruit AWGGN, Modulation BPSK, L'algorithme SOVA, Taux d'erreurs binaire BER.

Abstract

In this work, we are interested in the problem of the coding channel with AWGGN, whose goal is to correct the errors due to the channel at the time of a numeric transmission. In particular, the convolutional turbocodes constitutes the last progress in this domain and it is very close to the theoretical limit predicted by **Shannon** in 1948. The objective of this work is to study and to test this coding with the modulation BPSK, to improve the bit error rate (BER) in presence a additive white noise with GGD (AWGGN) using the SOVA algorithm.

Key Words

Convolutional Turbo Codes CTC, AWGGN, BPSK modulation, SOVA algorithm, Bit Error Rate BER.

Sommaire

Liste des figures

Glossaire

ADSL	**A**symmetric **D**igital **S**ubscriber **L**ine
AM	**A**mplitude **M**odulation
APP	**A** **P**osteriori **P**robability
ASK	**A**mplitude **S**hift **K**eying
AWGGN	**A**dditif **W**hite **G**eneralized **G**aussien **N**oise
AWGN	**A**dditif **W**hite **G**aussien **N**oise
BER	**B**it **E**rror **R**ate
BPSK	**B**inary **P**hase **S**hift **K**eying
CTC	**C**onvolutional **T**urbo **C**ode
DCS	**D**igital **C**ommunication **S**ystem
FM	**F**requency **M**odulation
FSK	**F**requency **S**hift **K**eying
GGD	**G**eneralized **G**aussian **D**istribution
I.I.d	**I**ndependent and **I**dentically **D**istributed
LLR	**L**ogarithm **L**ikelihood **R**atio
MAP	**M**aximum **A** **P**osteriori
ML	**M**aximum **L**ikelihood
NRNSC	**N**on **R**ecursive **N**on **S**ystematic **C**ode
NSC	**N**on **S**ystematic **C**ode

PDF	**P**robability **D**ensity **F**unction
PLC	**P**ower **L**ine **C**ommunication
PM	**P**hase **M**odulation
PSK	**P**hase **S**hift **K**eying
RSC	**R**ecursive **S**ystematic **C**ode
SC	**S**ystematic **C**ode
SNR	**S**ignal **N**oise **R**ate
SOVA	**S**oft **O**utput **V**iterbi **A**lgorithm
UWB	**U**ltra **W**ide **B**and

Introduction générale

Par définition, une transmission numérique permet d'acheminer des messages de nature numérique entre une source et un destinataire. Cette source délivre, avec un certain rythme, une suite de symboles qui prennent leurs valeurs dans un ensemble fini appelé alphabet. Le caractère numérique de la transmission se manifeste de manière évidente dans le fonctionnement du récepteur lors de l'échantillonnage et de la prise de décision. Il est beaucoup moins apparent au niveau du signal transmis qui présente plus nettement le caractère discret de la source.

Par une convention préalable, le destinataire a la connaissance de l'alphabet utilisé par la source. Il peut donc interpréter l'information qu'il reçoit en fonction de cet alphabet. Il compare les signaux reçus (déformés et perturbés par la transmission dans le canal) à la liste de caractères possibles et il déduit par une «décision» lequel de ces caractères est le plus probablement à l'origine du signal reçu.

La transmission de l'information entre la source et le destinataire ne s'effectue pas sans risques d'erreurs à cause du bruit de distorsion du canal. Afin de remédier à cela et donner une grande fiabilité au système de transmission, il existe plusieurs techniques de protections de l'information se basant toutes sur le principe d'ajouter une certaine redondance au message original avant la transmission à l'aide d'un codeur qui maximise la distance de **Hamming** entre les mots-code. Cette opération qui consiste à l'ajout des bits de redondance est dite "codage de canal". Parmi les célèbres codes qui ont le pouvoir de combattre les erreurs de transmission, sont les codes convolutifs car leurs prépotences de correction s'accroissent lorsque la longueur du registre de codage augmente. Pour satisfaire les applications les plus courants du codage de canal, une mémoire ν de l'ordre de 30 ou 40 serait nécessaire (donc 2^{30} ou 2^{40} états, c'est à dire plus d'un milliard d'états)

puisque à partir d'une certaine longueur de registre et pour un rendement de codage 1/2 la distance minimale de **Hamming** d'un code convolutif de mémoire ν est de l'ordre de grandeur de ν . Si l'on savait décoder aisément un code convolutif à plus d'un milliard d'états, on ne parlerait plus beaucoup de codage de canal [1] ! Une telle proposition ne convienne plus. Pour cela, **C.Berrou** et **A. Glavieux** [2] en 1993, ont inventé des très bons codes qui font parfaitement l'affaire, constitués par une concaténation de deux codes convolutifs via un entrelaceur entre les deux codes. Ces codes concaténés sont non seulement des codes puissants, mais leur structure permet de plus une faible complexité de décodage grâce au décodage itératif. Le décodage itératif consiste à décoder alternativement les codes constitutifs et à passer de l'information entre les décodeurs constitutifs. Afin d'exploiter au mieux l'information produite par chaque décodeur constitutif, il a été établi que passer des décisions souples plutôt que des décisions fermes peut conduire à d'excellentes performances. Ce décodage itératif basé sur les décisions souples est appelé aussi le décodage turbo. Pour cette raison ces codes sont appelés "turbocodes convolutifs".

Ces turbocodes sont étudiés et évalués dans l'hypothèse de gaussianité du bruit, puisque l'objectif était d'obtenir des relations simples sur les performances et des mises en oeuvre peu coûteuses. Même si cette hypothèse est justifiée dans un large éventail de scénarios de situations réelles, il y a des situations où le théorème central limite ne peut plus être invoqué et où les expériences mettent en exergue un bruit non gaussien à caractère impulsif où la queue de la distribution de son amplitude est lourde.

Parmi ces situations, nous pouvons citer les canaux de courant porteur en ligne PLC, certaines lignes ADSL ou encore les canaux Ultra-Large-Bande (UWB) impulsionnelle.

En conséquence, il est mieux pratiquement d'étudier l'efficacité des turbocodes convolutifs, avec un algorithme de décodage optimal ou moins sous optimal bien choisi, dans un canal à bruit peut être impulsif. Pour ces raisons notre travail est consacré sur ce type de codes avec l'algorithme de **Viterbi** modifié SOVA dans un canal à bruit AWGGN (ce type de bruit contient la classe de bruits impulsifs). Ce type de bruit est présenté dans ce travail par une distribution gaussienne généralisée symétrique GGD [3].

Dans ce travail, nous nous intéressons au problème du codage de canal, dont le but est de corriger les erreurs dues au canal à bruit AWGGN lors d'une transmission numérique. En particulier, les turbocodes convolutifs constituent la dernière avancée dans ce domaine. L'objectif de ce travail est d'étudier et de tester ces codes avec la modulation BPSK, pour améliorer le taux d'erreurs binaire (BER).

Le premier chapitre consiste en des généralités sur les systèmes de communication numérique y compris la modélisation d'un canal. Dans ce chapitre les notions de la distribution gaussienne généralisée et du bruit impulsif sont présentées.

Le second chapitre est réservé aux turbocodes convolutifs CTC et à l'algorithme de SOVA. Une étude bien détaillée sur l'algorithme SOVA et son application, dans le cas d'un bruit AWGGN, est exposée dans ce chapitre. Le principe du décodage itératif avec SOVA et la notion de l'information extrinsèque dans le cas d'un bruit impulsif sont aussi présenté dans cette partie.

Le troisième chapitre est dédié à la simulation et l'évaluation de la performance en termes de BER d'un CTC en fonction du rapport signal à bruit SNR, pour un canal à bruit blanc additif et à GGD symétrique (AWGGN) et avec une modulation BPSK. L'implémentation de l'algorithme SOVA est présentée dans ce chapitre. Les résultats de notre simulation sont tirés à l'aide du MATLAB.

Chapitre 1

Généralités sur les systèmes de communication numériques

1.1. Introduction

La tâche de la conception d'un système de communication numérique est de réaliser un système efficace raisonnable pour la transmission d'information à partir d'une source à un débit et un niveau de fiabilité qui sont acceptables pour l'utilisateur.

Les deux paramètres clés lors de la conception d'un tel système sont

- la puissance du signal transmis ;
- la bande passante du canal.

Ces deux paramètres ensemble avec la densité spectrale de puissance du bruit du récepteur, déterminent le rapport de l'énergie du signal par bit sur la densité spectrale de puissance du bruit E_b/N_0.

Pour un rapport E_b/N_0 fixé, la seule façon pratique d'améliorer la qualité de transmission est d'utiliser le codage de canal.

Une autre motivation pratique pour l'utilisation de ce type de codage est de réduire le rapport E_b/N_0 exigé, ce qui permet aussi de réduire la taille de l'antenne et par conséquent le coût de la transmission.

Le codage de canal consiste à ajouter au message à transmettre des symboles de contrôle (non informatifs) suivant une loi donnée et à venir vérifier au décodage que cette loi est respectée. Si c'est le cas, on considère qu'il n'y a pas eu d'erreurs lors de la transmission. Dans le cas contraire, on détecte la présence d'erreurs que l'on peut éventuellement corriger.

1.2. Modèle d'un système de communication numérique

Le modèle est présenté par la figure 1.1.

La transmission d'information numérique actuelle à travers un canal réel est accomplie par un modulateur et un démodulateur, qui peut opérer à un débit d'information numérique R.

La probabilité d'erreur P_e à la réception, en utilisant seulement le modem, dépend du débit R et du rapport signal sur bruit. Cette probabilité dépasse une valeur inférieure prescrite chaque fois qu'on augmente le débit R, d'où le besoin d'un recours à la conception des codeurs et des décodeurs de canaux afin d'obtenir des probabilités d'erreurs acceptables. [4]

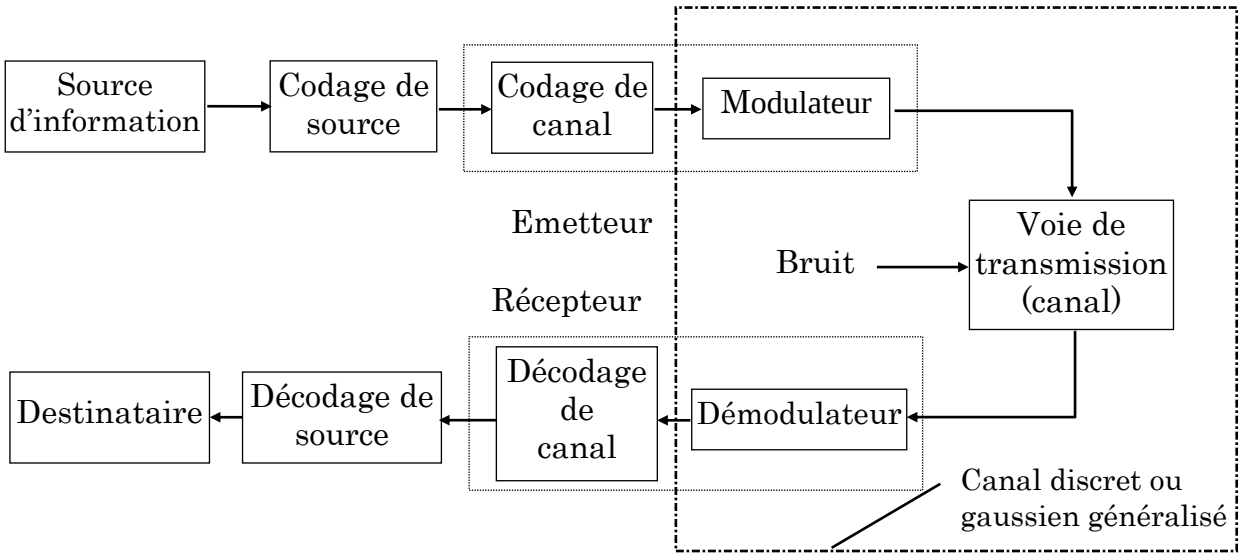


Fig. 1.1 Modèle d'un système de communication numérique.[4]

Nous décrivons ici brièvement les différentes fonctions d'un système de communication numérique

- La source d'information délivre le message contenant l'information à transmettre ;
- Le codeur de source supprime toute redondance dans le message ;
- Le codeur de canal introduit une redondance contrôlée dans le message issu du codeur de source. Cette redondance connue aussi au niveau du récepteur, permettra la détection et/ou la correction des erreurs dues au bruit inévitable ;

- Le modulateur a pour fonction principale de modifier le signal message en une forme plus adaptée à la transmission à travers le canal ;
- Le canal représente le milieu physique entre l'émetteur et le récepteur. Il peut être l'espace libre, un câble coaxial ou une fibre optique ;
- Le démodulateur et les décodeurs, de canal, de source réalisent, respectivement, les fonctions inverses du modulateur, du codeur de canal et du codeur de source afin de restituer le signal original.[4]

1.3. Modélisation d'un canal

Deux modèles de canal sont souvent adoptés selon que la sortie du canal sera prise après ou avant le circuit de décision du démodulateur et selon le type de bruit affectant le canal.

Dans le premier cas, l'entrée et la sortie du canal seront discrètes, alors on parlera de *canal discret (canal discret = modulateur + canal réel + démodulateur)*. Dans le second cas, lorsque le bruit est un bruit blanc gaussien généralisé, on parlera du *canal gaussien généralisé (deux cas particuliers: gaussien ou impulsif)*. [5]

1.3.1. Canal discret

Un canal discret peut être défini par ses probabilités de transition p_{ij}^k où

$$p_{ij}^k = \Pr\{R_k = r_j / Y_k = y_i\}$$

avec

$$\sum_{j=0}^{m-1} p_{ij}^k = 1, \forall i \quad (1.1)$$

où Y_k et R_k représentent respectivement l'entrée et la sortie du canal à l'instant k et prennent leurs valeurs dans les ensembles : $[y_0, \dots, y, \dots, y_{n-1}]$ et $[r_0, \dots, r_j, \dots, r_{m-1}]$, p_{ij}^kest la probabilité d'avoir r_j en sortie lorsque y_i est introduit à l'entrée (est transmis).

Lorsque les probabilités de transition sont indépendantes du temps, le canal est *stationnaire*. Si un symbole à la sortie du canal à l'instant $t = kT$ (où T est la période d'échantillonnage) ne dépend que du symbole à l'entrée du canal à l'instant $t = kT$, le canal est dit *sans mémoire*.[5]

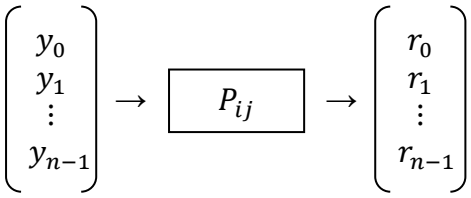


Fig. 1.2 Canal discret sans mémoire.

Lorsque les symboles à l'entrée et à la sortie du canal sont binaires et que les probabilités de transition sont symétriques, c'est-à-dire[5] :

$$p_{ij} = p_{ji}, \forall i,j \in \{0,1\} \tag{1.2}$$

Alors le canal est appelé *canal binaire symétrique,* et il est entièrement défini par sa probabilité d'erreur p d'un bit [5]:

$$p_{01} = p_{10} = p$$
$$p_{00} = p_{11} = 1 - p \tag{1.3}$$

1.3.2. Canal à bruit blanc gaussien généralisé additif (AWGGN)

La fonction de densité de probabilité (PDF) d'une variable gaussienne généralisée aléatoire X, avec moyenne μ et variance σ^2, est définie comme :

$$f(x) = \frac{\alpha}{2 \cdot A \cdot \Gamma(1/\alpha)} exp\{-|(x-\mu)/A|^{\alpha}\} \quad x \in \mathbb{R}, \tag{1.4}$$

où

$$A = \sigma \left[\frac{\Gamma(1/\alpha)}{\Gamma(3/\alpha)}\right]^{1/2} \quad et \quad \Gamma(z) = \int_0^{+\infty} t^{z-1} e^{-t} dt, \quad z > 0 \tag{1.5}$$

(Γ est la fonction gamma).

La distribution gaussienne généralisée (GGD) est symétrique par rapport à μ. A est un facteur d'échelle qui définit la dispersion de la distribution, d'où il est une mesure généralisée de la variance. $\alpha > 0$ est le paramètre de forme qui décrit le taux exponentiel de la décroissance : queues lourdes correspondent à des valeurs plus petites de μ.

Lorsque $\alpha = 2$ (resp. $\alpha = 1$, $\alpha \to +\infty$) $f(x)$ dans (1.4) est la PDF d'une distribution gaussienne (resp. Laplacienne, uniforme).

La distribution présente des caractéristiques d'impulsion plus que $\alpha \to 0$. Puisque nous sommes intéressés au bruit AWGGN, **nous considérons dans tout ce travail le cas où la GGD est symétrique.** Le chapitre 3 présente bien la notion de cette distribution.

Un canal à bruit gaussien généralisé additif (AWGGN) est un canal à entrée binaire et sortie analogique. La sortie se représente par une variable aléatoire continue R_k.

$$R_k = Y_k + W_k \tag{1.6}$$

où Y_k est le symbole binaire émis et W_k est une variable aléatoire gaussienne généralisée centrée $(\mu = 0)$ de variance σ^2 correspondant au bruit impulsif du canal. Les densités de probabilité de transition du canal s'écrivent

$$P(R_k/Y_k\ émis) = \frac{\alpha}{2 \cdot A \cdot \Gamma(1/\alpha)} exp\{-|(R_k - Y_k)/A|^{\alpha}\} \tag{1.7}$$

Avec $A = \sigma \left[\frac{\Gamma(1/\alpha)}{\Gamma(3/\alpha)}\right]^{1/2}$ $\quad et \quad$ $\Gamma(z) = \int_0^{+\infty} t^{z-1} e^{-t} dt\,, \quad z > 0$

La variance σ^2 est fonction du rapport signal à bruit $SNR = E_b/N_0$

$$\sigma^2 = \frac{E_b}{2 \cdot SNR} \tag{1.8}$$

où E_b est l'énergie du signal par bit, et N_0 est la densité spectrale de puissance mono latérale du bruit additif.

Si le canal est stationnaire et sans mémoire, les densités de probabilité sont indépendantes du temps et la sortie du canal à l'instant $t = kT$ (où T est la période d'échantillonnage, généralement prise égale à l'unité du temps) ne dépend que du symbole d'entrée à l'instant $t = kT$.

1.4. Théorèmes fondamentaux sur le codage du canal

1.4.1. Théorème de codage de Shannon

Il est possible, en utilisant un codage de canal approprié, de transmettre de l'information sous forme d'une suite de symboles discrets avec une probabilité d'erreur aussi faible que l'on veut si le débit d'information R est inférieur à la capacité du canal C.

Le théorème de **Shannon** affirme l'existence de codes dont la probabilité d'erreur est arbitrairement faible mais ne montre pas comment ces codes peuvent être construits.

Ce théorème affirme une chose toute à fait surprenante à savoir que, quel que soit le niveau des perturbations d'un canal, on peut toujours y passer des messages codés d'une manière appropriée avec une probabilité d'erreur aussi faible que l'on veut, c'est la raison pour laquelle ce théorème était la cause de l'énorme développement de la théorie des codes.

En pratique, dans tous les cas où $R < (1/2)C$, il existe des codes qui réalisent une probabilité d'erreur très faible, où R représente le débit d'information et C est la capacité du canal [6].

1.4.2 Théorème de Nyquist

Le théorème de **Nyquist** affirme qu'à travers un canal équivalent à un filtre passe-bas idéal avec une fréquence de coupure B, il est possible de transmettre des signaux binaires (impulsions) indépendants avec un débit de moments $R_s \leq 2B$ impulsions /seconde sans interférences entre symboles, où R_s est donné par définition [6]

$$R_s = \frac{nombre\ d'impulsions\ dans\ l'intervalle\ T}{T}$$

1.5 Types d'erreurs

Les erreurs dans les systèmes de communication numériques sont dues au bruit du canal de transmission. Généralement, deux sortes de bruits peuvent être distinguées dans les canaux de transmission. [4]

- Le bruit blanc gaussien : constitue le souci principal dans la conception des modulateurs, des codeurs, des décodeurs, et des démodulateurs pour la transmission de l'information numérique. Les erreurs de transmission introduites par le bruit blanc gaussien sont considérées comme des erreurs aléatoires.
- Le bruit impulsif est caractérisé par des intervalles de courte durée mais avec des amplitudes importantes du bruit. Ce type de bruit résulte de plusieurs causes naturelles ou artificielles telles que les interrupteurs de commutation, des centraux de communication, arcs électriques,..., etc. Quand un paquet d'erreurs apparaît, il affecte plusieurs symboles ou bits, et il y a d'habitude une dépendance des erreurs dans les symboles affectés. Ces erreurs apparaissent en paquets.

Les schémas de contrôle d'erreurs qui s'occupent d'erreurs aléatoires sont appelés codes correcteurs d'erreurs aléatoires, tandis que les schémas de codage conçus pour corriger les paquets d'erreurs sont appelés codes correcteurs de paquets d'erreurs.

1.6 Types de codes

Les codes de canal appelés aussi codes correcteurs d'erreurs sont répartis en trois catégories : Les codes en blocs linéaires, les codes convolutifs et les codes concaténés ou les turbocodes.[7]

Le codage en blocs consiste à associer, à chaque bloc de k bits d'information, un bloc de n bits $(n > k)$ contenant $n - k$ bits de redondance. Les 2^k blocs de n bits délivrés par un codeur sont appelés les mots-code. Le rapport k/n est appelé le rendement du code.

Les opérations de codage et de décodage dans les codes en blocs se font à l'aide d'addition et de multiplications sur des éléments binaires. Ces dernières correspondant, respectivement, aux opérations logiques «OU» et «ET» exclusif.

Il existe un type spécial de code en blocs linéaires appelé codes cycliques. Ce sont des codes en blocs linéaires vérifiant quelques propriétés supplémentaires [4], [8].

Les codes convolutifs forment une classe extrêmement souple et efficace de codes correcteurs d'erreur. Ce sont les codes les plus utilisés dans les communications fixes et mobiles. Les codes convolutifs ont les mêmes caractéristiques que les codes en bloc sauf qu'ils s'appliquent à des séquences infinies de symboles d'information et génèrent des séquences infinies de symboles de code [8].

Un turbo codeur classique résulte de l'association de deux (ou plus) codeurs. Il s'agit souvent de codeurs convolutifs récursifs systématiques RSC car leur récursivité apporte des propriétés pseudo-aléatoires intéressantes.

Le principe des turbocodes, comme tout code correcteur d'erreur, est d'introduire une redondance dans le message afin de le rendre moins sensible aux bruits et perturbations subies lors de la transmission. Le codage consiste à utiliser deux codeurs simples, dont les entrées ont été entrelacées; ainsi, chaque codeur voit une série d'informations différentes à son entrée.

Le décodage est une collaboration entre les décodeurs, chacun donnant son « avis » (notion de confiance) sur chaque bit décodé. Cette information est ensuite fournie à l'entrée du prochain décodeur, et ainsi de suite. D'où l'appellation « turbo » [1].

1.7 Techniques de modulation

La fonction de modulation a pour objectif d'adapter le spectre de signal à émettre au canal de transmission. Lorsqu'il s'agit d'une transmission numérique à travers un canal du type passe bande, il est nécessaire de transposer (ou de translater) le spectre de fréquence du signal à bande de base. Ceci peut se faire au moyen d'une porteuse qui est en général une fonction sinusoïdale pure. Cette opération s'appelle *modulation numérique* ou bien *modulation analogique discrète* car la porteuse est analogique et l'information (signal de base) est discrète. [8] [11]

En tout cas, on peut distinguer dans la modulation analogique discrète

- ASK : modulation par déplacement d'amplitude (Amplitude Shift Keying) ;
- PSK : modulation par déplacement de phase (Phase Shift Keying) ;
- FSK : modulation par déplacement de fréquence (Frequency Shift Keying)
- Ou une modulation composite ou hybride comme la modulation QAM qui est une combinaison des modulations ASK et PSK.

Ces modulations peuvent être considérées comme des cas spéciaux des modulations analogiques AM, PM, FM, et AM-PM respectivement.

Sur un canal à bruit impulsif, le choix d'une modulation se fait en considérant l'occupation spectrale, les performances et la complexité du couple modulateur/démodulateur. Il est à noter que la faible occupation spectrale et les performances du système sont deux contraintes antagonistes, ce qui nécessite en pratique un compromis lors du choix d'une modulation.

Nous allons maintenant présenter un rappel sur le type de modulation utilisée dans notre travail qui est la modulation BPSK.

La Phase-shift keying (ou PSK, soit « modulation par changement de phase ») désigne une famille de formes de modulations numériques qui ont toutes pour principe de véhiculer de l'information binaire via la phase d'un signal de référence (porteuse), et exclusivement par ce biais.

Comme pour toute technique de modulation numérique, la phase en question ne peut prendre qu'un nombre fini de valeurs. Chacune de ces valeurs représente un unique nombre binaire, dont la taille (et donc la quantité d'information transmise) dépend du nombre de valeurs possibles pour la phase. Généralement, pour une modulation PSK donnée, les nombres binaires représentés sont tous de même taille.

BPSK est la forme la plus simple du PSK. Elle utilise deux phases qui sont séparées de 180° ; on l'appelle également 2-PSK. Cette modulation est la plus robuste de toutes les PSK car il faut une grande déformation du signal pour que

le démodulateur se trompe sur le symbole reçu. Cependant on ne peut moduler qu'un seul bit par symbole (voir la figure 1.3), ce qui est un inconvénient pour les applications qui nécessitent un débit binaire élevé.

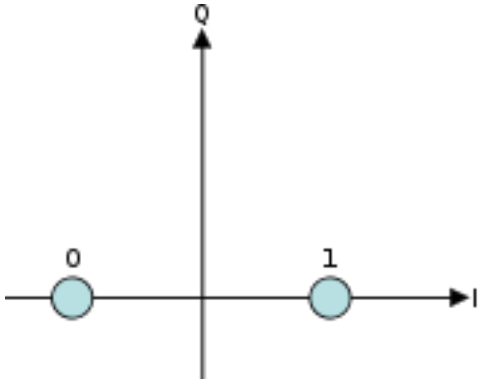


Fig. 1.3 Diagramme de constellation pour BPSK.

La probabilité d'erreur binaire du BPSK pour un bruit GGD est la probabilité de décider qu'un 1 avait été transmis alors que c'était un 0 (et inversement). Ecrivons ce qu'est P_e dans ce cas concret

$$P_e = P(\,0\ est\ trasmis\,)\,.P\ (1\ est\ choisit)\ + P(1\ est\ trasmis)\,.P\ (0\ est\ choisit)$$

$$P_e = \frac{1}{2}\,.P\ (1\ est\ choisit) + \frac{1}{2}\,.P\ (0\ est\ choisit) = \frac{1}{2}\,.P\ (R_k < 0) + \frac{1}{2}\,.P\ (R_k > 0)$$

$$P_e = \frac{1}{2}\,.P\left(\sqrt{E_b} + W_k < 0\right) + \frac{1}{2}\,.P\left(\sqrt{E_b} + W_k > 0\right) = \frac{1}{2} - P\left(0 \le W_k < \sqrt{E_b}\right)$$

et avec l'équation (1.4) on trouve:

$$P_e = \frac{1}{2} - \int_0^1 \frac{\alpha \cdot (\Gamma(3/\alpha))^{1/2} \cdot \sqrt{SNR}}{\sqrt{2} \cdot (\Gamma(1/\alpha))^{3/2}} exp\left\{-\left(\sqrt{2} \cdot \left(\frac{\Gamma(3/\alpha)}{\Gamma(1/\alpha)}\right)^{1/2} \cdot \sqrt{SNR} \cdot x\right)^{\alpha}\right\} dx \qquad (1.9)$$

où $SNR = \frac{E_b}{2\sigma^2}$. Comme il y a un bit par symbole, cela correspond également au taux d'erreur de symbole.

La figure suivante présente la variation de la probabilité d'erreur en fonction du SNR dans le cas d'un bruit impulsif à GGD.

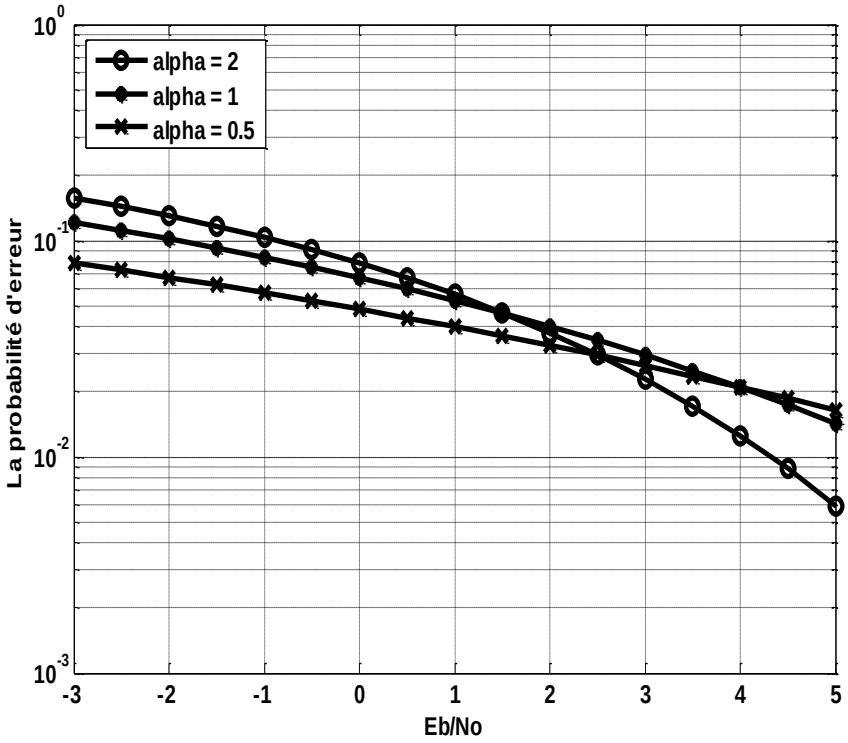


Fig. 1.4 La probabilité d'erreur en fonction du SNR.

1.8 Conclusion

On a présenté dans ce chapitre quelques généralités sur des systèmes de communication numériques et ses parties constituantes, en particulier les fonctions du codage et de la modulation.

On a exposé la notion d'un bruit AWGGN présenté par une GGD symétrique.

Toutefois dans cette présentation, on a essayé de donner l'essentielle de ce thème pour ne pas être exhaustif. Dans la section 1.7, nous avons présenté la formule de la probabilité d'erreur de la modulation BPSK pour des formes d'ondes non codées, dans le cas d'un bruit AWGGN. Ceci est très utile pour les comparaisons que nous devons faire entre les performances des systèmes utilisant des séquences codées et non codées.

Il est à noter que le choix de code dans un système de communication numérique est très primordial car le codage améliore colossalement la performance du système en corrigeant les erreurs de la transmission. Parmi les codes largement utilisés dans les systèmes de communication actuels, les turbocodes convolutifs, qui ont un grand pouvoir de lutter contre les erreurs. Ceci fait l'objet du chapitre suivant.

Chapitre 2

Les turbocodes convolutifs CTC et l'algorithme SOVA

2.1 Introduction

Le théorème de **Shannon** affirme qu'il existe au moins un code de canal qui nous permet de transmettre de l'information sous forme d'une série de symboles discrets avec une probabilité d'erreur faible que l'on veut à condition que le débit d'information soit inférieur à la capacité de canal.

Après un long tâtonnement **C. Berrou**, **A. Glavieux** et **P. Thitimajshima**, en 1993 [9] ont trouvé un très bon code correcteur d'erreurs fonctionnant à moins de 0.5 dB de la limite de **Shannon** pour un canal à bruit gaussien [10]. Cette rupture technologique dans le domaine du codage de canal a tout d'abord surpris la communauté scientifique, mais les résultats ont été très rapidement confirmés. La solution proposée, qui consiste en une concaténation parallèle ou série de deux codes convolutifs, généralement récursifs systématiques identiques [11], au travers d'un entrelaceur, est connue sous le nom « Turbocodes Convolutifs CTC ».

Le but de l'entrelaceur (Interleaver) dans un CTC est de faire la permutation des données émises afin de casser les paquets d'erreur surviennent lors de la transmission, et ce pour rendre leur distribution plus uniforme, ce qui donne au CTC une grande efficacité de corriger les erreurs [12].

Le décodeur associé à un CTC est un décodeur souple ou doux, car ce type de décodeurs est très performant qu'un décodeur dur ou ferme pour un canal à bruit impulsif, et très facile à mettre en œuvre pour tous les codes acceptant des représentations en treillis, par la raison qu'il utilise un algorithme de Viterbi à sortie douce SOVA.

Un tel décodage permet d'extraire de l'information sur chacun des décodeurs constitutifs et de l'échanger entre ces deux décodeurs. L'information extraire d'un décodeur est dite « extrinsèque » et est réinjectée à l'itération suivante dans un autre décodeur composant afin de bénéficier de la diversité de codage. Le processus pouvant se répéter plusieurs fois, on parle de décodage itératif. L'appellation« Turbo » provient justement de la réinjection des informations extrinsèques pour rappeler l'image des moteurs turbo qui réutilisent une partie de l'énergie gaspillée par le moteur pour donner plus de puissance au moteur.

2.2 Structure des CTC

Les concaténations les plus célèbres pour les CTC sont la concaténation parallèle et la concaténation série de deux codes convolutifs. Il est possible de fabriquer des turbocodes hybrides en combinant la série et le parallèle. Nous jugeons que les structures hybrides sont peu intéressantes car un très grand gain en performances est déjà atteint par les structures classiques.

Donc, l'idée primordiale du CTC est d'utiliser deux codes convolutifs systématiques récursifs RSC concaténés en parallèle via un entrelaceur entre les deux, ou d'utiliser un code convolutif quelconque (RSC, NRNSC …) dit « code externe » et un code convolutif RSC dit « code interne » en série, avec un entrelaceur entre les deux (Fig. 2.1).

Si les codes convolutifs peuvent coder un flux continu d'information, les turbocodes ne peuvent coder l'information que par paquets de longueur finie égale à celle de l'entrelaceur. Ainsi, après chaque bloc d'information, le codeur est forcé à un état connu, généralement l'état 00. Alors, les bits de terminaison sont placés à la suite des bits d'information avant d'effectuer le décodage [13][14].

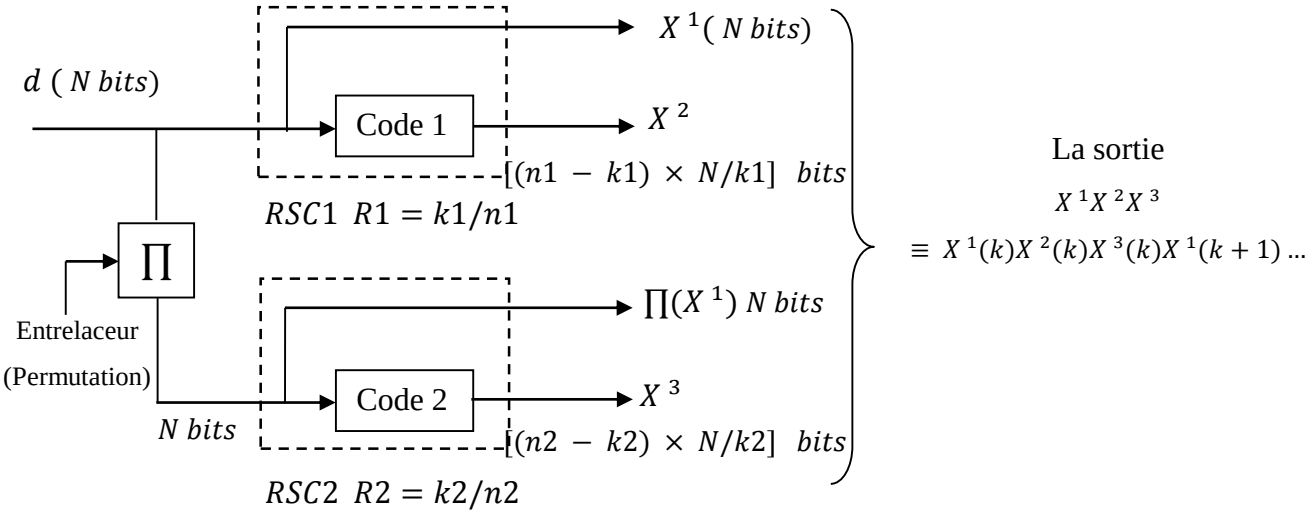


Fig. 2.1.a Turbocodeur parallèle[14].

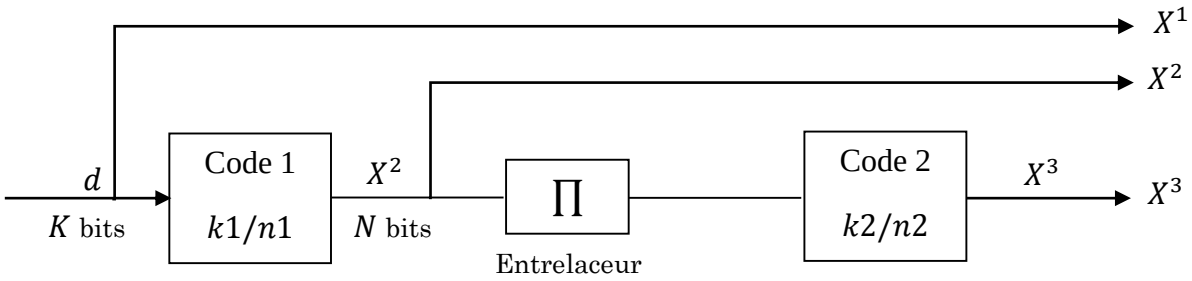


Fig. 2.1.b Turbocodeur série[14].

Remarque

Il faut noter que les turbocodes basés sur les NRNSC ne présentent pas de gain d'entrelaceur, puisque ils ont un grand coefficient d'erreur [14].

2.2.1 Les CTC parallèles

Un turbocode parallèle est un code linéaire binaire qui donne pour une séquence d'entrée à N bits une séquence de sortie à J bits (Fig. 2.1.a). Le treillis des deux codes RSC1 et RSC2 démarre de l'état zéro et se termine après $\frac{N}{Ki+Ti}$ branches dans l'état zéro. Chaque code convolutif RSCi $(i = 1,2)$ nécessite T_i branches dans la phase de terminaison du treillis.

La longueur de la fenêtre de terminaison est égale à

$$T = (L - 1)/K \quad branches \tag{2.1}$$

L est la longueur de contrainte du RSC. K est le nombre des sorties d'un RSC.

En pratique, on prend souvent RSC1 = RSC2.

D'après la Fig. 2.1.a, les N bits d'information d sont codés par le codeur RSC1, entrelacés à travers Π et ensuite codés par le codeur RSC2. Chaque code RSC génère $(n_i - k_i) \cdot N/k_i$ bits de parité ou de contrôle noté X^i. Les bits d'information $d = X^1$ et $\prod(X^1)$ étant les mêmes à une permutation près, nous ne transmettrons pas l'entrée du code RSC2 afin d'augmenter le rendement de CTC. [14].

En négligeant la fermeture des deux treillis, le rendement R du CTC s'obtient par l'expression simple

$$R = \frac{N}{J} = \frac{R1R2}{R1 + R2 - R1R2} = \frac{R1}{2 - R1} \; si \; R1 = R2 \qquad (2.2)$$

Ainsi le cas le plus fréquent correspond à $R = 1/3$ avec $R1 = R2 = 1/2$.

2.2.2 Les CTC séries

Un turbocode série est un code linéaire binaire qui génère pour une séquence d'information de longueur K une séquence de sortie de longueur J. Ce turbocode est construit en concaténant de manière série selon le schéma de la Fig. 2.1.b deux codes convolutifs (code1) et (code2).

Le premier code est dit externe et le deuxième est dit code interne. Comme pour le turbocode parallèle, le treillis des deux codes constituant le turbocode série démarre de l'état zéro et se termine dans l'état zéro après $\frac{K}{K1} + T1$ branches pour le premier code et après $\frac{N}{K2} + T2$ branches pour le deuxième code.

En pratique, on prend un code externe NRNSC ou RSC, mais le code interne doit toujours être RSC. D'après la Fig. 2.1.b, les K bits d'information d sont codés par le premier codeur, entrelacés à travers Π et ensuite recodés par le deuxième codeur. Le code externe génère $N = \frac{K}{R1}$ bits codés X^2. Ces bits codés contiennent les bits d'information X^1 si le premier code est RSC. Le code interne RSC génère $J = \frac{N}{R2}$ bits codés X^3 contenant ses bits d'information $\Pi(X^2)$. En négligeant la fermeture des deux treillis, le rendement du CTC série s'obtient par le simple produit $R = \frac{K}{J} = R1 \times R2$. Ainsi, le cas le plus fréquent correspond à $R = 1/2$ avec $R1 = 2/3$ et $R2 = 3/4$ par exemple. Nous remarquons déjà, dans ce

cas, que les codes constituants d'un CTC série ont un rendement plus élevé que les constituants d'un CTC parallèle. Les treillis des codes constituant un CTC série sont alors plus complexes que ceux des codes constituant un CTC parallèle de même rendement. Pour cette raison les CTC parallèles sont favorisés aux CTC séries. [14]. **Dans tout ce qui suit de ce travail on se sera polarisé sur les CTC parallèles.**

2.2.3 L'entrelacement (permutation) dans les CTC

Dans plusieurs applications de communication le canal utilisé soumis à des erreurs qui arrivent en salves (c'est le cas du bruit impulsif comme l'AWGGN de paramètre de forme inférieur à 1). L'utilisation d'une technique appelée « Entrelacement » dans un CTC permet d'améliorer colossalement la capacité de correction du code et de lutter contre les erreurs en salves. Le but de la technique est d'espacer les symboles de bruit consécutifs. Nous pouvons définie l'entrelaceur comme une technique qui prend les symboles d'information d'un alphabet fixe à l'entrée et qui reproduit les même symboles mais dans un ordre temporel différent. [15].

Les types d'entrelaceurs

Il existe deux types ou familles d'entrelaceur : l'entrelacement « bloc » et l'entrelacement « convolutionnel ». La famille d'entrelacement bloc regroupe plusieurs types d'entrelacement comme l'entrelacement bloc classique, pseudo-aléatoire classique, aléatoire, hélicoïdal etc. Nous nous sommes surtout intéressés à l'entrelacement bloc classique et au pseudo-aléatoire.

Un entrelaceur bloc classique est un entrelaceur de période $P = N \times M$ décrit par une matrice de taille $N \times M$. Ce type d'entrelaceur se caractérise par un processus dans lequel les données sont écrites par ligne dans une matrice et ensuite ces données sont lues par colonne. Il existe quatre différentes façons de lire les données. Ces façons dépendent de l'ordre dans lesquels les colonnes GD : de gauche à droite ou DG : de droite à gauche, et les lignes HB : du haut vers le bas ou BH : du bas vers le haut, sont écrites. Ces modes d'opérations sont illustrés à la Fig. 2.2.

Fig. 2.2 Les quatre modes d'opérations d'un entrelaceur bloc classique.

Les entrelaceurs pseudo-aléatoires sont les plus performants pour les turbocodes dans le cas des blocs d'information de tailles moyenne et grande [15]. Leurs inconvénients par rapport aux entrelaceurs blocs classiques résident dans leur complexité. Notons au passage qu'il existe plusieurs façons de décrire des entrelaceurs, ce qui est utile pour les étudier et les concevoir. En particulier, les questions de délai, d'espace mémoire requis et la capacité de disperser ou d'espacer les symboles bruités. La conception des entrelaceurs pour les turbocodes est loin d'être une science exacte d'où la nécessité de valider la performance des entrelaceurs par simulation. Cependant, chaque simulation démontre seulement les performances de l'entrelaceur pour un choix particulier de codeur et de décodeur. Donc, la valeur générale d'une méthode de conception d'un entrelaceur ne fait pas l'unanimité parmi les chercheurs.

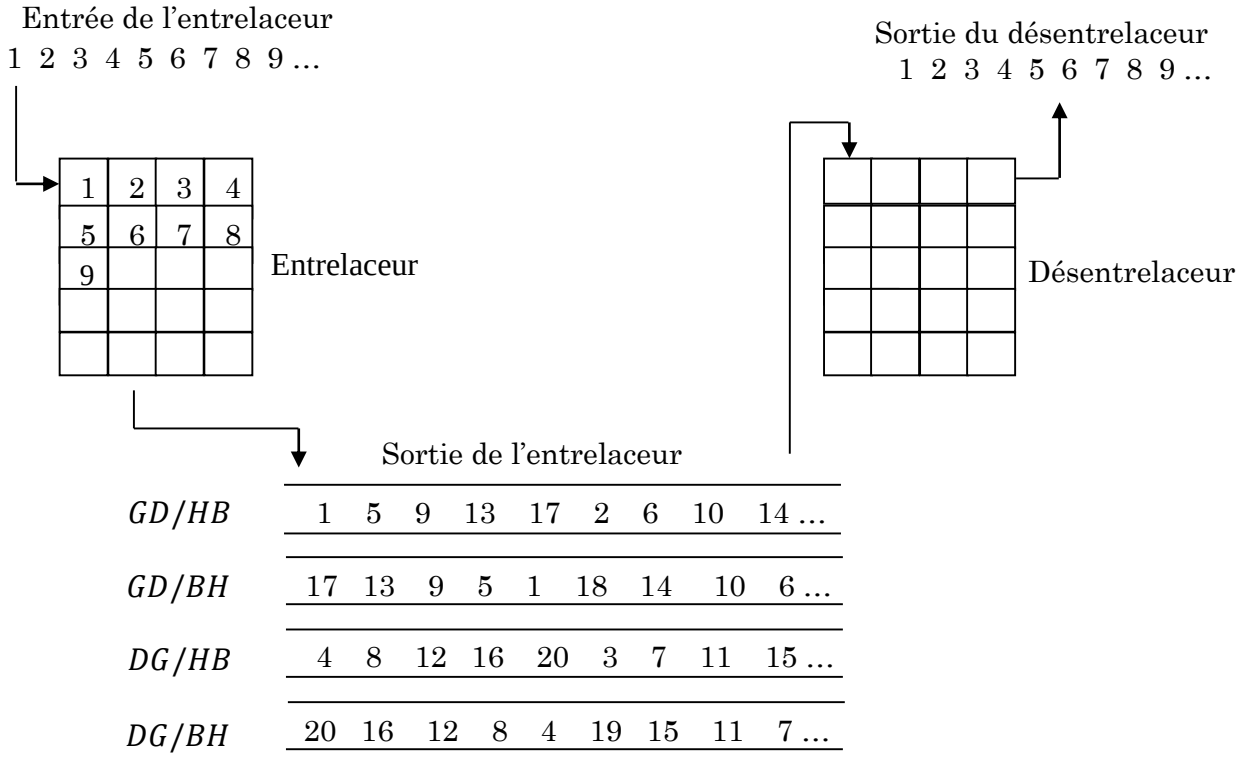


Fig. 2.3 Exemple d'un entrelaceur bloc classique. [15].

Nous allons essayer à l'aide d'un exemple d'illustrer le fonctionnement de l'entrelaceur bloc classique. Supposons qu'une séquence de 20 symboles ($N = 4,\ M = 5,\ P = 20$) soit reçue à l'entrée de l'entrelaceur. Nous allons d'abord placer ces 20 symboles dans une matrice (voir la Fig. 2.3) dans un ordre croissant des nombres indiqués dans les cases (c.à.d. le premier symbole reçu est placé dans la case 0, le deuxième dans la case 1 et ainsi de suite). En appliquant les quatre différents modes d'opérations à la séquence de la Fig. 2.3 les symboles sont réécrits à la sortie de l'entrelaceur bloc classique dans l'ordre indiqué dans la même figure.

2.2.4 Exemple sur les CTC

On considère le code convolutif RSC (5,7) (notation octale) et du rendement $R = 1/2$ de la Fig. 2.4. A partir de ce code RSC, on construit un CTC avec un entrelaceur bloc classique GD/HB de taille 2×2, indiqué dans la Fig. 2.5.

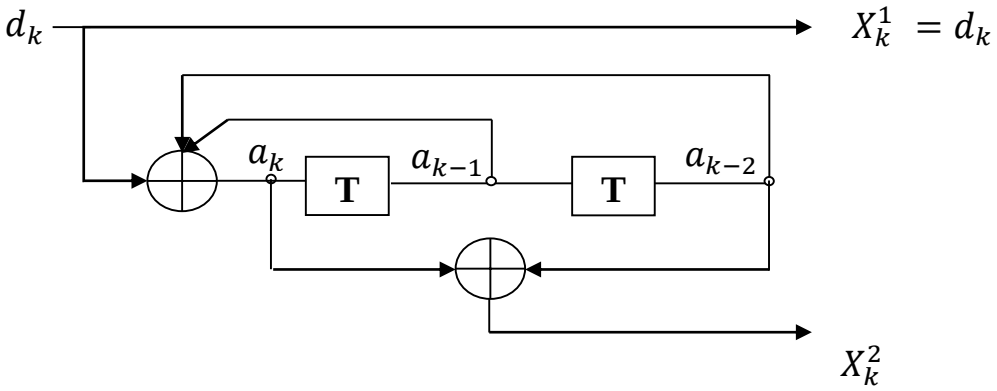


Fig. 2.4 Le codeur RSC [2] (5,7), $R = 1/2$.

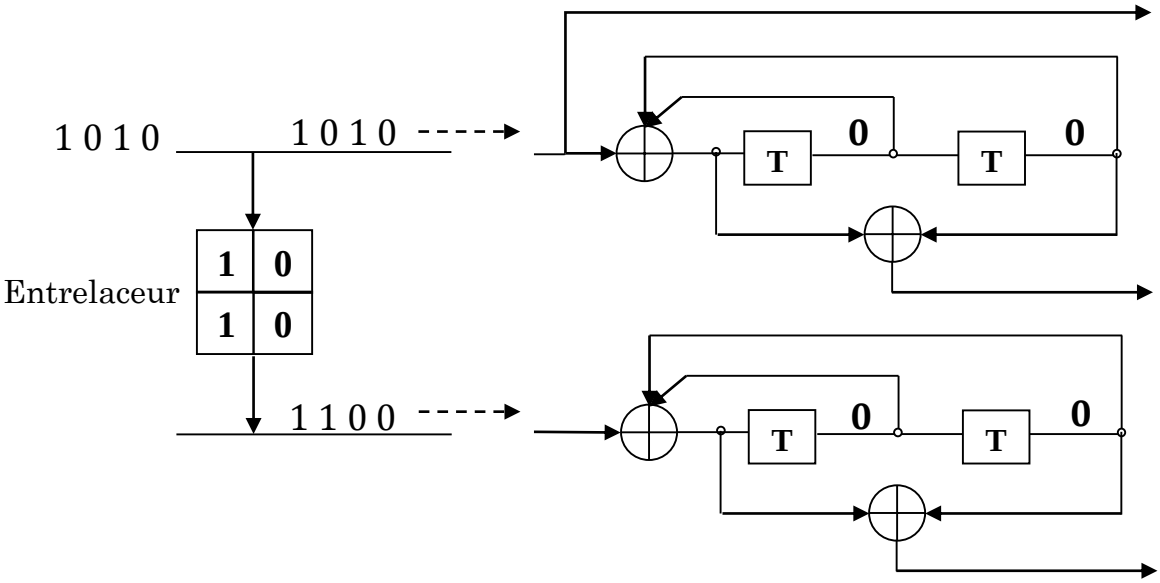
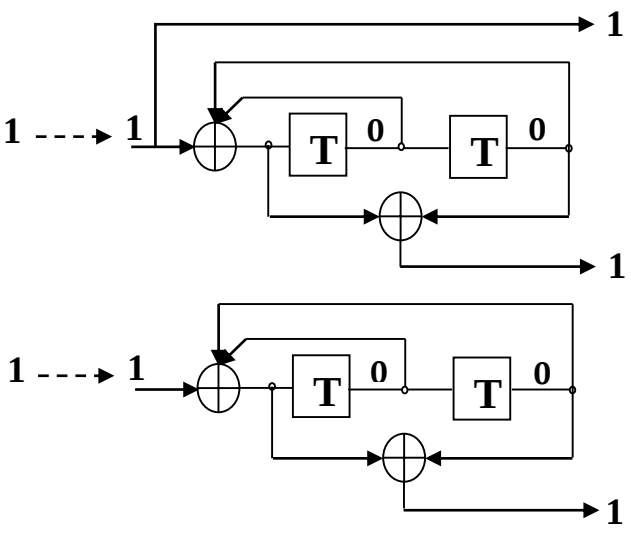


Fig. 2.5 Le codeur CTC basé sur le RSC (5,7), $R = 1/3$.

Les étapes de codage de l'exemple précédent

Premier mot

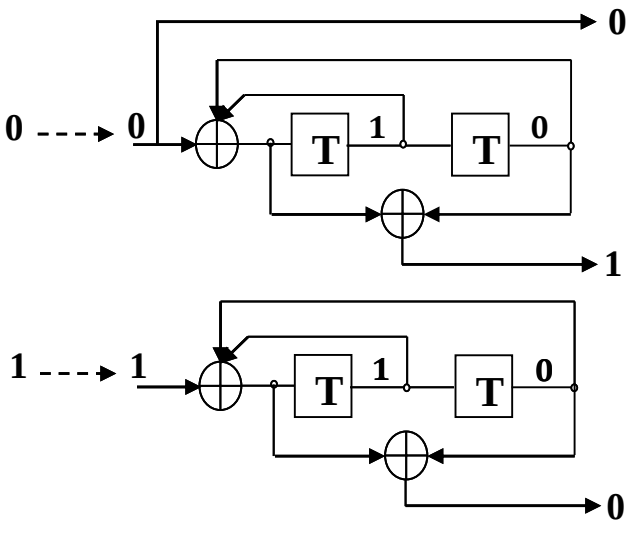


Le ***RSC1*** *: l'entrée est* : **1**, *l'état* : **0 0**.

Le ***RSC2*** *: l'entrée est* : **1**, *l'état* : **0 0**.

Le mot code est : **1 1 1**.

Deuxième mot

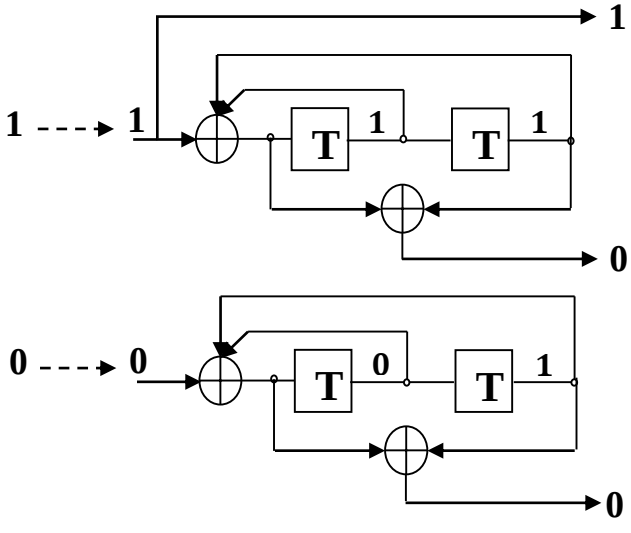


Le ***RSC1*** *: l'entrée est* : **0**, *l'état* : **10**.

Le ***RSC2*** *: l'entrée est* : **1**, *l'état* : **1 0**.

Le mot code est : **0 1 0**.

Troisième mot

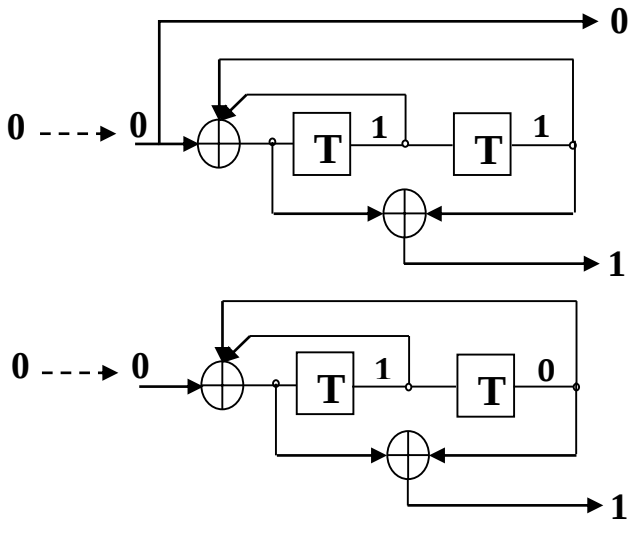


Le ***RSC1*** *: l'entrée est* : **1**, *l'état* : **1 1**.

Le ***RSC2*** *: l'entrée est* : **0**, *l'état* : **0 1**.

Le mot code est : **1 0 0**

Quatrième mot



Le ***RSC1*** *: l'entrée est* : **0**, *l'état* : **1 1**.

Le ***RSC2*** *: l'entrée est* : **0**, *l'état* : **1 0**.

Le mot code est : **0 1 1**

La séquence codée est donc : 1 1 1 0 1 0 1 0 0 0 1 1

On suppose que la séquence d'entrée est : 1 0 1 0, donc, la séquence d'entrée dans le premier codeur est 1 0 1 0. La séquence d'entrée dans le deuxième codeur après l'entrelacement est 1 1 0 0. (voir la figure de la page précédente)

2.3 Fermeture du treillis

Inversement à un code convolutif simple qui fait le codage d'une manière continue, un turbocode convolutif fait le codage de la séquence d'entrée par paquets de taille égale à celle de l'entrelaceur.

Pour le premier paquet de la séquence d'information les codeurs convolutifs du CTC sont à l'état initial, généralement, 00 (dit aussi l'état 0). A seule fin que tous les autres paquets de la séquence en question confrontent le même état (donc les mêmes conditions de codage), on doit forcer le CTC à l'état 0 par l'ajout des bits non informatifs appelés la queue du paquet ou de la séquence, on obtient alors pour chaque paquet de la séquence un treillis fermé.

Pour le CTC de la Fig. 2.5, dont son treillis est illustré dans la Fig. 2.6, les bits de forçage du codeur sont donnés dans le tableau suivant.

L'état $a_{k-1}a_{k-2}$		La queue q_1q_2	
0	0	0	0
1	0	1	1
0	1	1	0
1	1	0	1

Ce tableau donne $q_1 = a_{k-1}(1 - a_{k-2}) + (1 - a_{k-1})a_{k-2}$ et $q_2 = a_{k-1}$

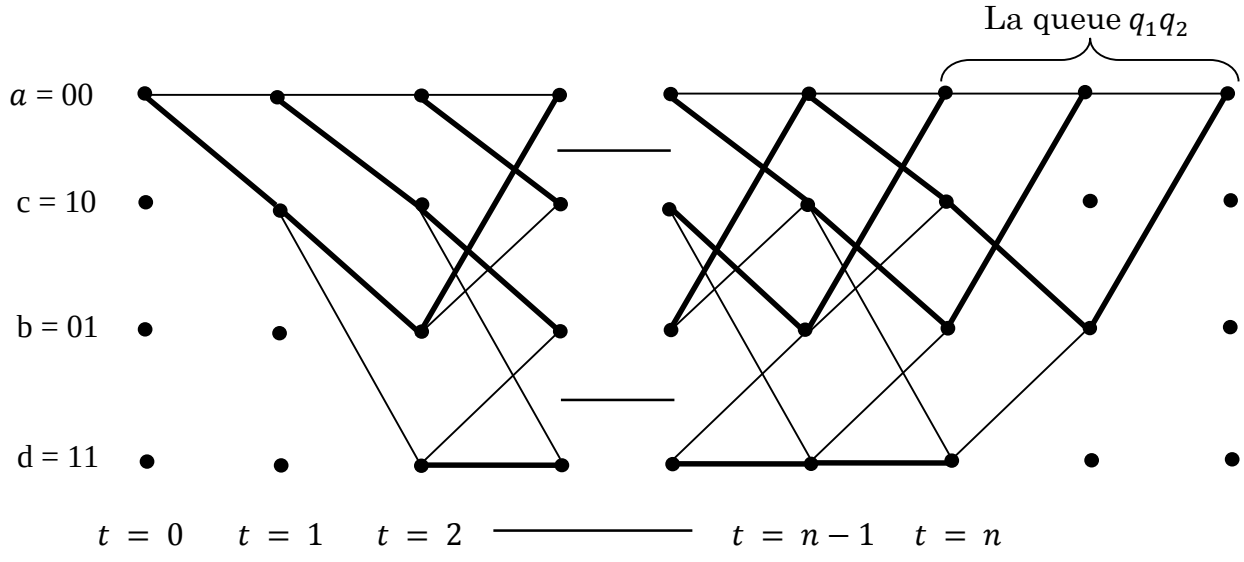


Fig. 2.6 Fermeture du treillis.

2.4 Le décodage itératif et l'Algorithme SOVA

Dans un système de communication numérique sur un canal bruité, un codeur est souvent utilisé à l'émetteur avant l'étape de modulation, afin de corriger les erreurs de transmission au récepteur. Lorsque la structure du code s'y prête, le processus de décodage peut être exécuté en plusieurs étapes ou itérations simples, d'où le nom de décodage itératif.

Le décodage itératif est basé toujours sur un algorithme de décodage SISO. Historiquement de nombreux algorithmes existent pour décoder un code convolutif. Les premiers algorithmes de **Fano** puis **Viterbi** étaient à entrées et sorties binaires. L'algorithme de **Viterbi**, le plus efficace des deux, a d'abord été modifié pour accepter des entrées douces ou souples [18] avec à la clé une amélioration du décodage. La concaténation des codes a ensuite poussé l'apparition d'algorithmes de décodage à entrées et sorties pondérées SISO tels que l'algorithme SOVA [19] et l'algorithme MAP aussi appelé BCJR [20]. Ces deux algorithmes sont couramment utilisés pour le décodage de codes représentés en treillis, particulièrement les codes convolutifs. La différence entre ces deux algorithmes est que le SOVA calcule le chemin dont la vraisemblance est maximale et en plus à chacun des symboles associés aux transitions du chemin choisi, il fait correspondre une information de fiabilité en approximant sa probabilité a posteriori. Il s'agit donc d'un algorithme de décodage à ML par séquence ou par chemin sous optimal. Par contre l'algorithme MAP ou BCJR, il calcule de manière exacte la probabilité a posteriori des symboles présents sur les branches. Il s'agit donc aussi d'un algorithme à décision MAP qui calcule la probabilité de chaque symbole à partir des probabilités de tous les chemins possibles dans le treillis entre l'état initial et l'état final. Il est un algorithme optimal d'une maximisation symbole par symbole.[16] [17]

Puisque notre travail est basé sur les CTC présentés dans le chapitre 1 avec l'algorithme SOVA, nous présentons dans ce paragraphe la notion et le principe de l'algorithme SOVA.

2.4.1 La métrique de Vraisemblance Logarithmique ou de corrélation

Nous définissons dans ce paragraphe une grandeur importante dite la métrique de vraisemblance logarithmique ou la métrique de corrélation [21]. Cette métrique sera utilisée par la suite dans l'algorithme SOVA.

Une fois codés, les bits d'information sont transmis sur le canal. A la réception, partant des mots-code reçus et comportant, à cause du bruit, des erreurs, il faut trouver une règle de décision qui optimise un critère comme par exemple celui de maximiser la probabilité de décision correcte.

Dans un système de communication numérique le canal est caractérisé par un alphabet d'entrée, un alphabet de sortie et une loi de probabilité de la sortie conditionnellement à l'entrée (probabilité de transition).

Dans notre étude on s'intéresse au canal binaire à bruit impulsif blanc additif présenté par une distribution gaussienne généralisée (GGD). Ce canal est à entrée binaire et sortie analogique. La sortie se représente par une variable aléatoire continue R_k dite l'observation du canal.

$$R_k = Y_k + W_k \tag{2.3}$$

où Y_k est le symbole binaire émis et W_k est une variable aléatoire gaussienne centrée de variance σ^2 correspondant au bruit du canal.[21]

Pour une modulation BPSK l'alphabet du symbole Y_k est $\{+\sqrt{E_b}, -\sqrt{E_b}\}$ tel que la fonction de modulation $m(X_k)$ fait correspondre à un bit $X_k = 0$ le symbole $Y_k = +\sqrt{E_b}$ et à un bit $X_k = 1$ le symbole $Y_k = -\sqrt{E_b}$ où X_k représente la sortie binaire du CTC et E_b est l'énergie du signal par bit. Sans perdre de généralité, on peut prendre $\sqrt{E_b} = 1$ et la fonction de modulation devient :

$$Y_k = m(X_k) = \begin{cases} +1 \; pour \; un \; bit \; X_k = 0 \\ -1 \; pour \; un \; bit \; X_k = 1 \end{cases} \tag{2.4}$$

ou aussi $Y_k = 1 - 2X_k = (-1)^{X_k}$

Les densités de probabilité de transition du canal s'écrivent

$$P(R_k/Y_k\ émis) = \frac{\alpha}{2 \cdot A \cdot \Gamma(1/\alpha)} exp\{-|(R_k - Y_k)/A|^{\alpha}\} \qquad (2.5)$$

avec $A = \sigma \left[\frac{\Gamma(1/\alpha)}{\Gamma(3/\alpha)}\right]^{1/2}$, $0 < \alpha \leq 1$ *et* $\Gamma(z) = \int_0^{+\infty} t^{z-1} e^{-t} dt$, $z > 0$

Pour tirer profit de l'information portée par le symbole reçu R_k dans la sortie du canal, on doit étudier avec un soin jaloux les statistiques du bruit impulsif W_k, du coup, il faut prendre toutes les valeurs possibles de la sortie du canal. On parle donc d'un canal à décision douce ou souple ou soft.

Au niveau du récepteur on doit trouver à partir de l'observation R_k une estimation du symbole Y_k de sorte que la probabilité moyenne de l'erreur de décision soit minimisée, donc la probabilité conditionnelle de la relation (2.5) soit maximisée. Le principe de Maximum de Vraisemblance (Maximum Likelihood ML) donne la solution cherchée à ce problème. Le ML cherche à maximiser la probabilité $P(R_k/Y_k émis)$ au sens d'**Euclid**, ça veut dire il maximise le logarithme népérien de cette probabilité. On élimine toutes les constantes dans l'expression de $log(P(R_k/Y_k émis))$ on trouve l'expression à maximiser qui est $-|(R_k - Y_k)/A|^{\alpha}$. Maximiser la dernière expression revient à minimiser la quantité $|R_k - Y_k|$. Pour que cette dernière quantité soit minimale il faut que R_k soit de même signe que R_k, autrement dit, il faut que le produit $R_k \times Y_k$ soit positif. On peut donc définir **la métrique de Vraisemblance Logarithmique ou de corrélation** comme le produit $M = R_k \times Y_k$ qui est une mesure de la probabilité $P(R_k/Y_k émis)$.

Pour calculer la métrique M on utilise la règle suivante **« si $X_k = 0$, alors $Y_k = 1$, la métrique $M = R_k$ (pas de changement de signe de l'observation R_k), et si $X_k = 1$, alors $Y_k = -1$, la métrique $M = -R_k$ (il y a un changement de signe de R_k) »** [21].

2.4.2 Algorithme de Viterbi à sortie douce SOVA

2.4.2.1 Algorithme de Viterbi classique avec la métrique de corrélation

Parmi les algorithmes les plus utilisés dans le décodage des codeurs convolutifs est celui de **Viterbi** classique qui est surtout basé sur le diagramme en

treillis du codeur. Cet algorithme suit la règle de décision du ML. Le décodage optimal au sens du ML d'un code convolutif est obtenu par la recherche dans le treillis du code de la séquence la plus probable au moyen de cet algorithme .Selon la nature de l'estimation fournie par le démodulateur, dure ou douce, les métriques utilisées seront respectivement une métrique de **Hamming** ou une métrique euclidienne (métrique de corrélation). Une métrique cumulative dite métrique de branches est évaluée le long du treillis du code. Nous appelons survivants les chemins conservés à une certaine étape du décodage. Il y a un unique survivant par état, et autant de survivants que d'états. Chaque survivant est associé à une valeur de la métrique. A un instant donné, le choix entre deux chemins possibles se fait en conservant celui de métrique maximale. Si deux ou plusieurs chemins ont la même métrique, nous en choisissons aléatoirement un survivant parmi les candidats possibles.

Pour présenter clairement l'algorithme de **Viterbi** classique avec la métrique euclidienne de corrélation définie dans le paragraphe précédent, nous supposons que le canal de transmission soit de type stationnaire gaussien et sans mémoire avec une modulation BPSK. Le codeur convolutif a une longueur de contrainte L. L'algorithme peut être partagé en deux parties

- La première s'étend du début du treillis, $k = 1$ jusqu'au niveau $k = L = \nu + 1$, où L est la longueur de contrainte et ν est le nombre des registres (mémoire du codeur). Le décodage consiste à attribuer une métrique nulle à l'état initial, et à chaque autre état $S^{(m2)}$, atteint par la transition d'un état $S^{(m1)}$, une métrique, qui est la somme des métriques à l'état $S^{(m1)}$ et la métrique de transition ou de branche. La métrique de branche à un niveau k du treillis entre deux états $S^{(m1)}$ et $S^{(m2)}$ notée $BM_{(m1)}^{(m2)}$, est la somme des métriques de corrélation des symboles reçus à ce niveau. [22][18][17].
- La deuxième partie s'étend du niveau $k = L + 1$ jusqu'au niveau $k = \left[\frac{N}{m_e}\right] + L - 2$ où N est la longueur du message, et m_e est le nombre des entrées du codeur.

Le décodage consiste à choisir la branche pour laquelle la métrique au nœud $S^{(m)}$ sera maximale et éliminer toutes les autres. La branche choisie est

appelée *survivant* et les autres branches sont appelées *concurrents*. Si le choix de plusieurs branches donne la même métrique maximale au nœud $S^{(m)}$, alors peu importe, nous choisirons une de ces branches. Cette opération est répétée avec tous les 2^{ν} états, et cela pour tous les niveaux de cette partie. A la fin de cette partie nous obtiendrons 2^{ν} chemins, desquels le décodeur choisira celui qui a la métrique de corrélation la plus grande.

En notant, qu'il est nécessaire de stoker 2^{ν} chemins jusqu'à la fin, où la décision sera prise, nous se rendons compte que le décodeur devient très complexe et nécessite une capacité mémoire importante pour une longueur de contrainte importante.

Toutefois, lorsque nous examinons les différents survivants à une étape k, nous remarquons qu'avec une grande probabilité, ils ont tous le même passé entre les étapes 1 et $k - dp + 1$ où dp est la profondeur de décision [23].

Nous pouvons par conséquent décider définitivement à l'étape k toutes les données émises jusqu'à l'étape $k - dp$. Comme règle empirique, nous pouvons fixer le paramètre dp égal à 5 ou 6 fois le nombre d'états du décodeur [23]. Avec ces valeurs les performances du décodeur sont quasiment les mêmes que lorsque la décision se fait en fin de la séquence.

Cette stratégie permet de réduire la complexité du décodeur et le retard de décision. La complexité d'un décodeur de **Viterbi** est proportionnelle au nombre d'états de treillis. Celui-ci croit exponentiellement avec la longueur de contrainte. Pour cette raison les codes utilisés en pratique ont une longueur de contrainte inférieure à 10 [23].

Pour un CTC la profondeur de décision dp peut être prise égale à la taille de l'entrelaceur si cette dernière n'est pas importante.

Exemple 2.1

Considérons le codeur RSC (5,7) de la figure 2.4 dont la représentation en treillis est donnée par la figure 2.7, le décodage à ML consiste à choisir le chemin correspondant à la séquence des symboles qui possède une métrique de corrélation maximale.

Soit $d = (11011)$ la séquence à envoyer. La sortie du codeur RSC (5,7) est $X = (1110001011)$. La sortie du modulateur est $Y = (-1-1-1+1+1+1-1+1-1-1)$. Cette dernière séquence est envoyée dans le canal bruité. Nous supposons que nous avons trouvé à la sortie du canal les observations $R = (-4-3\ -3-2+2+4-3+3-3+1)$.

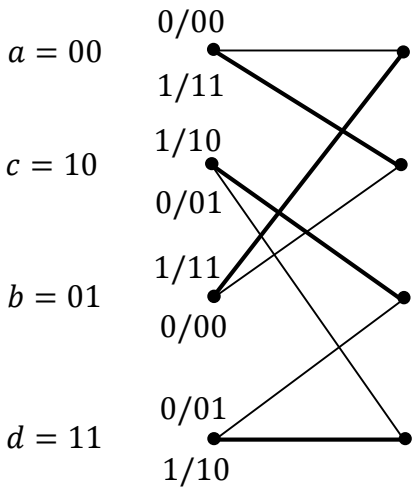
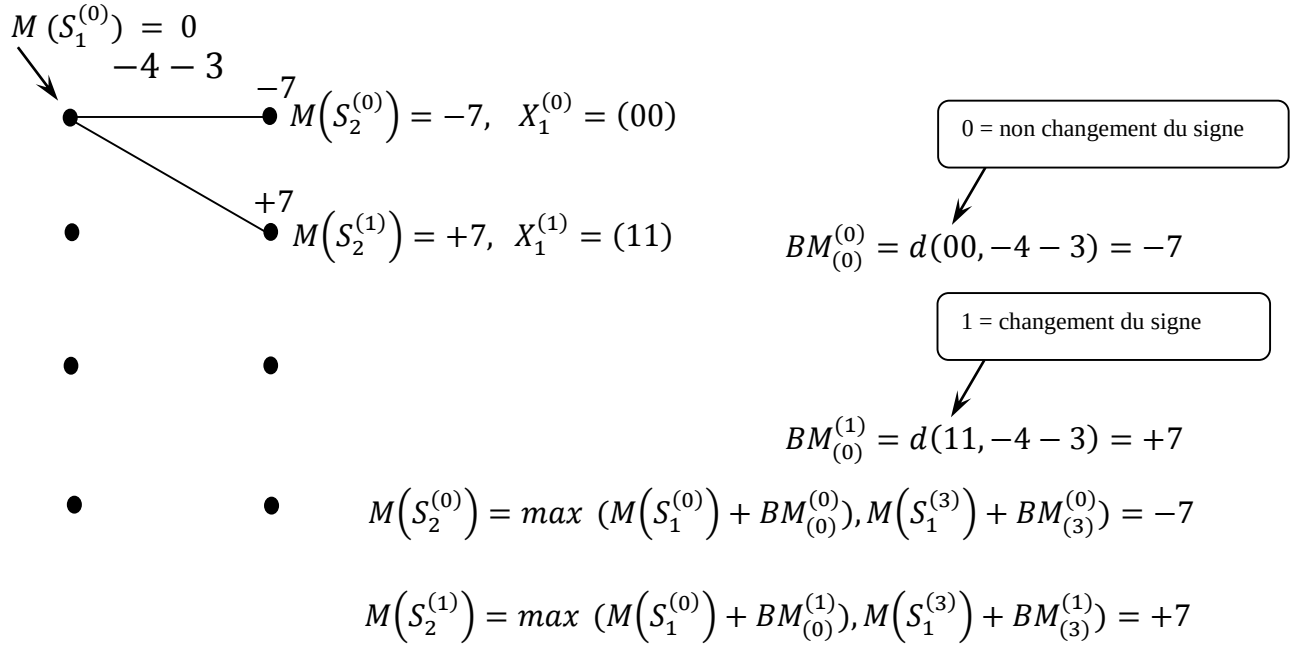
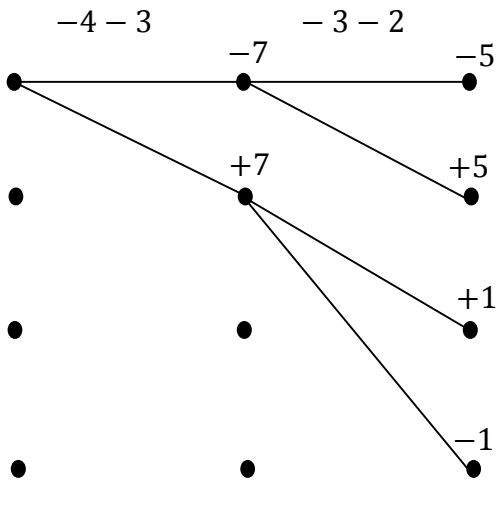


Fig. 2.7 Le treillis simplifié du RSC (5,7).



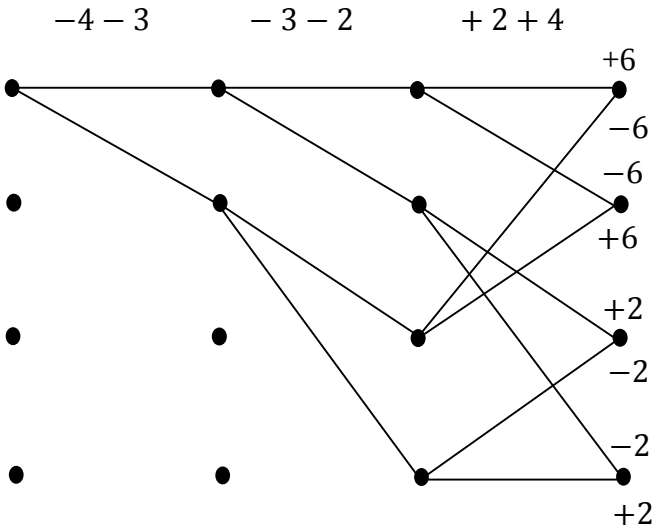


$M\left(S_2^{(0)}\right) = -12, \quad X_2^{(0)} = (00,00)$

$M\left(S_2^{(1)}\right) = -2, \quad X_2^{(1)} = (00,11)$

$M\left(S_2^{(2)}\right) = +8, \quad X_2^{(2)} = (11,10)$

$M\left(S_2^{(3)}\right) = +6, \quad X_2^{(3)} = (11,01)$

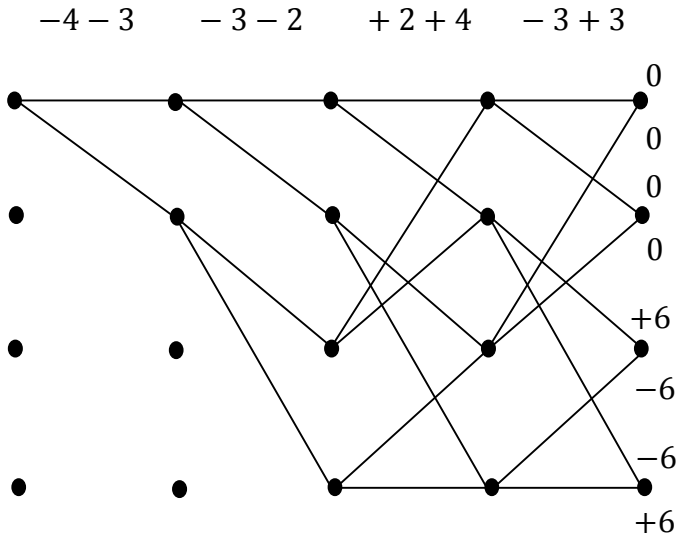


$M\left(S_3^{(0)}\right) = +2, \quad X_3^{(0)} = (11,10,11)$

$M\left(S_3^{(1)}\right) = +14, \quad X_3^{(1)} = (11,10,00)$

$M\left(S_3^{(2)}\right) = +4, \quad X_3^{(2)} = (00,11,10)$

$M\left(S_3^{(3)}\right) = +8, \quad X_3^{(3)} = (11,01,10)$



$M\left(S_4^{(0)}\right) = +4, \quad X_4^{(0)} = (00,11,10,11)$

$M\left(S_4^{(1)}\right) = +4, \quad X_4^{(1)} = (00,11,10,00)$

$M\left(S_4^{(2)}\right) = +20, \quad X_4^{(2)} = (11,10,00,10)$

$M\left(S_4^{(3)}\right) = +14, \quad X_4^{(3)} = (11,10,00,01)$

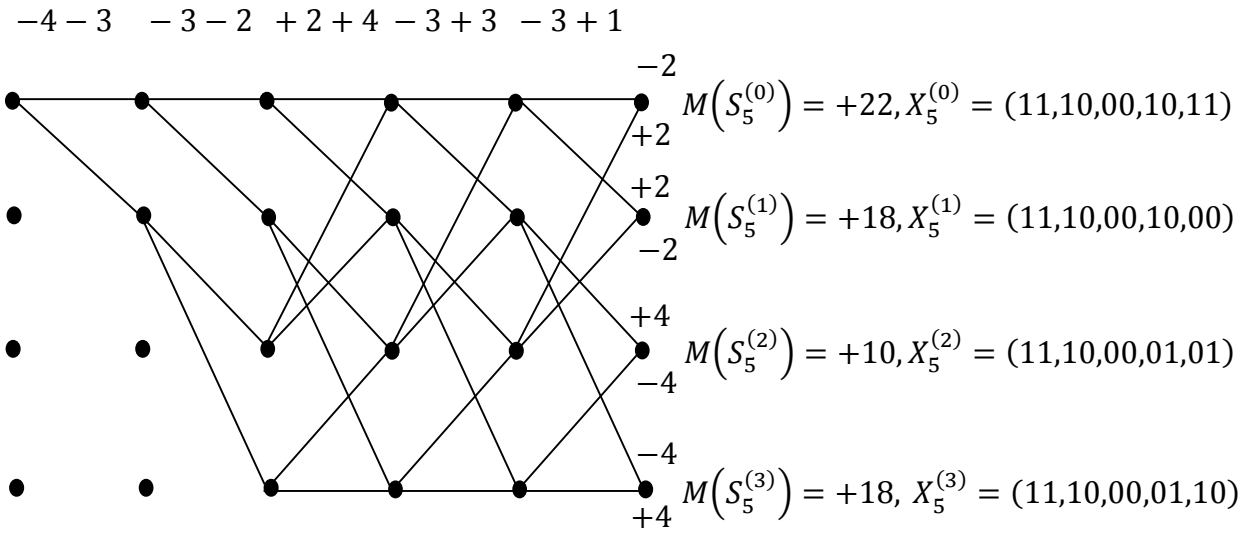


Fig. 2.8 L'algorithme de Viterbi classique avec métrique de corrélation pour le RSC (5,7).

Etat/Etape	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$
$S_k^{(0)}$	0	−7	−12	2	4	22
$S_k^{(1)}$	Ø	7	−2	14	4	18
$S_k^{(2)}$	Ø	Ø	8	4	20	10
$S_k^{(3)}$	Ø	Ø	6	8	14	18

Tab. 2.1 Tableau des survivants du treillis de la figure 2.8.

D'après le treillis illustré ci-dessus, le survivant est le chemin qui passe par les états : $S_1^{(0)}S_2^{(1)}S_3^{(2)}S_4^{(1)}S_5^{(2)}S_6^{(0)}$, qui correspond à une séquence de sortie du codeur $X_5^{(0)} = (11,10,00,10,11)$ et une séquence d'entrée estimée $\hat{d} = (11011)$, qui est exactement la séquence envoyée.

Il est important de signaler qu'un décodeur à décision ferme donne à partir de la séquence reçue R la séquence estimée $\hat{Y} = (-1-1-1\underline{\mathbf{-1}}+1+1-1+1-1\underline{\mathbf{+1}})$ qui contient deux erreurs mentionnées en gras.

2.4.2.2 Algorithme de Viterbi à sortie douce SOVA

L'algorithme SOVA est une modification de l'algorithme de **Viterbi** classique pour lui permettre d'être utilisé comme élément de décodage dans le turbo décodeur [24][25]. Cette modification réside en deux points

- Le calcul des métriques de branches est modifié pour permettre de prendre en compte les informations a priori.
- L'algorithme est modifié de façon qu'il produise une information de sortie douce sous la forme d'information a posteriori d'un rapport logarithmique de vraisemblance LLR pour chaque bit décodé.

Autrement dit, l'algorithme de **Viterbi** est modifié afin de fournir à sa sortie une valeur de confiance ou de fiabilité (approximant une probabilité a posteriori) associée à chaque bit décodé. La valeur des bits décodés est toujours donnée par le chemin ξ ayant la métrique de branches (ou la métrique cumulée) maximale $M(\xi)$ fournie par l'algorithme classique.

La probabilité que le chemin ξ soit émis sachant la séquence R soit reçue est proportionnelle à la probabilité conditionnelle de la séquence R soit reçue sachant le chemin ξ soit émis. Notons $M(1)$ la métrique de branches maximale jusqu'à l'étape k dans le treillis avec le bit estimé de d_k est $\hat{d}_k = 1$. De même, notons $M(0)$ la métrique de branches maximale jusqu'à l'étape k dans le treillis avec le bit estimé de d_k est $\hat{d}_k = 0$. Alors le rapport logarithmique de vraisemblance du bit d_k dans le chemin ξ est approximé par [26]

$$LLR(d_k) \triangleq \ln\left(\frac{P(d_k = 1|R)}{P(d_k = 0|R)}\right) \cong M(1) - M(0) \qquad (2.6)$$

Cette valeur s'appelle la décision douce ou souple correspondant au bit d_k. La relation (2.6) contient trois membres. Le troisième membre est calculé à partir du treillis du codeur en appliquant l'algorithme de **Viterbi** classique avec la métrique de corrélation euclidienne. Quant au deuxième membre, nous pouvons le développer de plus dans le but de calculer ce que nous appelons l'information extrinsèque. Pour ce but, nous devons passer par les fondations mathématiques du décodage à décisions douces qui reposent sur le théorème de **Bayes**.

Soient A_i, $i = 1,2,\cdots,K$, les signaux à transmettre d'un ensemble donné de signaux, et soient $P(A_i)$ les probabilités a priori de ces signaux. Soit B le signal reçu qui n'est qu'un des signaux A_i corrompu par du bruit. B est une variable aléatoire continue de probabilité $P(B)$. L'expression $P(A_i|B)$ est la probabilité a posteriori APP du signal A_i sachant que le signal reçu B a été observé. Le détecteur MAP cherche à maximiser l'APP $P(A_i|B)$. Cette dernière s'exprime sous la forme [22]

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)} \tag{2.7}$$

où $P(B|A_i)$ est la probabilité conditionnelle du signal observé B, et $P(A_i)$ est la probabilité a priori du $i^{ème}$ signal transmis. Le dénominateur de l'équation (2.7) est donné par

$$P(B) = \sum_{i=1}^{K} P(B|A_i)P(A_i) \tag{2.8}$$

Notons que $P(B|A_i)$ est appelée aussi la fonction de vraisemblance. Le critère de décision qui consiste à prendre le maximum de $P(B|A_i)$ parmi les K signaux est le principe du ML si les signaux sont équiprobables.

Si la distribution des signaux A_i est uniforme, c'est-à-dire les signaux A_i sont équiprobables, or $P(B)$ est indépendant du signal A_i transmis, donc maximiser $P(A_i|B)$ revient à maximiser l'expression $P(B|A_i)$. Donc le détecteur basé sur le principe MAP fera la même décision que celui basé sur le principe du ML.

Dans notre étude les signaux A_i sont les signaux binaires d_k, donc $K = 2$, et le signal reçu B représente la séquence R observée à la sortie du canal. Alors l'équation (2.7) devient

$$P(d_k = i|R) = \frac{P(R|d_k = i)P(d_k = i)}{P(R)}, \quad i = 0,1 \tag{2.9}$$

La règle de décision MAP consiste à comparer les probabilités APP $P(d_k = 0|R)$ et $P(d_k = 1|R)$, ensuite à en prendre le maximum, ceci peut se résumer dans l'équation suivante

$$P(d_k = 1|R) \lessgtr_{H_1}^{H_0} P(d_k = 0|R) \tag{2.10}$$

Dans l'équation (2.10) l'hypothèse H_1 signifie que le détecteur MAP assigne d_k à la valeur 1 dans le cas ou $P(d_k = 1|R)$ est supérieur à $P(d_k = 0|R)$. Dans le cas contraire le détecteur MAP choisit $d_k = 0$ ce qui correspond à l'hypothèse H_0.

La décision ferme optimale qui donne $\hat{d}_k$ au sens du MAP est alors la suivante

$$\hat{d}_k = \begin{cases} 1 & si \quad P(d_k = 1|R) > P(d_k = 0|R) \\ 0 & sinon. \end{cases} \tag{2.11}$$

La décision douce est définie par le rapport logarithmique de vraisemblance LLR comme suit (voir l'équation (2.6))

$$LLR(d_k) \triangleq \ln\left[\frac{P(d_k = 1|R)}{P(d_k = 0|R)}\right] \tag{2.12}$$

Il s'ensuit, d'après (2.11) et (2.12) que le signe de la décision douce détermine la décision ferme et que la valeur absolue de la décision douce détermine la fiabilité de cette décision. Il est important à signaler que le LLR obtenu est valable pour un seul bit ou symbole.

Si nous notons par $\overline{R_k}$ toute la séquence reçue R sauf R_k qui est le symbole reçu à l'instant k, nous pouvons écrire grâce à la loi de **Bayes**

$$P(R|d_k = i) = P(\overline{R_k}\,|R_k, d_k = i) \cdot P(R_k|d_k = i), \qquad i = 0, 1 \tag{2.13}$$

et avec la relation (2.9), le rapport $LLR(d_k)$ peut-être décomposé sous la forme [21][27]

$$LLR(d_k) = \underbrace{\ln\left(\frac{P(R_k|d_k = 1)}{P(R_k|d_k = 0)}\right)}_{LLR_c(R_k)} + \underbrace{\ln\left(\frac{P(\overline{R_k}\,|R_k, d_k = 1)}{P(\overline{R_k}\,|R_k, d_k = 0)}\right)}_{LLR_e(\overline{R_k})} + \underbrace{\ln\left(\frac{P(d_k = 1)}{P(d_k = 0)}\right)}_{LLR_a(d_k)} \tag{2.14}$$

où $LLR_c(R_k) = \ln\left(\frac{P(R_k|d_k = 1)}{P(R_k|d_k = 0)}\right)$ est le rapport logarithmique de vraisemblance du canal, $LLR_e(\overline{R_k}) = \ln\left(\frac{P(\overline{R_k}\,|R_k, d_k = 1)}{P(\overline{R_k}\,|R_k, d_k = 0)}\right)$ est le rapport logarithmique de vraisemblance représentant la connaissance acquise grâce au processus de

décodage et $LLR_a(d_k) = \ln\left(\frac{P(d_k=1)}{P(d_k=0)}\right)$ est le rapport logarithmique de vraisemblance a priori.

Finalement, nous pouvons écrire la sortie douce $LLR(d_k)$ d'un décodeur SOVA comme suit [21][27]

$$LLR(d_k) = LLR_c(R_k) + LLR_e(\overline{R_k}) + LLR_a(d_k) = LLR_c(R_k) + \Psi(d_k) \tag{2.15}$$

où la nouvelle quantité $\Psi(d_k) = LLR_e(\overline{R_k}) + LLR_a(d_k)$ est appelée **information extrinsèque**, qui est indépendante de R_k.

Le critère de décision au sens du MAP peut alors s'écrire

$$\hat{d}_k = \begin{cases} 1 & si \quad LLR(d_k) > 0 \\ 0 & sinon. \end{cases} \tag{2.16}$$

Il s'ensuit que le signe de la décision douce détermine la décision ferme et que la valeur absolue de la décision douce détermine la fiabilité de cette décision.

En particulier, si nous considérons un bruit de type AWGGN de moyenne nulle et de variance σ^2 qui sera utilisé par la suite dans le chapitre 3, le rapport logarithmique de vraisemblance du canal a pour expression

$$LLR_c(R_k) = \left(\sigma^2 \frac{\Gamma(1/\alpha)}{\Gamma(3/\alpha)}\right)^{-\alpha/2} \cdot (|R_k - 1|^\alpha - |R_k + 1|^\alpha) \tag{2.17}$$

et l'information extrinsèque $\Psi(d_k)$ devient alors

$$\Psi(d_k) = \left(\sigma^2 \frac{\Gamma(1/\alpha)}{\Gamma(3/\alpha)}\right)^{-\alpha/2} \cdot (|R_k + 1|^\alpha - |R_k - 1|^\alpha) + LLR(d_k)$$

et pour $\alpha = 2$, on trouve

$$LLR_c(R_k) = -\frac{2}{\sigma^2} R_k \tag{2.18}$$

et l'information extrinsèque $\Psi(d_k)$ devient alors

$$\Psi(d_k) = \frac{2}{\sigma^2} R_k + LLR(d_k) \tag{2.19}$$

Remarque

Généralement $\Psi(d_k)$ et $LLR(d_k)$ sont de même signe [2], alors

$$\hat{d}_k = \begin{cases} 1 & si \quad \Psi(d_k) > 0 \\ 0 & sinon. \end{cases} \tag{2.20}$$

Calcul pratique des $LLR(d_k)$

L'approximation du $LLR(d_k)$ indiquée dans l'équation (2.6) est utilisable lorsque nous appliquons l'algorithme de **Viterbi** classique avec la métrique de corrélation dans un seul sens du treillis (sens d'aller seulement). Pour améliorer amplement cette approximation, nous devons utiliser en plus le sens de retour du treillis, c'est-à-dire nous devons appliquer l'algorithme de **Viterbi** classique dans les deux sens du treillis, aller et retour après avoir fermé ce treillis. La fermeture du treillis pour l'algorithme SOVA est bien illustré dans le paragraphe 2.3 de ce chapitre. Avec les mêmes données du paragraphe 2.4.2.1, l'algorithme SOVA à utiliser consiste à appliquer l'algorithme de **Viterbi** classique dans les deux sens du treillis, pour calculer le $LLR(d_k)$ de chaque bit d'information d_k, en respectant les étapes suivantes

- **Etape 1 : Le sens d'aller ou direct :** Dans ce sens nous appliquons l'algorithme de **Viterbi** classique sur le treillis du codeur de l'étape $k = 1$ jusqu'à l'étape finale $k = N + 1$, où N est la longueur de l'information envoyée. Cette étape a deux buts. Le premier but est de calculer et sauvegarder toutes les métriques de branche qui lient les états entre eux. Le deuxième but est de trouver le survivant qui correspond à la métrique maximale M_{max}. Les métriques sauvegardées sont utilisées par la suite dans le calcul des LLR.

- **Etape 2 : Le sens de retour ou inverse :** Le but de cette étape est le calcul des contributions minimales dues à la prolongation des chemins (à chaque niveau de profondeur et pour chaque nœud). Nous appliquons toujours l'algorithme de **Viterbi** classique sur le treillis du codeur mais dans le sens inverse, donc de l'étape finale $k = N + 1$ jusqu'à l'étape initiale $k = 1$. Dans cette deuxième phase, nous ne calculons et ne sauvegardons que les métriques des états dans toutes les étapes du treillis, parce que les métriques de branche sont déjà calculées et sauvegardées dans l'étape 1.

- **Etape 3 : Calcul des $LLR(d_k)$:** Le survivant trouvé dans l'étape 1 qui possède une métrique notée M_{max}, donne la séquence estimée de la vraie séquence d'information. Le $LLR(d_k)$ d'un bit d_k est pratiquement donné par [21][28]

$$LLR(d_k) = M(1) - M(0), \qquad M(d_k) = M_{max} \tag{2.21}$$

où $M(i)$ est la métrique de branches maximale jusqu'à l'étape k avec la valeur estimée de d_k qui est $\hat{d}_k = i$ où $i = 0, 1$. La branche du survivant dans l'étape k, qui correspond au bit d_k, possède plusieurs concurrents qui correspondent au complément de d_k noté $d_k \oplus 1$. La métrique de branches du meilleur concurrent est donnée par la relation [21][28][29]

$$M(d_k \oplus 1) = \max_{m1,m2} \left\{ M_f\left(S_{k-1}^{(m1)}\right) + BM(d_k \oplus 1) + M_b(S_k^{(m2)}) \right\} \tag{2.22}$$

où $m1,\ m2 \in \{0, 1, 2, \cdots, 2^{\nu} - 1\}$, $M_f\left(S_{k-1}^{(m1)}\right)$ est la métrique de survivant dans l'état $S_{k-1}^{(m1)}$ à l'étape $k-1$ dans le sens d'aller (direct), $BM(d_k \oplus 1)$ est la métrique de branche correspond au complément de d_k qui lie les deux états $S_{k-1}^{(m1)}$ et $S_k^{(m2)}$ et $M_b(S_k^{(m2)})$ est la métrique de survivant dans l'état $S_k^{(m2)}$ à l'étape k dans le sens de retour (inverse).

L'algorithme SOVA à utiliser nécessite alors d'appliquer l'algorithme de **Viterbi** classique avec la métrique de corrélation deux fois : la première est dans le sens d'aller (forward) et la seconde est dans le sens inverse (backward). C'est pourquoi cet algorithme est parfois appelée SOVA bidirectionnel.

Pour tirer au clair le calcul pratique du $LLR(d_k)$ nous donnons l'exemple suivant.

Exemple 2.2

Nous considérons les mêmes données de l'exemple 2.1. Le treillis fermé illustré ci-dessous dans la figure 2.9, contient toutes les métriques de branches dans les deux sens. Ce qui sont indiquées en orange représentent les métriques des branches, celles indiquées en noir sont les métriques cumulatives sur les états en aller et celles indiquées en bleu sont les métriques cumulatives sur les états en retour. Le chemin dessiné en rouge représente le survivant. Une branche

fine ou simple correspond à un bit d'information égal à 0 et une branche en gras correspond à un bit d'information égal à 1.

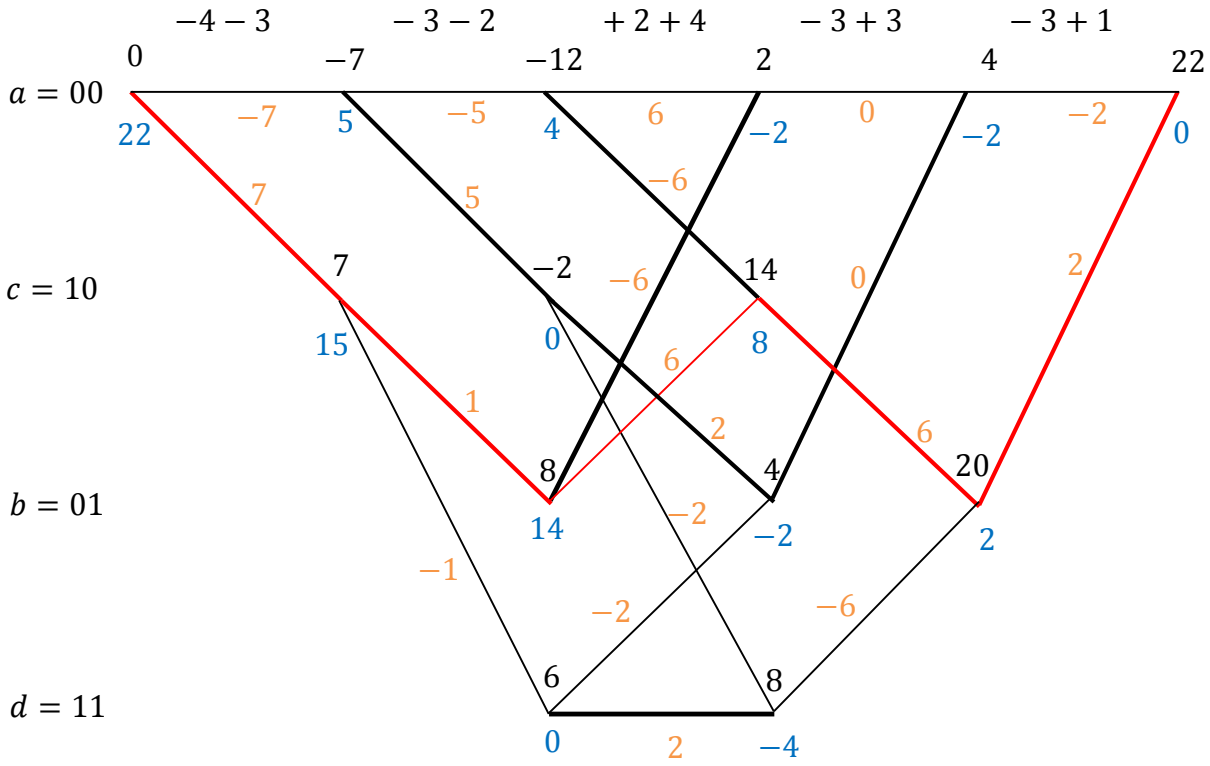


Fig. 2.9 Calcul des métriques de branches dans les deux sens du treillis.

Le survivant est le chemin qui passe par les états : $S_1^{(0)}S_2^{(1)}S_3^{(2)}S_4^{(1)}S_5^{(2)}S_6^{(0)}$, qui correspond à une séquence d'entrée estimée $\hat{d} = (11011)$, et sa métrique $M_{max} = 22$.

Pour le premier bit $d_1 = 1$, $M(1) = M(d_1) = 22$ et $M(0) = M(d_1 \oplus 1) = 0 - 7 + 5 = -2$, donc $LLR(d_1) = 22 - (-2) = 24$.

Pour le deuxième bit $d_2 = 1$, $M(1) = M(d_1) = 22$ et $M(0) = M(d_1 \oplus 1) = \max(7 - 1 + 0, -7 - 5 + 4) = 6$, donc $LLR(d_2) = 22 - 6 = 16$.

Pour le troisième bit $d_3 = 0$, $M(1) = M(d_1 \oplus 1) = \max(0{,}2, -10{,}8) = 8$ et $M(0) = M(d_1) = 22$, donc $LLR(d_3) = 8 - 22 = -14$.

Pour le quatrième bit $d_4 = 1$, $M(1) = M(d_1) = 22$ et $M(0) = M(d_1 \oplus 1) = \max(0{,}2) = 2$, donc $LLR(d_4) = 22 - 2 = 20$.

Pour le cinquième bit $d_5 = 1$, $M(1) = M(d_1) = 22$ et $M(0) = M(d_1 \oplus 1) = 2$, donc $LLR(d_5) = 22 - 2 = 20$.

Par conséquent, $LLR(d) = (24, 16, -14, 20, 20)$ et nous pouvons vérifier facilement que $d_k = (1 + \text{sign}(LLR(d_k))/2$.

2.5 Schéma de principe d'un décodeur itératif

Nous considérons les turbocodes CTC formés de deux codes convolutifs RSC identiques concaténés en parallèle comme les montre la figure 2.1.a. Au niveau du récepteur le signal reçu est $R_k = Y_k + W_k$, où Y_k sont les symboles émis et W_k sont des variables aléatoires i.i.d qui représentent les observations du bruit à moyenne nulle. A l'aide d'un démultiplexeur 1 vers 3, la séquence d'observations R_k se dégroupe en trois sous séquences R_k^1, R_k^2 et R_k^3.

Le décodage itératif de ces turbocodes peut être réalisé à partir de deux décodeurs à entrées-sorties douces DEC1 et DEC2 associés selon le principe représenté sur la figure 2.10. Si l'algorithme SISO utilisé est optimal au sens du ML, ces deux décodeurs élémentaires sont ainsi, mais malgré ceci le décodage itératif ne l'est pas.[11]

Les symboles de redondance bruités R_k^2, R_k^3 sont envoyés vers les décodeurs DEC1 et DEC2 respectivement lorsque la redondance est produit par les codeurs RSC1 et RSC2 respectivement. Le décodeur DEC1 reçoit des symboles bruités R_k^1 et R_k^2 issus du démodulateur et produit une décision relative à chaque symbole d_k. Un entrelaceur Π, se place entre les deux décodeurs élémentaires permet d'éclater les paquets d'erreurs produit par le décodeur DEC1. Les décodeurs DEC1 et DEC2 travaillant à partir de décisions pondérées ou douces, le décodeur DEC1 associe à chaque symbole décodé d_k une mesure de fiabilité sous forme du logarithme de son rapport de vraisemblance $LLR_1(d_k)$, et DEC2 associe au même symbole d_k une autre mesure de fiabilité $LLR_2(d_k)$. L'information extrinsèque $\Psi_1(d_k)$ extraire du décodeur DEC1 est réinjectée à l'itération suivante dans le décodeur DEC2 afin de bénéficier de la diversité de codage, et l'information extrinsèque $\Psi_2(d_k)$ extraire du décodeur DEC2 passe par le désentrelaceur puis elle est réinjectée dans le décodeur DEC1. Le processus pouvant se répéter plusieurs fois (ce qui justifie l'appellation décodage itératif), et les deux décodeurs DEC1 et DEC2 doivent converger vers la même solution. Après un nombre précisé des itérations, une décision ferme du décodeur DEC2 (ou DEC1) donne la

valeur estimée $\hat{d}_k$ du bit originale d_k en basant sur le signe du rapport $LLR_2(d_k)$ (ou $LLR_1(d_k)$) : $\hat{d}_k = \frac{1}{2} \cdot \left(1 + \text{sign}\left(LLR_2(d_k)\right)\right)$.

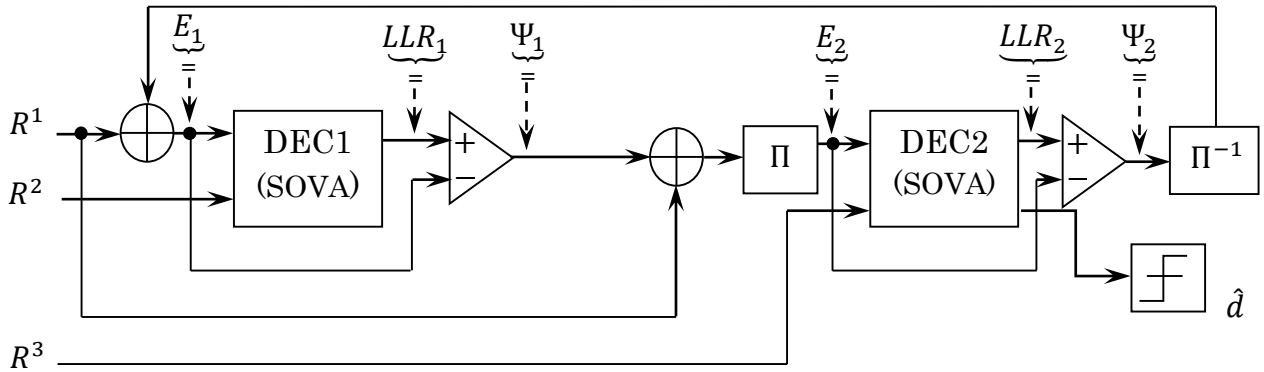


Fig. 2.10 Schéma bloc d'un décodeur itératif associé à un CTC.[11]

2.6 Conclusion

On a présenté avec des exemples détaillés, la théorie des CTC basés sur la concaténation des deux codes convolutifs via un entrelaceur entre les deux. Ces deux codes convolutifs doivent être de type RSC pour une concaténation parallèle.

Le procédé d'entrelacement a été exposé soigneusement en donnant un exemple sur les entrelaceurs en blocs classiques de type ligne-colonne.

Nous avons défini une métrique euclidienne dite métrique de corrélation en basant sur la maximisation de la probabilité de transition du canal AWGGN au sens du ML. L'algorithme de **Viterbi** classique qui utilise cette métrique euclidienne a été exposé avec subtilité. Le survivant qui donne la séquence estimée correspond ici à la métrique maximale, contrairement à ce qui existe dans la grande majorité des littératures.

Nous avons vu que la technique d'appliquer l'algorithme de **Viterbi** classique avec la métrique de corrélation dans les deux sens du treillis, permet de prendre une décision ferme sur le message émis en utilisant le survivant, et de calculer en plus le LLR qui représente une mesure très opportune de la fiabilité

de cette décision. La technique en question est appelée l'algorithme SOVA bidirectionnel ou SOVA tout court.

Pour n'est pas être exhaustif, nous avons essayé de donner l'essentiel dans le développement des formules qui apportent le LLR. A propos de ceci, nous avons exposé avec un grand développement un exemple qui montre les calculs numériques des fiabilités de décision LLR.

Nous avons vu que le SOVA est un algorithme de décodage à ML par séquence ou par chemin sous optimal.

A la fin, nous avons présenté le principe du décodage itératif d'un CTC qui permet aux deux décodeurs constitutifs SOVA de travailler conjointement, c'est à dire que le premier décodeur SOVA profite de l'entrée non systématique R_k^3 du deuxième décodeur SOVA, et que ce dernier avec son rôle profite de l'entrée non systématique R_k^2 du premier décodeur SOVA en utilisant ce que nous appelons l'information extrinsèque.

Pour étudier les performances d'un CTC pour un bruit AWGGN avec l'algorithme SOVA il suffit de passer en simulation numérique. Ce souci sera exposé dans le prochain chapitre qui est la mise en œuvre d'une simulation afin d'évaluer les performances des turbocodes pour un bruit AWGGN.

Chapitre 3

Simulation, résultats et interprétations

3.1 Introduction

La simulation permet l'évaluation des performances des systèmes de communication numériques, et constitue aussi un outil puissant qui aide à l'étude et la conception de tels systèmes. Nous citons quelques raisons montrant l'importance de cet outil

- Quelques niveaux d'analyse ne sont pas faisables avec les outils mathématiques traditionnels, surtout qu'on est en face d'un modèle de canal et d'un système complexe ;
- Il est très facile de passer d'un modèle de simulation d'un système à son implémentation matérielle, et pour cela la distinction entre un modèle de simulation et un prototype d'un système donné devient minimale, d'où l'intérêt économique de la simulation qui réduit considérablement le temps et le coût de réalisation d'un système, et qui se prête plus facilement à la correction et à l'amélioration qu'un prototype.

L'évaluation des performances se fait en calculant la probabilité d'erreur pour un système de communication numérique donné (figure 1.1) et un bruit donné.

3.2 Modèle de simulation

Le modèle utilisé lors de notre simulation est représenté par la figure 3.1. Il comporte :

- Une source d'information binaire ;
- Un turbocodeur convolutif (CTC) parallèle ;
- Un modulateur BPSK ;
- Un canal gaussien généralisé (GG) supposé idéal (sans distorsions) ;
- Une source de bruit AWGGN
- Un décodeur itératif SOVA ;
- Un comparateur pour calculer le nombre de bits erronés après le décodage afin d'estimer la probabilité d'erreurs binaire BER.

Le décodage itératif SOVA est détaillé dans le chapitre précédent.

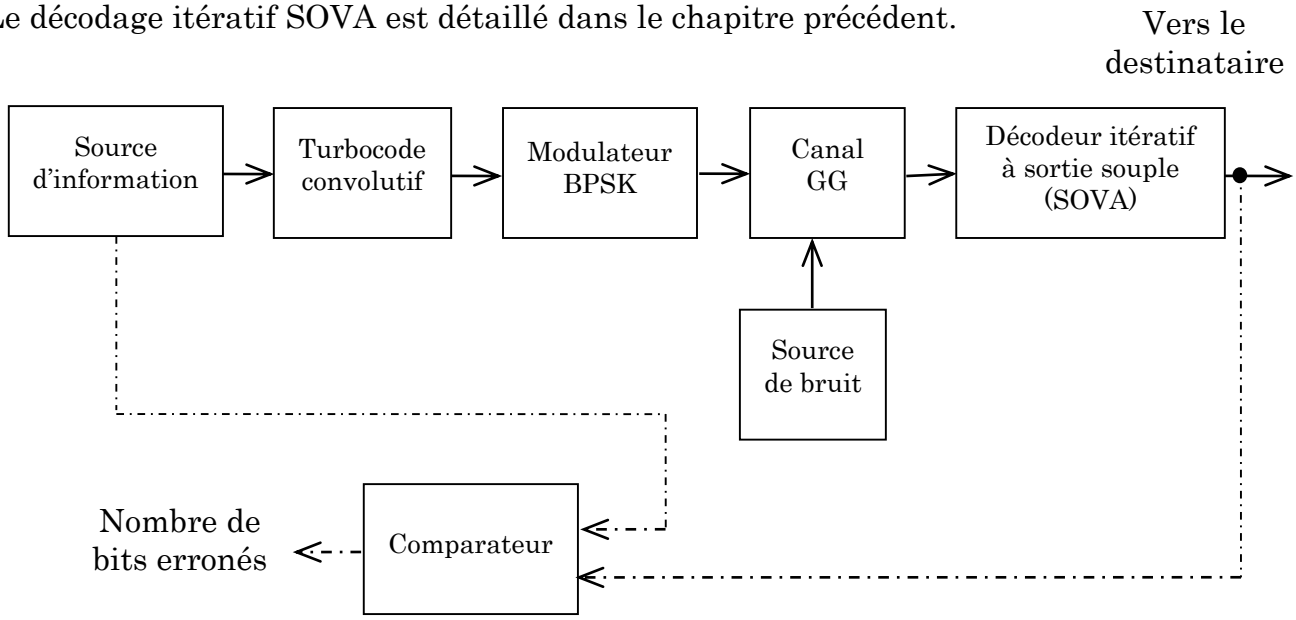


Fig. 3.1 Modèle de simulation.

3.2.1. Source d'information

Pour que l'évaluation soit effectuée dans des conditions proches de ce qui est rencontre en exploitation, on doit choisir une séquence d'information qui simule le mieux possible le trafic réel tout en permettant une mesure simple.

La méthode universellement utilisée dans toutes les simulations, consiste à utiliser une séquence dite pseudo-aléatoire, séquence périodique dont les propriétés statistiques sont «voisines » de celle d'un trafic réel «aléatoire» qui est généralement le résultat d'un brassage. Le générateur d'une telle séquence est constitué d'un registre à décalage comportant m étages, et des additionneurs

modulo-2 (Fig. 3.2). Le choix de m détermine la période de la séquence, qui est égale à $2^m - 1$.

Les connexions aux additionneurs modulo-2 sont déterminées par des polynômes primitifs de degré m ayant la forme [30]

$$h(d) = \sum_{j=0}^{m} h_j d^j \qquad \text{avec} \qquad \begin{cases} h_0 = h_m = 1 \\ h_j = 0 \text{ ou } 1 \end{cases} \text{ pour } j = 1, \ldots, m-1.$$

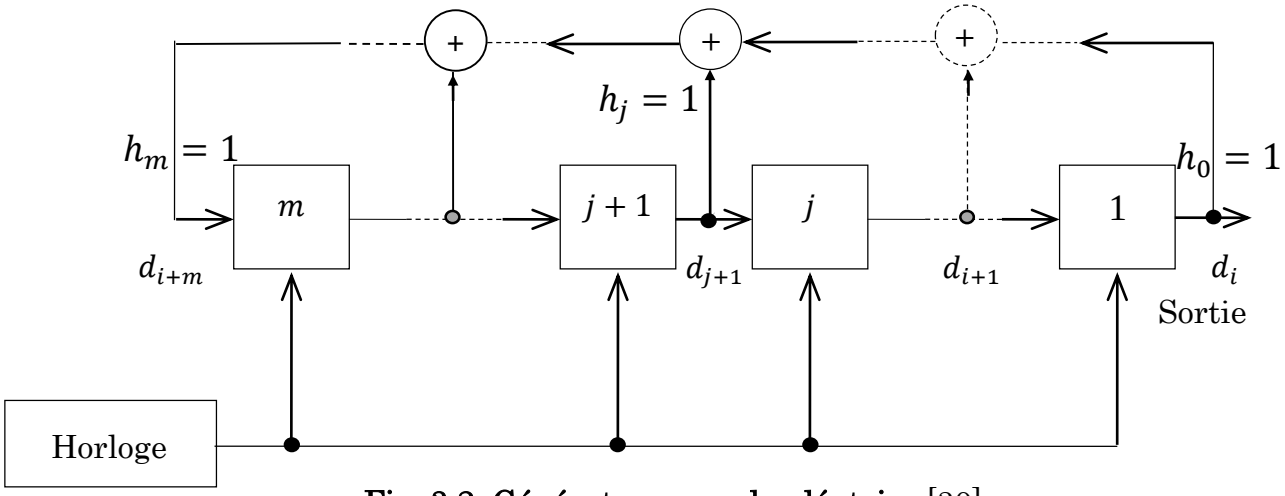


Fig. 3.2 Générateur pseudo-aléatoire [30]

Un polynôme primitif de degré m est un polynôme irréductible qui devise $x^{2m-1} + 1$ et ne devise pas $x^n + 1$ pour $n < 2^m - 1$.

Chaque coefficient h_j égal à 1 correspond à une connexion.

La séquence générée comporte 2^{m-1} éléments égaux à 1 et $2^{m-1} - 1$ à égaux à 0 ce qui donne une répartition presque uniforme des 1 et des 0.

Les principales propriétés de ces séquences sont bien détaillées dans les références [6] et [30].

- **Choix de la longueur de la séquence utilisée**

Le choix de la longueur de la séquence à utiliser dans la simulation, c'est-à-dire la longueur du registre qui l'engendre, est basé sur le fait qu'avec de faibles longueurs de registre (3 ou 4) on risque d'obtenir des résultats différents de la réalité, car une séquence courte ne représente pas un nombre statistiquement suffisant de configurations, donc il est recommandé d'utiliser des séquences relativement longues, dont leurs nombres des registres soient supérieurs à 15 [31]. A cause du facteur de temps et de la mémoire limitée de l'ordinateur utilisé, nous avons choisi $m = 16$ ($2^m - 1 = 65536$).

3.2.2. Turbocode convolutif CTC parallèle

Un turbocode convolutif parallèle est constitué de deux codes convolutifs RSC via un entrelaceur entre les deux.

- **Codes convolutifs utilisés**

On a utilisé dans notre simulation deux codes RSC très employés par **C. Berrou** [2] illustrés dans la figure 3.3. Le RSC de la figure 3.3.a contient quatre états et de rendement 1/2. Son diagramme en treillis est donné dans la figure 3.4.a. Le turbocode parallèle associé à ce RSC est alors à quatre états et de rendement de 1/3 (figure 3.5.a). Le RSC de la figure 3.3.b contient huit états et de rendement 1/2. Son diagramme en treillis est montré dans la figure 3.4.b. Le turbocode parallèle associé à ce RSC est alors à huit états et de rendement 1/3 (figure 3.5.b).

La performance de la modulation BPSK codée avec un RSC par rapport à celle de la BPSK non codée est mise en évidence dans le paragraphe 3.3.2.

- **Entrelaceurs utilisés**

On remémore que le but du procédé d'entrelacement est de permettre l'utilisation des codes destinés à corriger des erreurs indépendantes dans le cas des canaux à paquets d'erreurs.

L'idée maîtresse de ce procédé est de transmettre les symboles d'un mot code entrelacés à des symboles d'autres mots code, de sorte que deux symboles successifs du même mot se trouvent, l'un par rapport à l'autre, à une distance plus grande que la longueur du paquet d'erreurs. De cette sorte, un paquet d'erreurs ne peut affecter plusieurs symboles du mot code et par conséquent les symboles du mot code seront affectés par des paquets différents (donc indépendants) et donc leur effet sera celui produit par les erreurs indépendantes.

Le type le plus simple à utiliser dans les turbocodes est celui d'un entrelaceur en blocs classique ligne-colonne (voir chapitre 2). Dans notre simulation on a utilisé ce type d'entrelaceurs avec une taille moyenne 64×64 pour éviter les grands temps d'exécution.

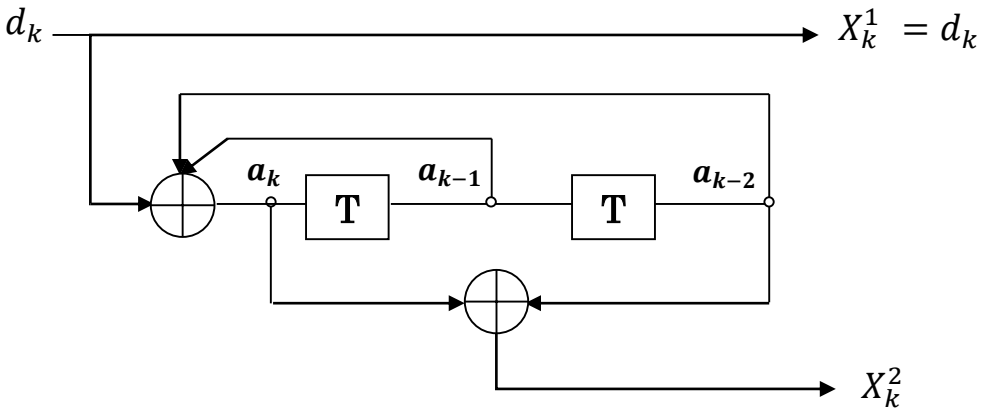


Fig. 3.3.a Code convolutif récursif systématique (5,7)[2]

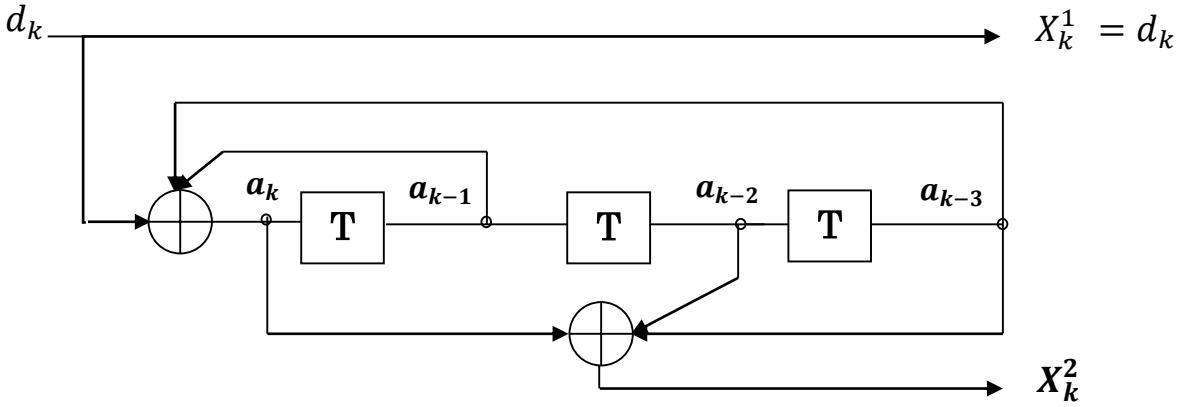


Fig 3.3.b Code convolutif récursif systématique (13,15)[2]

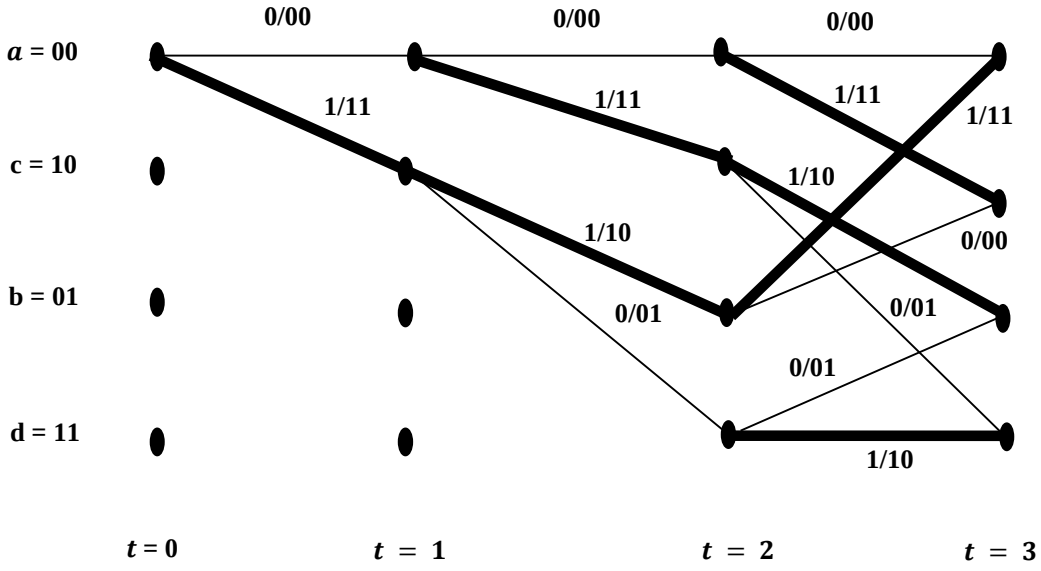


Fig. 3.4.a Représentation en treillis du codeur de la figure 3.3.a.

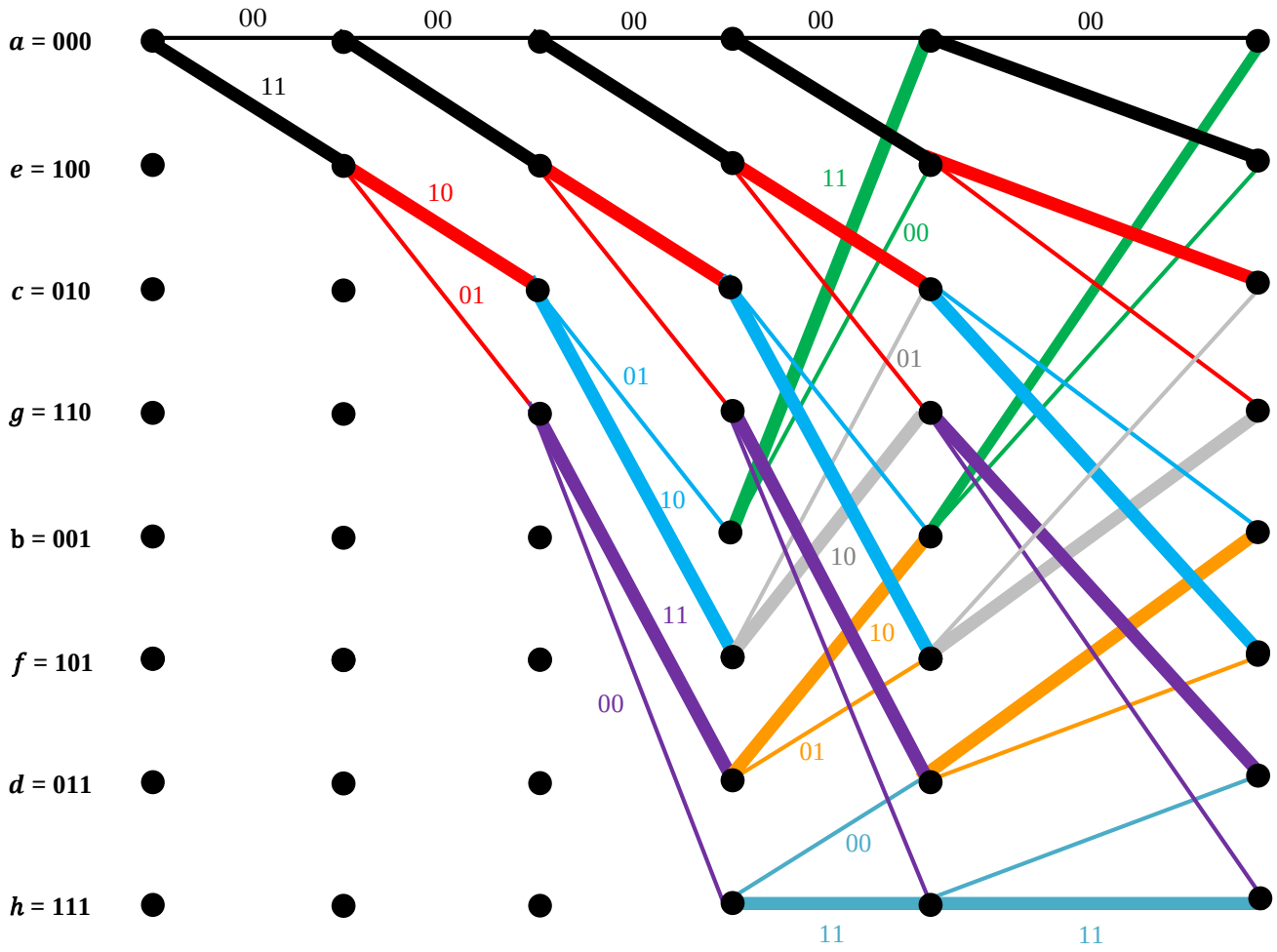


Fig. 3.4.b Représentation en treillis du codeur de la figure 3.3.b.

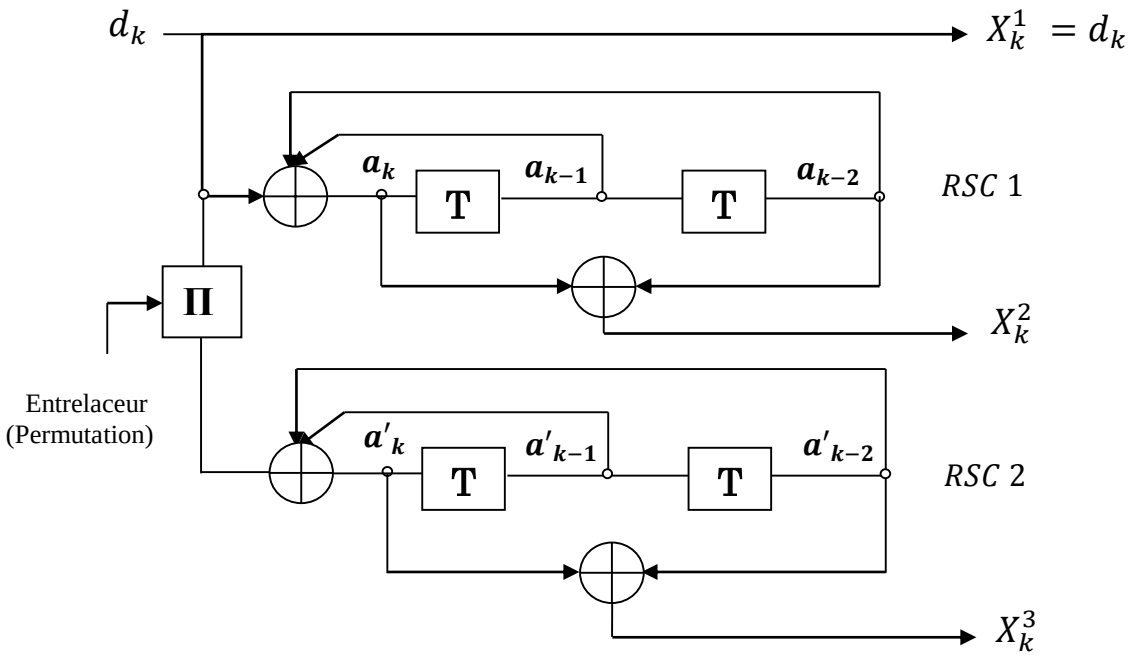


Fig. 3.5.a Turbocodes convolutifs parallèles (5,7)[14]

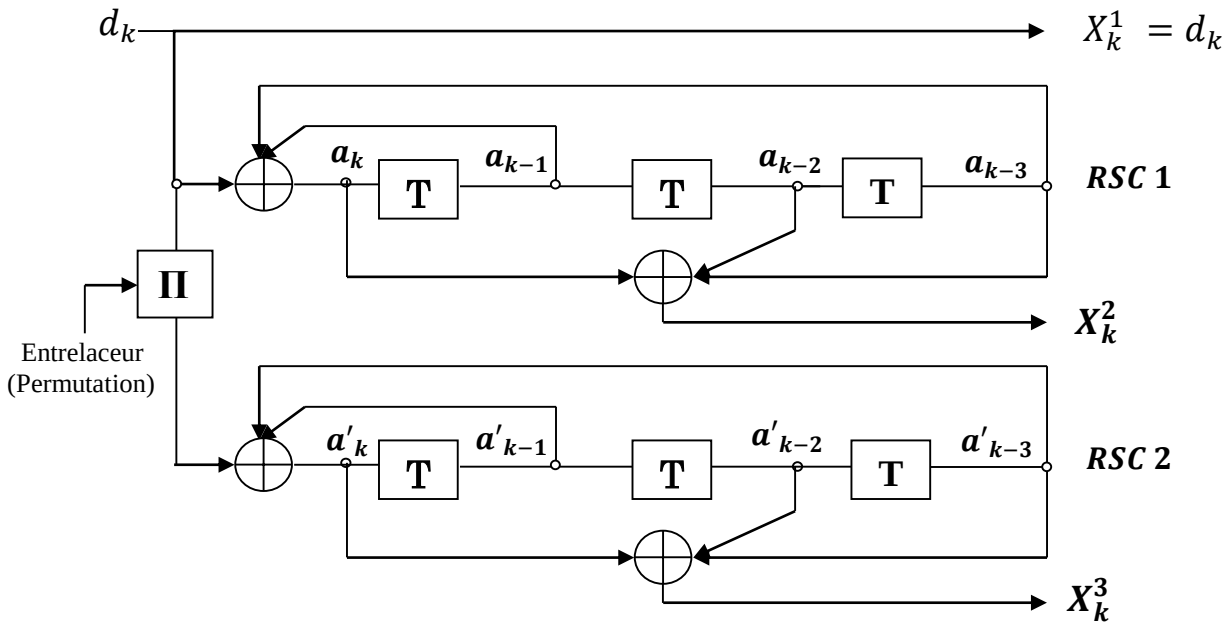


Fig. 3.5.b Turbocodes convolutifs parallèles (13,15)[14]

3.2.3 Modulation utilisée

Pour la modulation utilisée dans la simulation, notre choix reste limité à la modulation BPSK pour tous les turbocodes employés. La modulation des bits de sortie du turbocode est alors (voir chapitre 2)

$$Y(t) = \begin{cases} +\sqrt{E_b} & pour\ un\ bit\ de\ sortie\ X = 0 \\ -\sqrt{E_b} & pour\ un\ bit\ de\ sortie\ X = 1 \end{cases} \qquad (3.1)$$

Pour simplifier les choses, on a pris $\sqrt{E_b} = 1$ (donc le rapport signal à bruit pour chaque bit devient $1/N_0$ où N_0 représente la densité de puissance mono latérale du bruit. La modulation devient tout simplement

$$Y = \begin{cases} +1\ pour\ un\ bit\ X = 0 \\ -1\ pour\ un\ bit\ X = 1 \end{cases} \quad \text{ou} \quad S = 1 - 2X \qquad (3.2)$$

3.2.4. Source de bruit

Le bruit envisagé est

- blanc ;
- additif ;
- à distribution gaussienne généralisée avec un paramètre de forme $\alpha = 0.5$.

On n'a pas pris en compte de l'effet du filtre de réception, c'est-à-dire que le bruit reste comme il est à la réception (car le filtrage rend ce bruit coloré).

Soient p la probabilité à calculer et $\hat{p}$ la valeur estimée par la simulation. Dans la méthode de Monté-Carlo, Si on désire connaître la précision relative r pour un nombre d'essais N, et une confiance $1 - c$, on applique la formule [30]

$$r^2 \geq 2\left(erfc^{-1}(c)\right)^2 \frac{1}{N}\left(\frac{1}{p} - 1\right) \tag{3.3}$$

L'inconvénient majeur de cette méthode réside dans le fait que le nombre d'essais N, croît rapidement lorsque les probabilités d'erreur sont faibles. Malheureusement, on n'a pas pu calculer les précisions r dans notre simulation, à cause du facteur de temps !

3.2.4.1. La distribution gaussienne généralisée GGD

La fonction de densité de probabilité (PDF) d'une variable gaussienne généralisée aléatoire X, avec moyenne μ et variance σ^2, est définie comme

$$\begin{cases} f(x) = \dfrac{\alpha}{2 \cdot A \cdot \Gamma\left(\frac{1}{\alpha}\right)} exp\left\{-\left|\dfrac{x-\mu}{A}\right|^{\alpha}\right\} & x \in \mathbb{R}, \\ \text{où} & \\ A = \sigma\left[\dfrac{\Gamma(1/\alpha)}{\Gamma(3/\alpha)}\right]^{1/2} \quad et \quad \Gamma(z) = \displaystyle\int_0^{+\infty} t^{z-1}e^{-t}dt\,, & z > 0 \end{cases} \tag{3.4}$$

(Γ est la fonction gamma).

La distribution gaussienne généralisée (GGD) est symétrique par rapport à μ. A est un facteur d'échelle qui définit la dispersion de la distribution, d'où il est une mesure généralisée de la variance. $\alpha > 0$ est le paramètre de forme qui décrit le taux exponentiel de la décroissance : queues lourdes correspondent à des valeurs plus petites de α.

Il faut noter que α est appelé aussi l'exposant de la loi GG ou le coefficient de gaussianité. Si $\alpha < 2$, nous avons une loi sous-gaussienne et si $\alpha > 2$ une loi sur-gaussienne [32].

La famille du gaussien généralisé comprend une variété de variables aléatoires. Certaines classes bien connues de distributions sont générées par un paramétrage de la décroissance exponentielle de la GGD. Lorsque $\alpha = 1$, la GGD correspond à une distribution de Laplacien (ou double exponentielle). Pour $\alpha = 2$, on a distribution gaussienne. Quand $\alpha \to +\infty$ le GGD converge vers une distribution uniforme dans $[\mu - \sqrt{3}\sigma, \mu + \sqrt{3}\sigma]$, tandis que lorsque $\alpha \to 0^+$ nous avons une fonction de probabilité impulsive à $x = \mu$.

Tous les moments centraux impairs de la GGD définie avec (3.4) sont égaux à zéro, $E(|X-\mu|^r) = 0 \quad (r = 1,3,5,\dots)$, et les moments centraux pairs sont

$$E((X-\mu)^r) = \left[\frac{\sigma^2\Gamma(1/\alpha)}{\Gamma(3/\alpha)}\right]^{r/2} \frac{\sigma^2\Gamma((r+1)/\alpha)}{\Gamma(1/\alpha)} \qquad r = 2,4,6,\dots \tag{3.5}$$

On peut montrer que la relation (3.5) reste valable pour tout nombre réel $r > -1$ différent des entiers impairs [32].

Avec une normalisation simple et certaines réductions de (3.4), on obtient une variable aléatoire GG de moyenne nulle et de variance unité ayant comme densité la fonction suivante

$$f(x) = \frac{\alpha}{2 \cdot A \cdot \Gamma(1/\alpha)} \exp\{-|x/A|^\alpha\} \quad x \in \mathbb{R}, \qquad \text{où} \quad A = \left[\frac{\Gamma(1/\alpha)}{\Gamma(3/\alpha)}\right]^{1/2} \tag{3.6}$$

Le kurtosis K_x d'une variable aléatoire X est le rapport entre le moment d'ordre 4 et le moment d'ordre 2 à la puissance 2 de la variable aléatoire X [32]

$$K_x = \frac{E(X^4)}{\left(E(X^2)\right)^2}$$

Le kurtosis réalise une comparaison de la PDF de la variable aléatoire X par rapport à la loi de gauss de même variance qu'il possède un kurtosis égal à 3

- Si la densité de probabilité de X tend vers 0 à l'infini plus rapidement que la loi gaussienne, la variable aléatoire est dite sous-gaussienne et le kurtosis est supérieur à 3 ;
- Si la densité de probabilité de X tend vers 0 à l'infini moins vite que la loi gaussienne, la variable aléatoire est dite sur-gaussienne et le kurtosis est à inférieur 3 ;
- Si la densité de probabilité de X est gaussienne, le kurtosis est égal à 3.

Le kurtosis caractérise l'aplatissement de la PDF par rapport à celui d'une distribution gaussienne.

Le kurtosis de la distribution (3.6) est

$$K_x(\alpha) = \frac{\Gamma(1/\alpha)\Gamma(5/\alpha)}{[\Gamma(3/\alpha)]^2} \tag{3.7}$$

$K_x(\alpha)$ diminue avec α, et on peut constater que :

$$\lim_{\alpha\to 0^+} K_x(\alpha) = +\infty \qquad \lim_{\alpha\to+\infty} K_x(\alpha) = 1.8 \tag{3.8}$$

La figure 3.6 montre la densité d'une GGD, à moyenne nulle et de variance unité, pour différentes valeurs du paramètre α. La figure 3.7 montre le comportement du kurtosis pour $\alpha \in [0, 2]$ et $\alpha \in [0, 20]$.

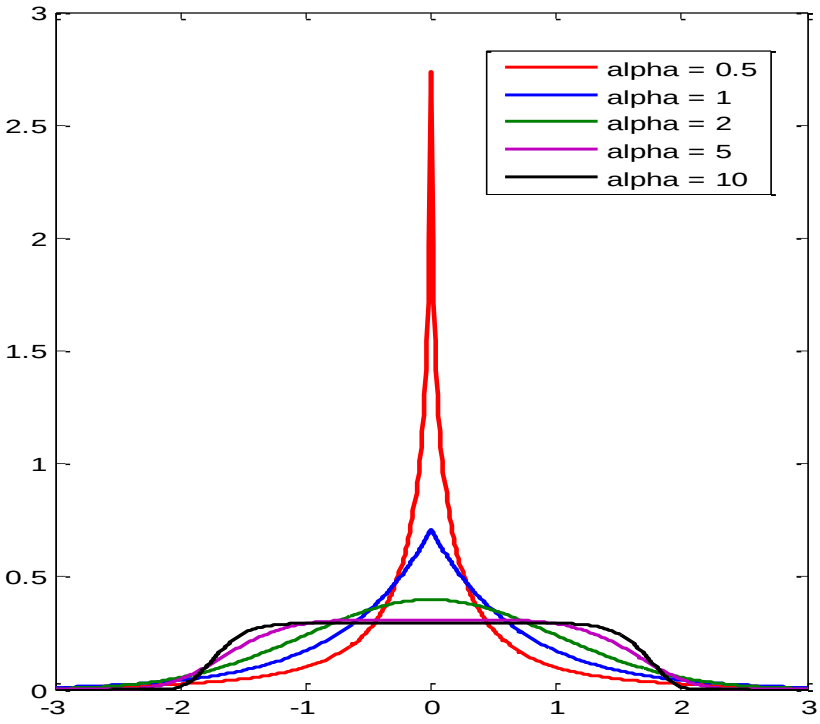


Fig. 3.6 La densité gaussienne généralisée pour différentes valeurs du paramètre α, avec $\mu = 0$ et $\sigma = 1$.

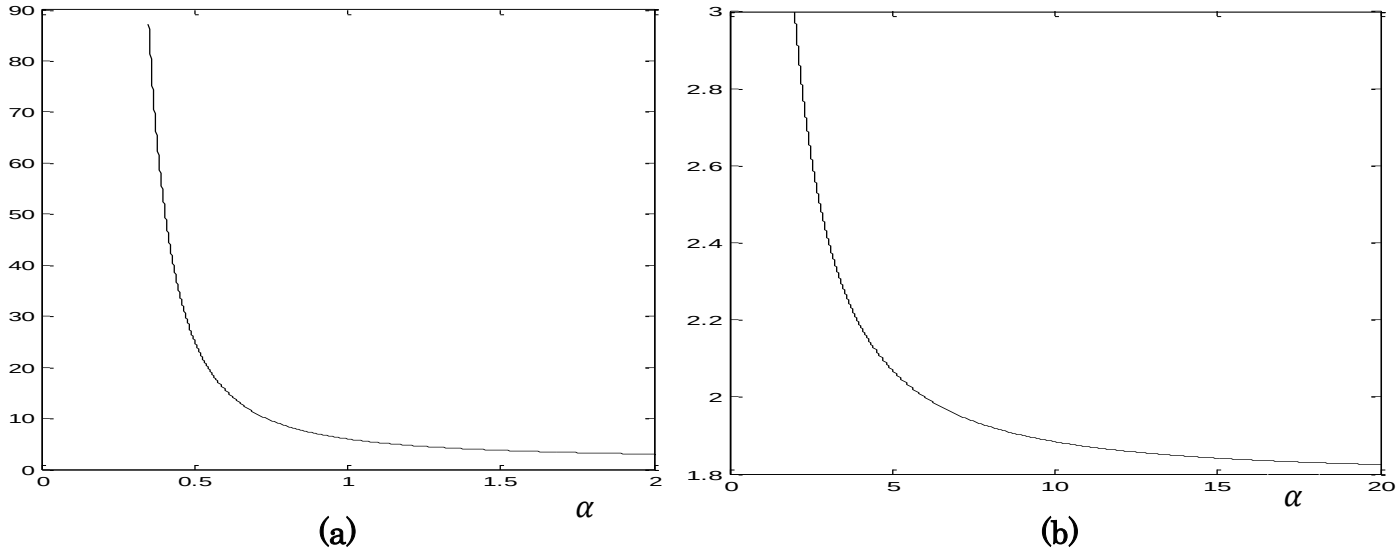


Fig. 3.7 Le Kurtosis de la GGD avec variance unité et α variable : (a) dans $[0, 2]$ (b) dans $[0, 20]$.

3.2.4.2. Simulation d'un bruit AWGGN

Dans ce qui suit, nous limitons notre attention à des variables aléatoires gaussiennes généralisées dont la densité est donnée dans (3.6).

Soient F la fonction de répartition d'une variable aléatoire X et F^{-1} son inverse. Il est bien connu que si F^{-1} peut être directement évaluée, un grand nombre de réalisations de X peuvent être obtenues par $x_i = F^{-1}(u_i)$, où u_i $(i = 1, 2, \ldots, n)$ sont de nombres aléatoires distribués uniformément dans $[0, 1]$. Si F^{-1} a une expression analytique, une telle méthode peut être appliquée de manière efficace, mais malheureusement, ce n'est pas toujours le cas. Néanmoins, si F^{-1} peut être évalué, il peut toujours être possible de simuler la variable aléatoire X par inversion numérique de sa fonction de répartition. Lorsque F^{-1} n'a pas d'expression de forme ferme, l'intégration numérique ou d'autres méthodes d'approximation sont nécessaires, au détriment d'une quantité croissante de calcul. Une autre méthode de simulation basée sur la transformation de la variable aléatoire X dont laquelle un générateur de nombres aléatoires est disponible. Une technique spécifique sera décrite ci-dessous pour le cas GGD.

Soit X une variable aléatoire gaussienne généralisée avec la fonction de répartition

$$F(x) = \int_{-\infty}^{x} \frac{\alpha}{2 \cdot A \cdot \Gamma(1/\alpha)} \exp\{-|x/A|^{\alpha}\}\, dt \tag{3.9}$$

où A a été défini ci-dessus. Une telle fonction peut être écrite sous la forme ferme seulement pour quelques cas spéciaux.

Afin de générer de valeurs d'une distribution gaussienne généralisée avec paramètre α, on peut utiliser les trois étapes suivantes

- simuler une variable aléatoire gamma $Z \sim Gamma\,(a, b)$, avec les paramètres $a = \alpha^{-1}$ et $b = A^{-\alpha}$;
- appliquer d'abord la transformation $Y = Z^{1/\alpha}$;
- finalement, appliquer une transformation de la forme

$$Y = |X| \tag{3.10}$$

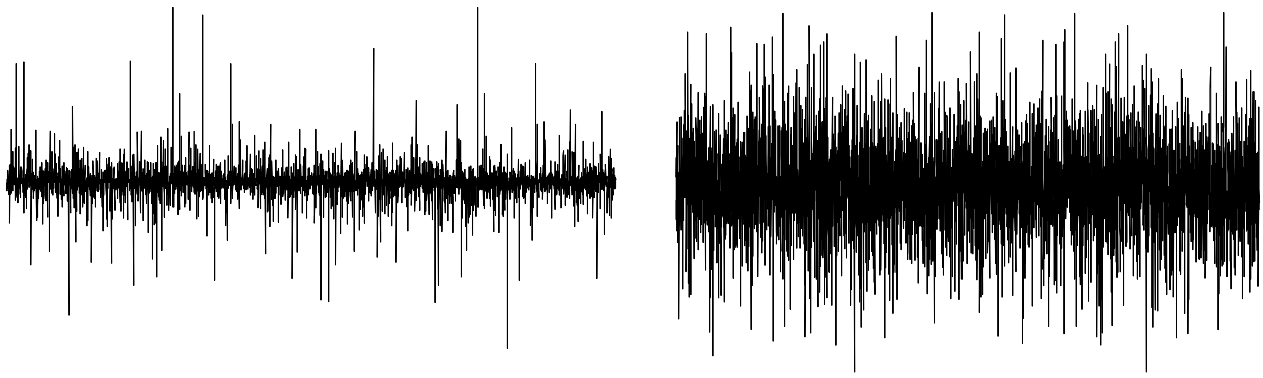
La relation (3.10) présente deux racines. Le problème est de savoir comment déterminer la probabilité de choisir chaque racine. On peut démontrer, en utilisant la symétrie de la GGD, qu'on peut prend les racines avec des probabilités égales. Pour chaque observation y aléatoire, une racine est choisi ($x = -y$ ou $x = y$). À cette fin, un essai auxiliaire de **Bernoulli** avec une probabilité $p = 1/2$ peut être effectuée.

Avec cette procédure on peut générer des échantillons i.i.d d'un bruit AWGGN

- de variance égale à l'unité ;
- de moyenne nulle.

Pour avoir des échantillons de variance σ^2, on multiplie les échantillons déjà générés du bruit par σ^2.

La figure 3.8 montre le tracé de 4096 échantillons d'un bruit AWGGN générés par la procédure précédente pour deux valeurs de α. Cette figure montre que plus α est proche de 0 plus le bruit est impulsif.



(a) (b)

Fig. 3.8 Le tracé de 4096 échantillons d'un AWGGN avec variance unité, moyenne nulle et α variable : (a) $\alpha = 0.5$ (b) $\alpha = 1$.

Pour n'est être exhaustif, dans notre simulation on se limite au bruit AWGGN où $\alpha = 0.5$, et pour avoir l'effet du paramètre de forme α sur le décodage itératif on peut consulter [33].

3.2.5 Décodeur itératif ML à sortie souple (SOVA)

Dans ce paragraphe on va définir les terminologies suivantes : treillis fermé direct et treillis fermé indirect ou inverse.

Définition 1 : on dit qu'un treillis fermé est direct si dans son ouverture les branches qui lient les états entre eux ne chevauchent pas. Par exemple le treillis de la figure 3.4.a.

Définition 2 : le treillis fermé indirect ou inverse est un treillis direct inversé en considérant l'étape finale comme était l'étape initiale.

Comme il est déjà expliqué dans le chapitre 2, le SOVA nécessite les calculs des métriques de corrélation dans les deux sens du treillis. A ce niveau une nouvelle idée est utilisée dans la programmation du SOVA qui facilite amplement les calculs des métriques. Cette idée est basée sur la remarque qu'un treillis inverse, dans les faits, est un treillis direct en arrangeant les états comme suit « on prend les états du treillis inverse, qui appartiennent aux branches, écrits en binaire, puis on fait l'inversion des bits, on obtient les états du treillis direct ». Par exemple dans le treillis inverse de la figure 3.4.b. si on prend le chemin $S_0^{(0)}S_1^{(4)}S_2^{(6)}S_3^{(3)}S_4^{(1)}S_5^{(0)}$, alors le chemin direct correspond à ce dernier est $S_0^{(inv(0))}S_1^{(inv(4))}S_2^{(inv(6))}S_3^{(inv(3))}S_4^{(inv(1))}S_5^{(inv(0))}$ où $inv(n)$ représente l'inversion des bits du nombre n écrit en binaire. Donc, $inv(0) = 0$, $inv(4) = 1$, $inv(6) = 3$, $inv(3) = 6$ et $inv(1) = 4$ et le chemin direct s'écrit tout simplement $S_0^{(0)}S_1^{(1)}S_2^{(3)}S_3^{(6)}S_4^{(4)}S_5^{(0)}$.

Le décodeur itératif de la figure 2.10 dans le chapitre 2 converge lentement pour les faibles rapports signal à bruit. Pour remédier à cela, on propose une idée simple, inspirée des méthodes numériques itératives, consiste à multiplier l'information extrinsèque calculée de chaque décodeur constitutif par un facteur β, dit **facteur de convergence,** avant de la réinjectée dans l'autre décodeur constitutif (voir la figure 3.9). L'utilisation de ce facteur diminue le nombre des itérations presque de la moitié. Le choix de ce facteur est fait par le tâtonnement. Pour le CTC de la figure 3.5.a. le meilleur facteur de convergence est $\beta \cong 0.1$, et pour celui de la figure 3.5.b. est $\beta \cong 0.13$. Une étude comparative entre ces résultats et ceux de **C. Berrou** [2], qui corrobore la convenance de ce facteur est bien présentée dans [11].

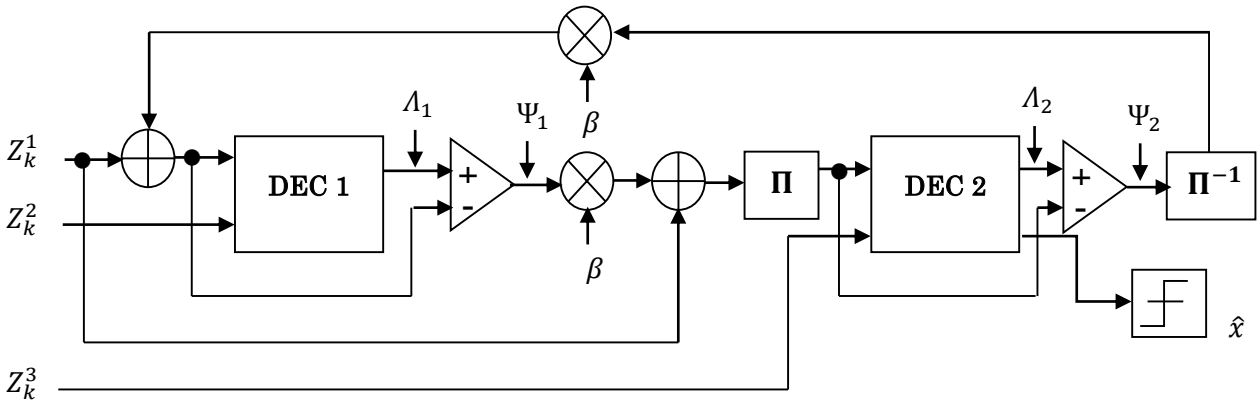


Fig. 3.9 Schéma bloc d'un décodeur itératif avec facteur de convergence[11]

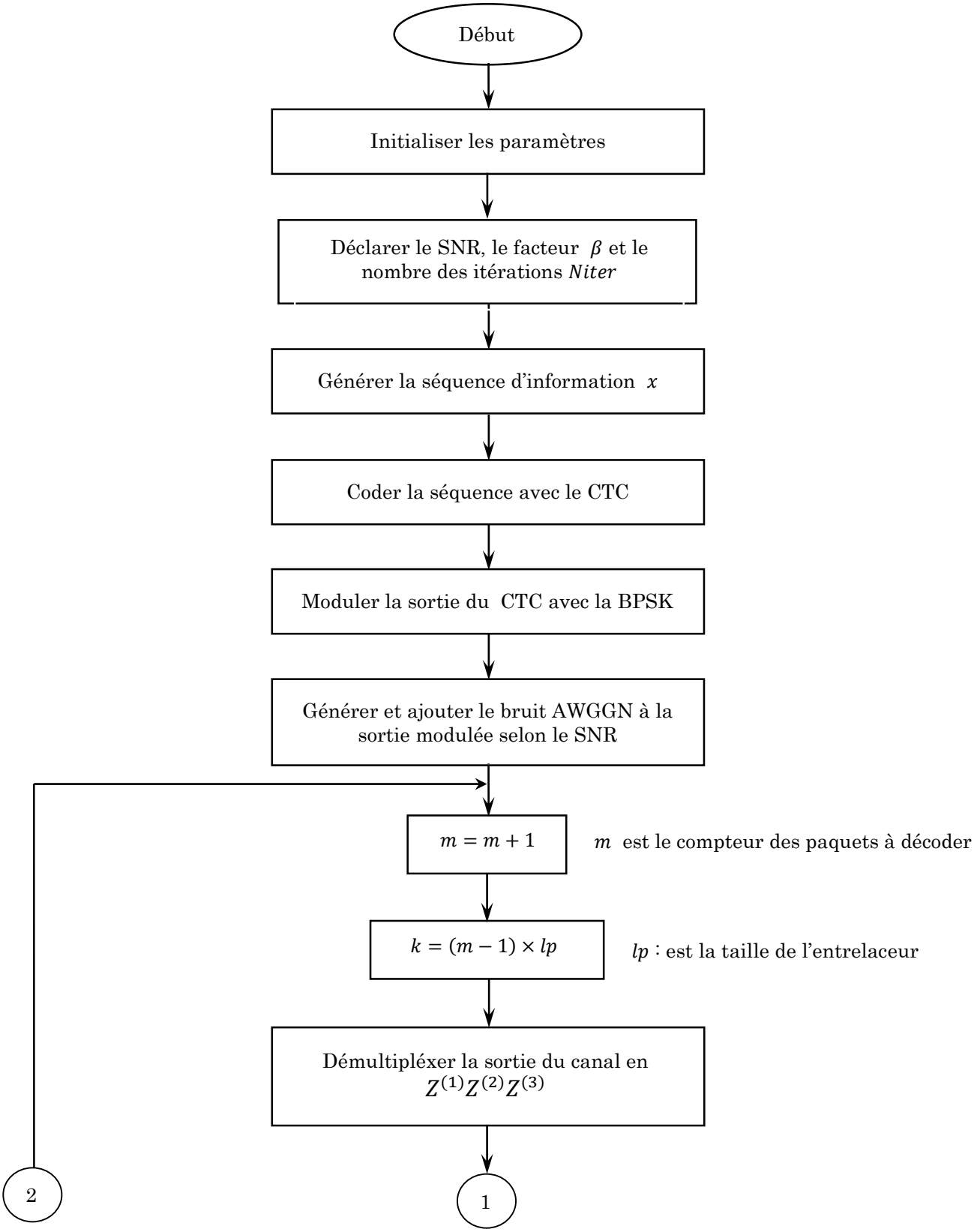
3.2.6 Calcul du taux d'erreurs binaire BER

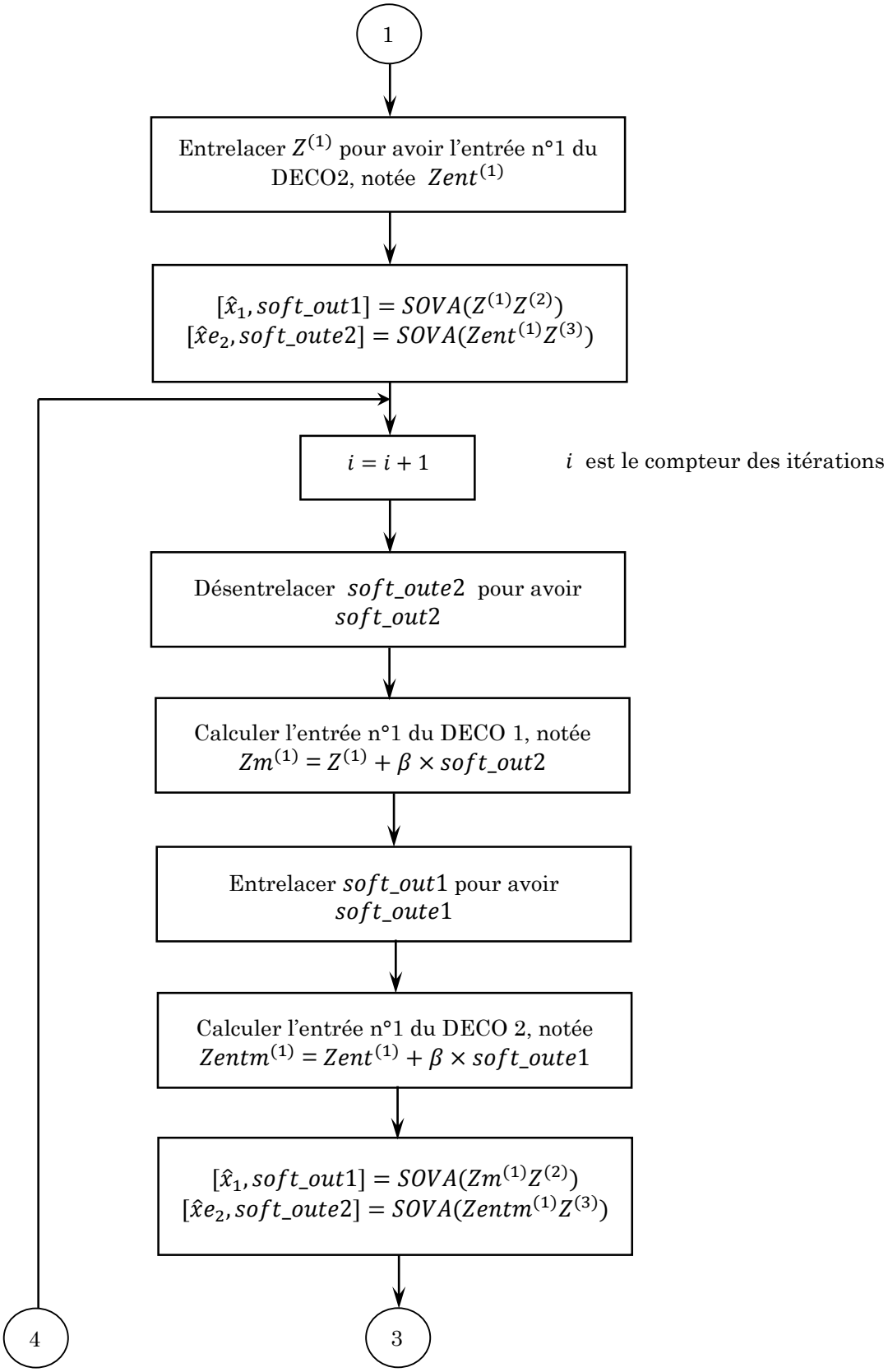
La séquence d'information pseudo-aléatoire générée par la source est de longueur $2^{16} = 65536$ bits, qui est un multiple de la taille de l'entrelaceur 64×64.

Le décodeur itératif décode la séquence reçue par paquets. Chaque paquet est de taille égale à celle de l'entrelaceur. Donc, dans notre cas on a $65536/(64 \times 64) = 16$ paquets.

Pour un SNR et un nombre d'itérations donnés, le décodeur itératif compte le nombre des bits erronés noté $nber$ par le truchement d'une comparaison entre la vraie séquence x et la séquence estimée $\hat{x}$ délivrée par l'un des décodeurs constitutifs (le deuxième décodeur par exemple). Le taux d'erreurs binaire BER est le rapport entre ce nombre des bits erronés et la longueur totale de la séquence notée $lseq$, $BER = nber/lseq$.

La programmation des décodeurs itératifs utilisés dans notre simulation est basée sur l'organigramme de la figure 3.10. Pour la programmation du SOVA, employé par les deux décodeurs constitutifs, on se servir de l'organigramme de la figure 3.11.


Début
Initialiser les paramètres
Déclarer le SNR, le facteur β et le nombre des itérations $Niter$
Générer la séquence d'information x
Coder la séquence avec le CTC
Moduler la sortie du CTC avec la BPSK
Générer et ajouter le bruit AWGGN à la sortie modulée selon le SNR
$m = m + 1$
m est le compteur des paquets à décoder
$k = (m - 1) \times lp$
lp : est la taille de l'entrelaceur
Démultiplexer la sortie du canal en $Z^{(1)}Z^{(2)}Z^{(3)}$
2
1


1
Entrelacer $Z^{(1)}$ pour avoir l'entrée n°1 du DECO2, notée $Zent^{(1)}$
$[\hat{x}_1, soft\_out1] = SOVA(Z^{(1)}Z^{(2)})$
$[\hat{x}e_2, soft\_oute2] = SOVA(Zent^{(1)}Z^{(3)})$
$i = i + 1$
i est le compteur des itérations
Désentrelacer $soft\_oute2$ pour avoir $soft\_out2$
Calculer l'entrée n°1 du DECO 1, notée $Zm^{(1)} = Z^{(1)} + \beta \times soft\_out2$
Entrelacer $soft\_out1$ pour avoir $soft\_oute1$
Calculer l'entrée n°1 du DECO 2, notée $Zentm^{(1)} = Zent^{(1)} + \beta \times soft\_oute1$
$[\hat{x}_1, soft\_out1] = SOVA(Zm^{(1)}Z^{(2)})$
$[\hat{x}e_2, soft\_oute2] = SOVA(Zentm^{(1)}Z^{(3)})$
4
3

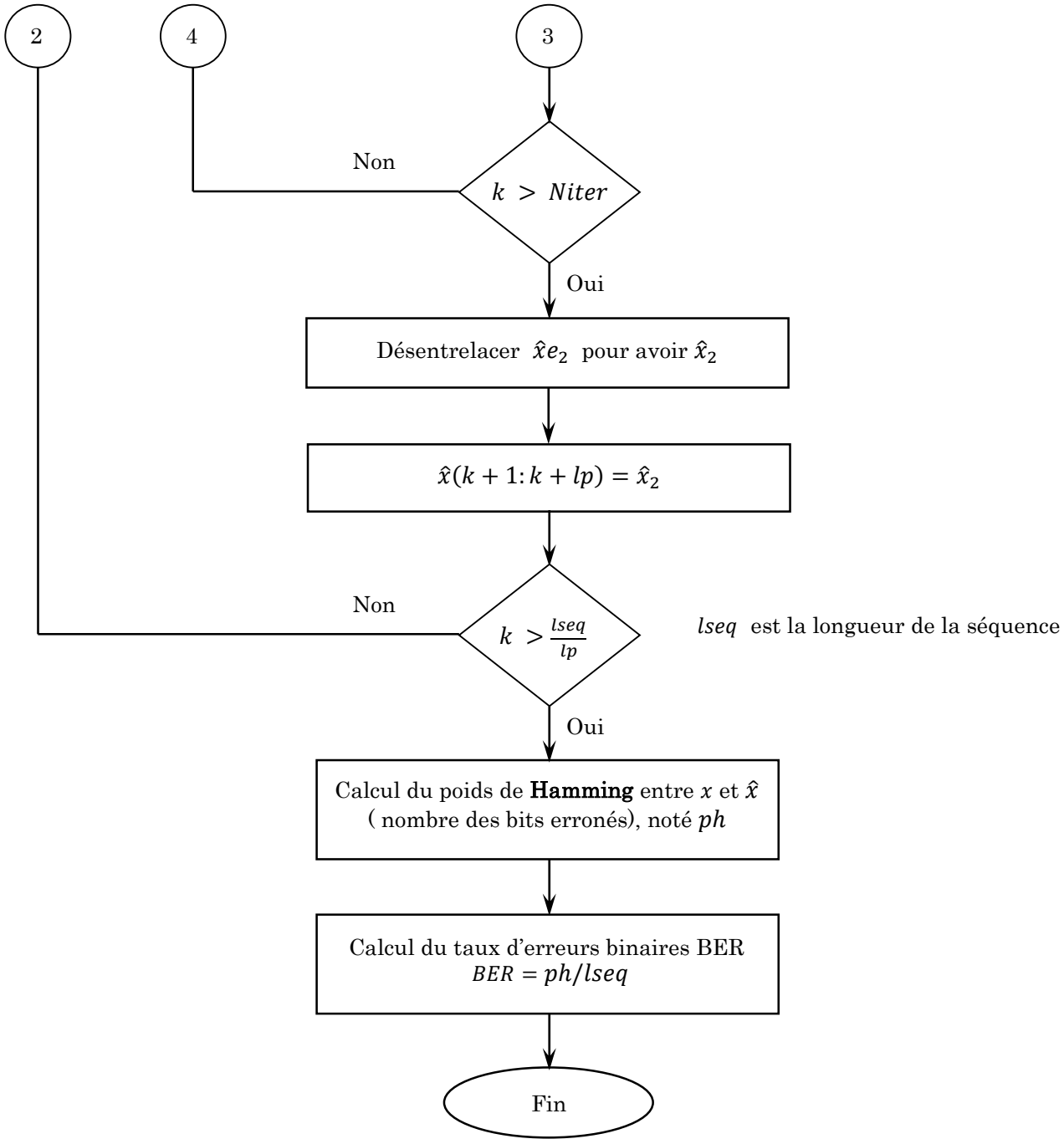
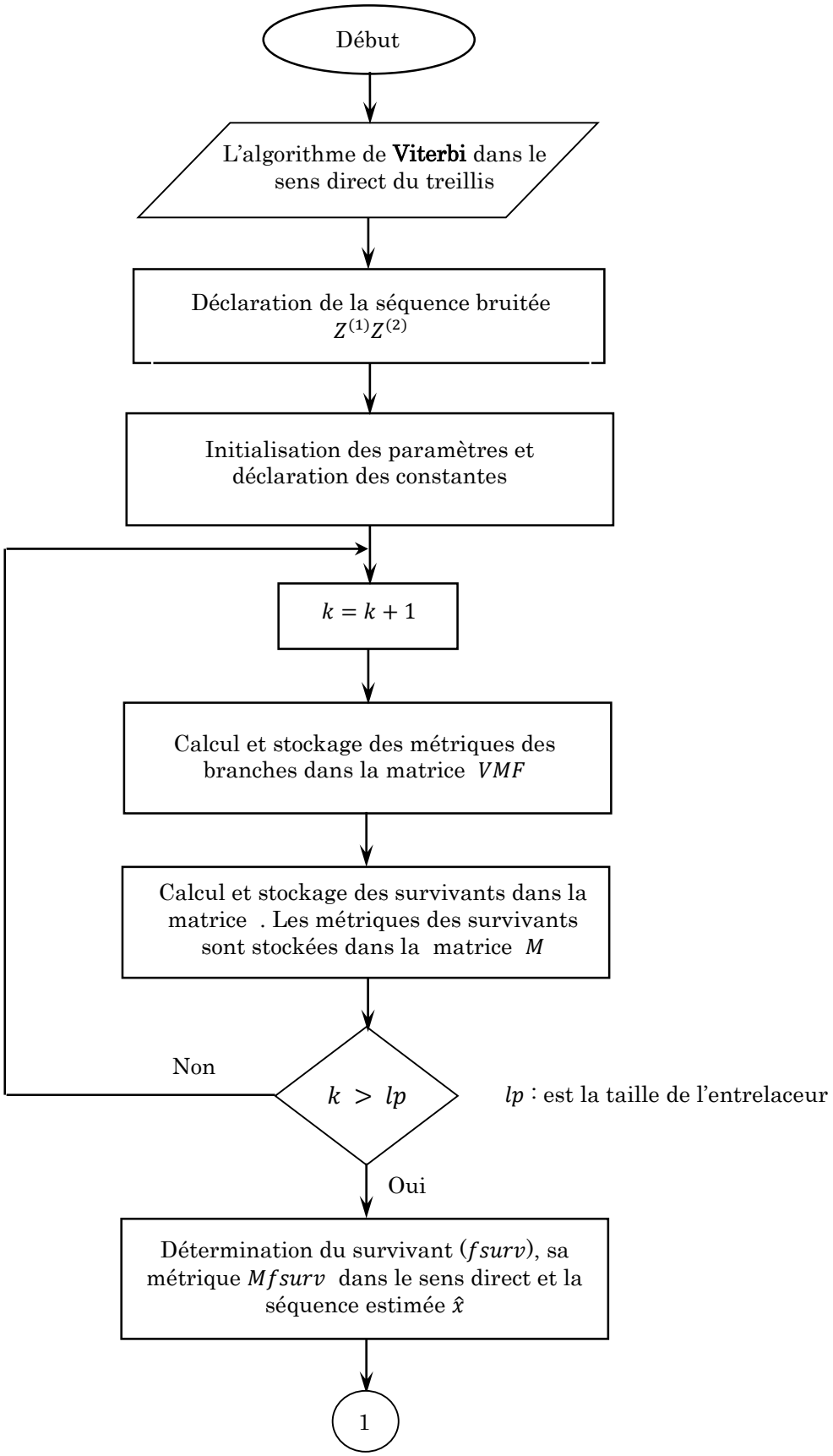
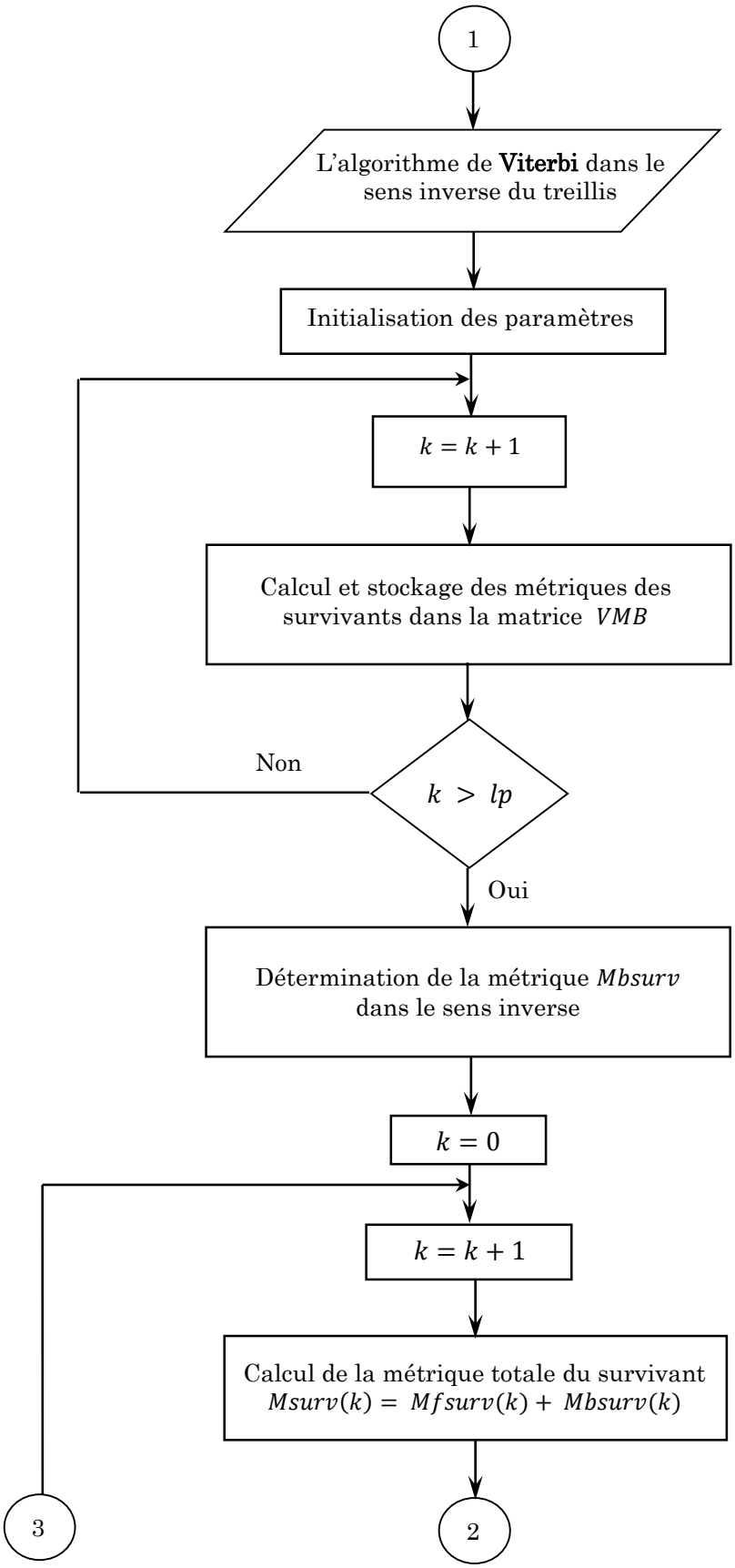


Fig. 3.10 Organigramme représentant le plan de simulation


Début
L'algorithme de **Viterbi** dans le sens direct du treillis
Déclaration de la séquence bruitée $Z^{(1)}Z^{(2)}$
Initialisation des paramètres et déclaration des constantes
$k = k + 1$
Calcul et stockage des métriques des branches dans la matrice VMF
Calcul et stockage des survivants dans la matrice . Les métriques des survivants sont stockées dans la matrice M
Non
$k > lp$
lp : est la taille de l'entrelaceur
Oui
Détermination du survivant ($fsurv$), sa métrique $Mfsurv$ dans le sens direct et la séquence estimée $\hat{x}$
1


1
L'algorithme de **Viterbi** dans le sens inverse du treillis
Initialisation des paramètres
$k = k + 1$
Calcul et stockage des métriques des survivants dans la matrice VMB
Non
$k > lp$
Oui
Détermination de la métrique $Mbsurv$ dans le sens inverse
$k = 0$
$k = k + 1$
Calcul de la métrique totale du survivant $Msurv(k) = Mfsurv(k) + Mbsurv(k)$
3
2

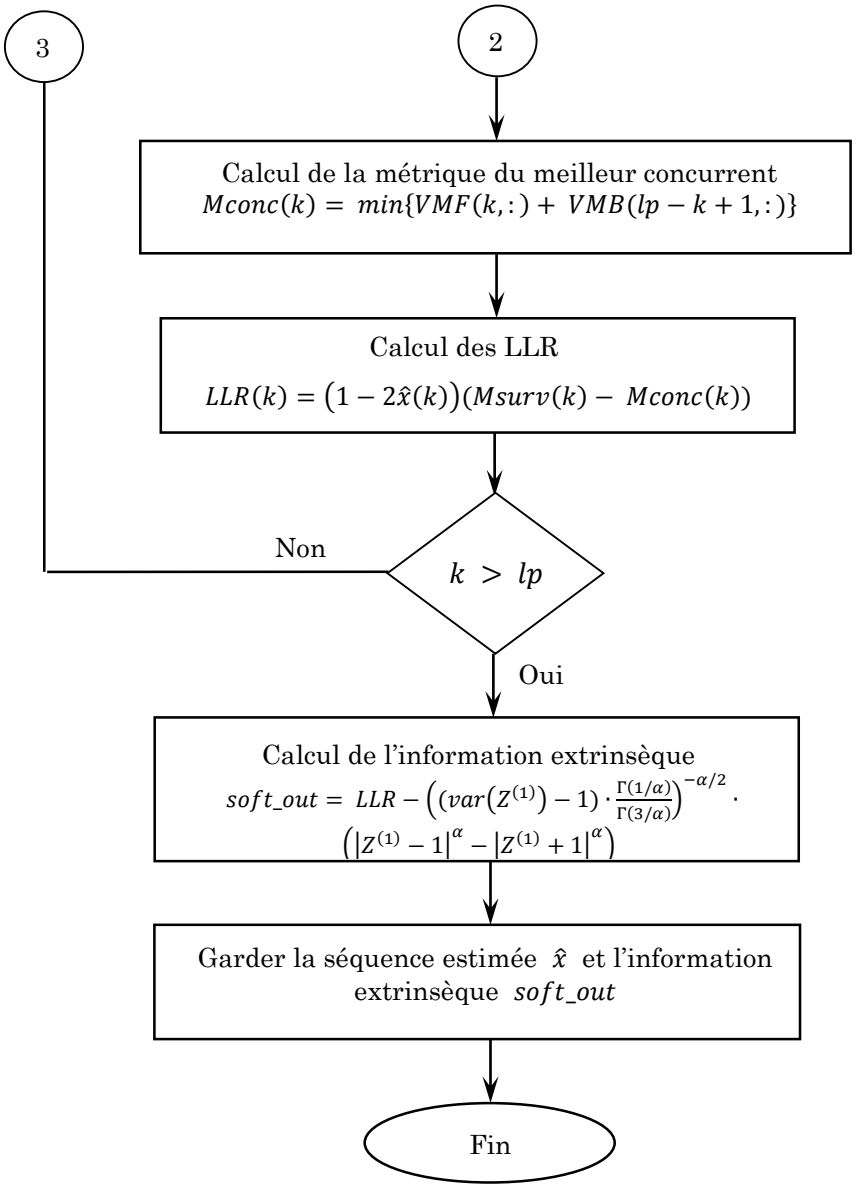


Fig. 3.11 Organigramme de l'algorithme SOVA

3.3. Résultats et Commentaires

3.3.1. Programmation

Pour la mise en œuvre de notre simulation, nous avons opté pour la réalisation des programmes le **Matlab** pour les raisons suivantes

- Le **Matlab** contient des fonctions prêtes à utiliser, telles que la génération du bruit, la fonction d'erreur, fonction d'autocorrection, etc. Ces fonctions permettent une réduction des programmes.

- Le **Matlab** contient un graphisme qui permet une visualisation et un traçage simple des courbes.

Nous avons réalisé donc, à l'aide du **Matlab** des programmes de simulation pour les deux CTC parallèles de la figure 3.5. avec un entrelaceur 64×64 de type ligne colonne.

Nous signalons que la partie décodage par l'utilisation de l'algorithme de **Viterbi** à sortie souple SOVA était la partie la plus difficile à mettre en œuvre du fait de sa complexité.

3.3.2. Taux d'erreurs binaire BER

Le taux d'erreurs binaire BER mesure la performance d'un système de communication numérique. Notre simulation a permis de calculer ce paramètre pour l'ensemble des systèmes étudiés et pour différentes valeurs de bruit.

La figure 3.13 montre les BER pour deux systèmes, l'un est sans codage et l'autre avec codage convolutif, en fonction du rapport signal à bruit SNR

- Pour le système BPSK sans codage son BER est bien approximée par la relation (1.9) (voir chapitre 1) et représentée par la courbe supérieure.
- Pour le système BPSK codé avec le RSC de la figure 3.3.a, son BER est représenté par la courbe inférieure.

L'analyse des figures 3.13 jusqu'à 3.17 permet de faire les constatations et les remarques suivantes

1) un système avec codage est plus performant que celui sans codage.
2) Le BER est inversement proportionnelle au SNR, ce qui confirme davantage la théorie, puisque pour un SNR grand la séquence émise n'est affectée que peu par le bruit. Donc, le décodeur de vraisemblance maximale aura à son entrée une séquence avec un nombre faible de bits erronés, d'où une probabilité faible, et vice versa.

3.3.3 Performance en fonction de l'entrelaceur

La figure 3.14 montre les BER des deux systèmes à modulation BPSK qui utilise le même code RSC de la figure 3.3.a, l'un est sans entrelaceur et l'autre

avec entrelaceur (voir la figure 3.12). On a supposé ici que le canal de transmission est subit à un bruit par paquets (en salves).

L'analyse de la figure 3.13 permet de conclure qu'un système qui utilise un entrelaceur est plus performant que celui sans entrelaceur, ce qui corrobore encore la théorie des entrelaceurs exposée dans le chapitre 3, puisque l'entrelaceur permet d'espacer les symboles de bruit consécutifs, ce qui facilite au décodeur à algorithme de **Viterbi** de corriger les erreurs facilement.

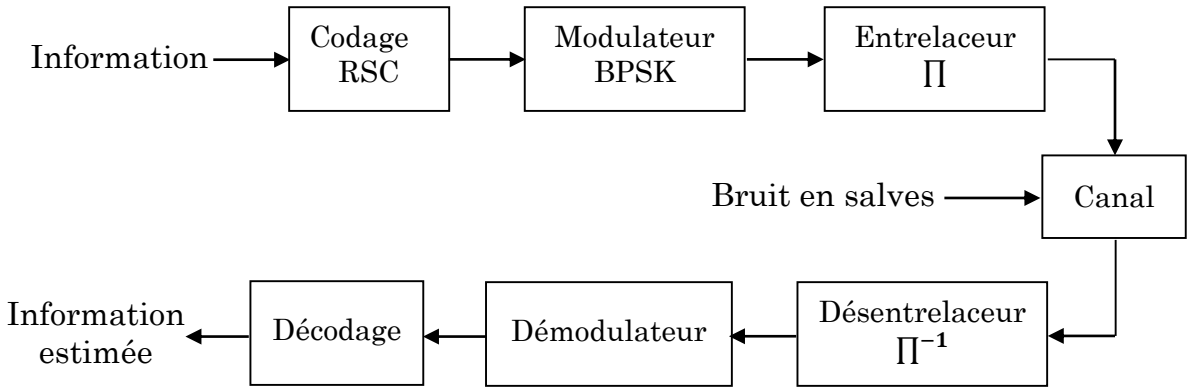


Fig 3.12 Chaîne de transmission numérique avec entrelaceur ∏

3.3.4. Performance en fonction du nombre d'états

La figure 3.15 montre les BER des deux systèmes, l'un utilise le RSC de la figure 3.3.a qui possède 4 états, et l'autre utilise le RSC de la figure 3.3.b qui possède 8 états. On peut dire, d'après la figure 3.15, qu'un système avec un RSC à 8 états est plus qualifié en termes de BER qu'un système avec RSC à 4 états de même rendement, surtout pour les forts rapports signal à bruit. Ce résultat est prévisible, parce que le treillis d'un codeur à 8 états contient plus des branches que celui d'un codeur à 4 états, donc plus de capacité de correction des erreurs.

3.3.5. Performance d'un système à CTC

Dans les sous paragraphes précédents nous avons montré l'indispensabilité des codes RSC et de l'entrelacement. Nous étudierons maintenant la performance des systèmes qui utilise deux RSC concaténés en parallèle via un entrelaceur entre les deux (c'est-à-dire un CTC).

Il faut signaler, qu'après un long tâtonnement, on a arrivé aux meilleurs facteurs de convergence qui sont $\beta = 0.1$ pour le CTC de la figure 3.5.a, et $\beta = 0.13$ pour le CTC de la figure 3.5.b.

Les figures 3.16 et 3.17 montrent les BER des deux systèmes, l'un utilise le CTC de la figure 3.5.a et l'autre utilise le CTC de la figure 3.5.b. L'analyse des courbes nous permet de conclure que

- Les décodeurs itératifs donnent des très bonnes performances si on augmente le nombre d'itérations même pour les faibles SNR, ce qui n'est pas le cas pour les décodeurs classiques;
- La rapidité de convergence des décodeurs itératifs augmente avec le SNR ;
- Le décodeur itératif à 8 états est plus performant que le décodeur itératif à 4 états. Donc l'augmentation du nombre d'états améliore la performance des turbocodes.

Une comparaison en termes de nombre d'itérations entre le décodeur itératif à 8 états et celui à 4 états est illustrée dans les figures 3.18 et 3.19. Ces figures montrent qu'on peut atteindre la performance d'un décodeur itératif à 8 états avec un décodeur à 4 états si on double le nombre d'itérations, et ceci pour des faibles SNR ($0 \leq SNR \leq 1$). Ce qui signifie que pour un BER donné le produit $nombre\ d'itérations \times nombre\ d'états$ est constant. On va utiliser ce résultat empirique dans la comparaison entre nos résultats et ceux de **C. Berrou** [2].

3.3.6. Comparaison entre nos résultats et ceux de C. Berrou

La figure 3.20 tirée de l'article de **C. Berrou** [2], montre que pour un $SNR = 0\ dB$ toutes courbes de BER passe par le même point quelque soit le nombre d'itérations, ce qui veut dire qu'il n'y a pas une amélioration de performance pour un $SNR = 0\ dB$ même si on augmente le nombre d'itérations. Dans notre cas c'est le contraire comme montrent les figures 3.16 et 3.17. L'augmentation du nombre d'itération donne des améliorations du BER pour un $SNR = 0\ dB$. Cette conséquence est due au facteur de convergence β.

C. Berrou a déclaré dans le même article qu'il a atteint un BER de 10^{-5} pour un $SNR = 0.65\ dB$ seulement avec 10 itérations en utilisant un CTC de

rendement 1/3, 16 états et d'entrelaceur 64×64. Avec le CTC de la figure 3.5.b, de rendement 1/3, 8 états et d'entrelaceur 64×64 on a arrivé au même BER de 10^{-5} pour un $SNR = 0.65\ dB$. La règle empirique exposée dans le sous paragraphe précédant montre qu'avec un CTC à 16 états, de même rendement 1/3 et d'entrelaceur 64×64, on peut atteindre le même BER de 10^{-5} avec 5 itérations seulement pour le même $SNR = 0.65\ dB$. Donc l'utilisation du facteur de convergence dans le décodeur itératif diminue le nombre d'itérations de la moitié !

Il est important de signaler que dans le calcul du $BER = 10^{-5}$, avec le CTC à 8 états, on a généré une séquence de taille $2^{20} = 1048576$ bits, qui contient $2^8 = 256$ paquets (chaque paquet contient 64×64 bits). Le décodage itératif d'une séquence bruitée de taille égale à 64×64 avec 10 itérations nécessite approximativement 8 minutes en utilisant le **MATLAB** dans un laptop i3 de fréquence 2.20 GHz et de RAM 4 Go. Le calcul du BER, pour une seule fois, avec la séquence considérée a pris $8 \times 256 = 2048$ minutes (un jour et demi) !

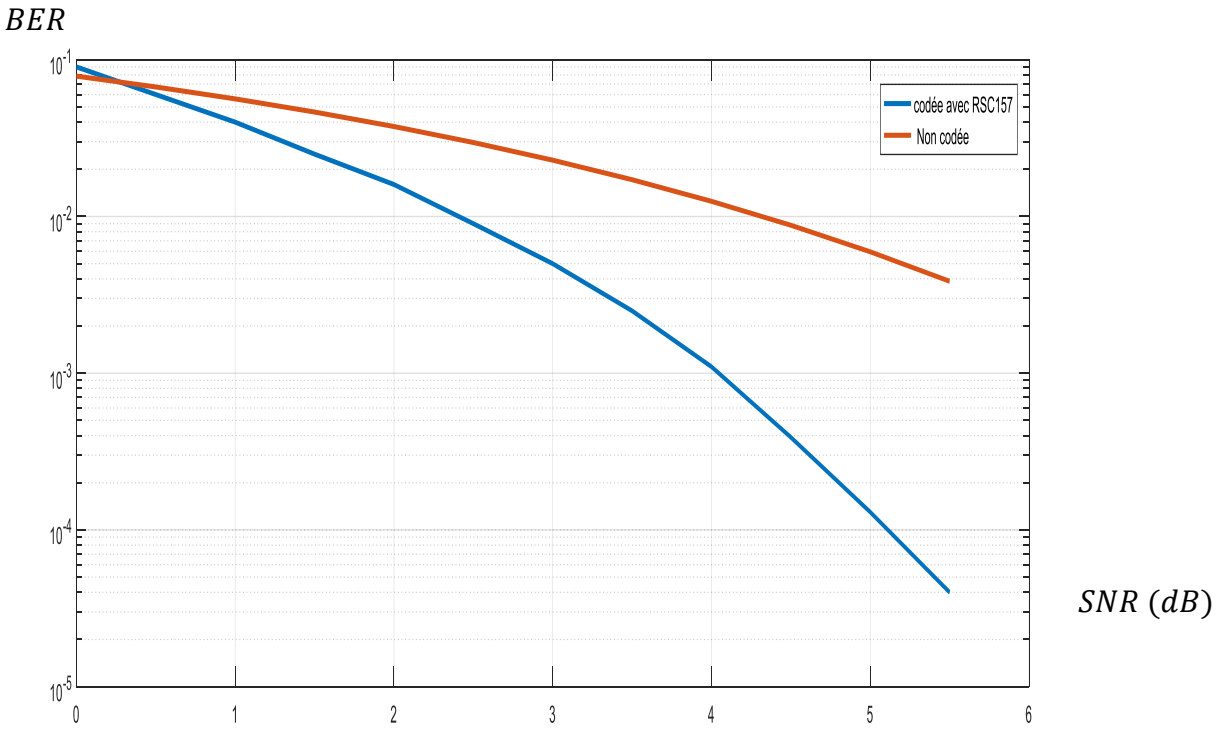


Fig. 3.13 Le BER en fonction du SNR pour un système avec codage RSC et un système sans codage

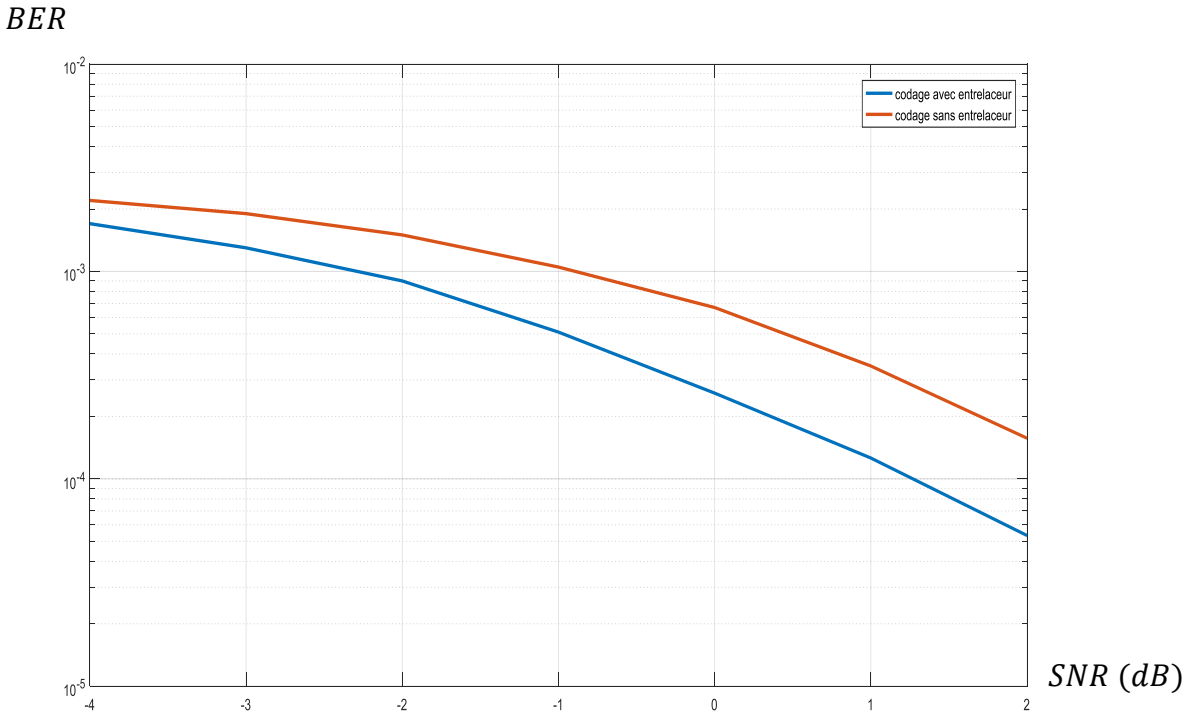


Fig. 3.14 Le BER en fonction du SNR pour un système codé avec entrelaceur et un système codé sans entrelaceur.

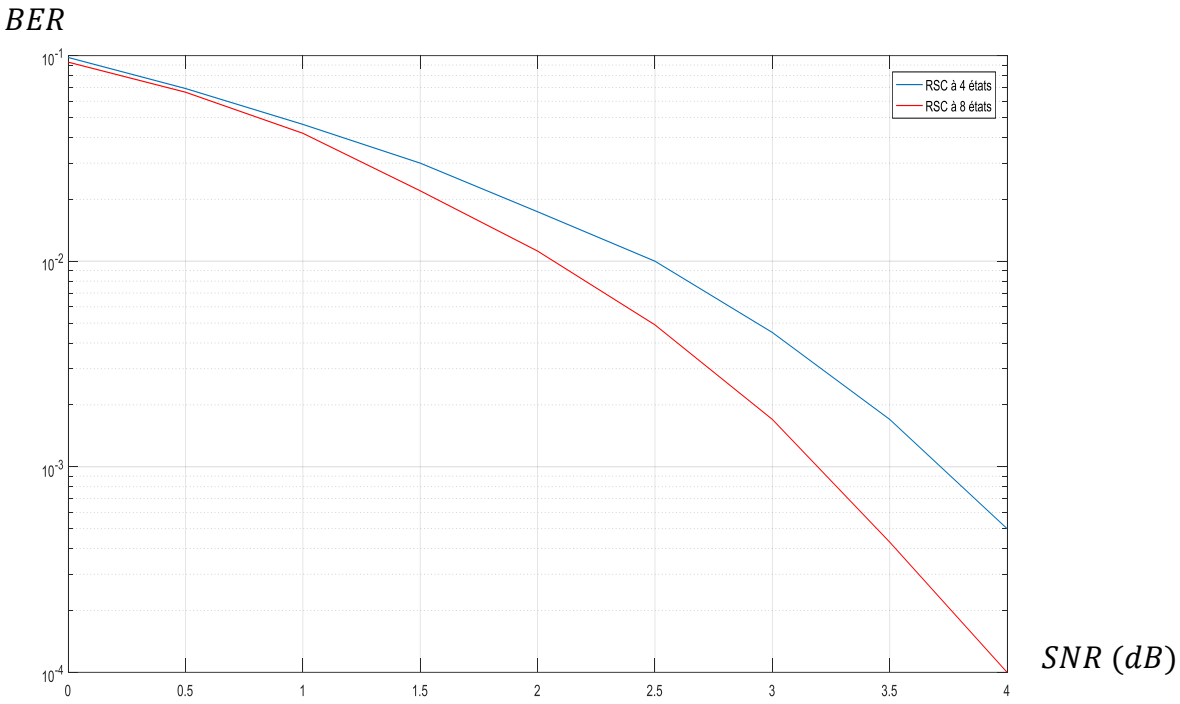


Fig. 3.15 Le BER en fonction du SNR d'un RSC à 8 états et un RSC à 4 états.

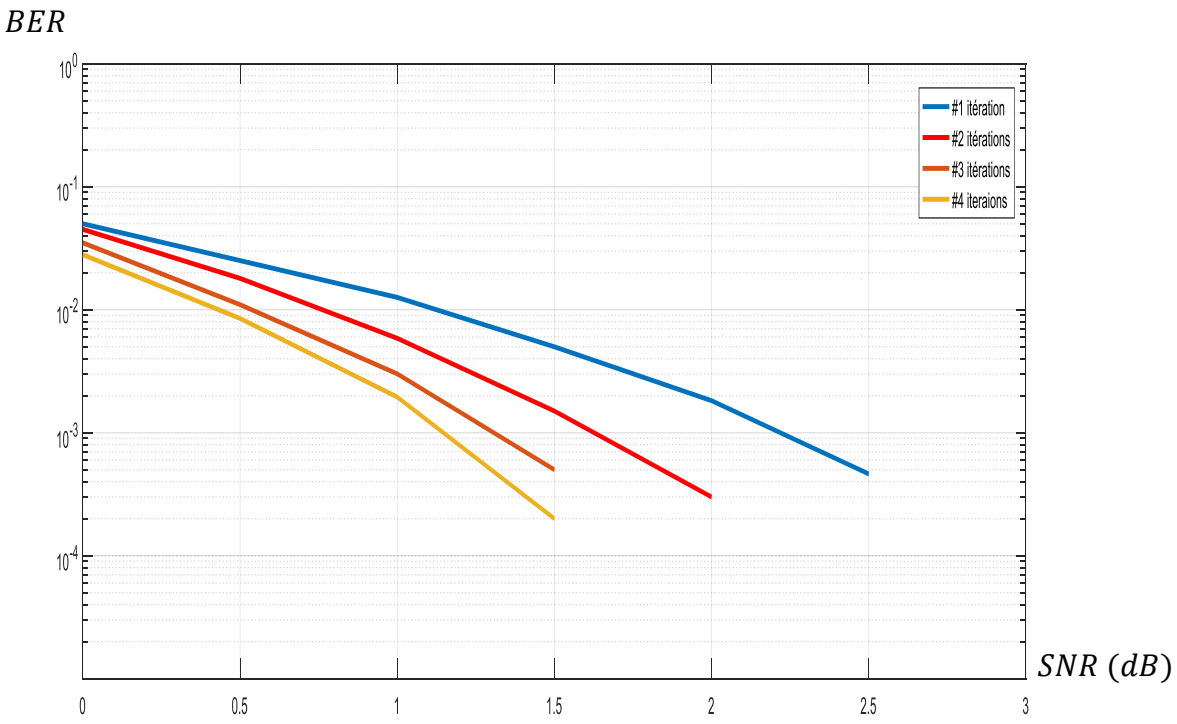


Fig. 3.16 Le BER en fonction du SNR du CTC de la figure 3.3.a pour différentes itérations.

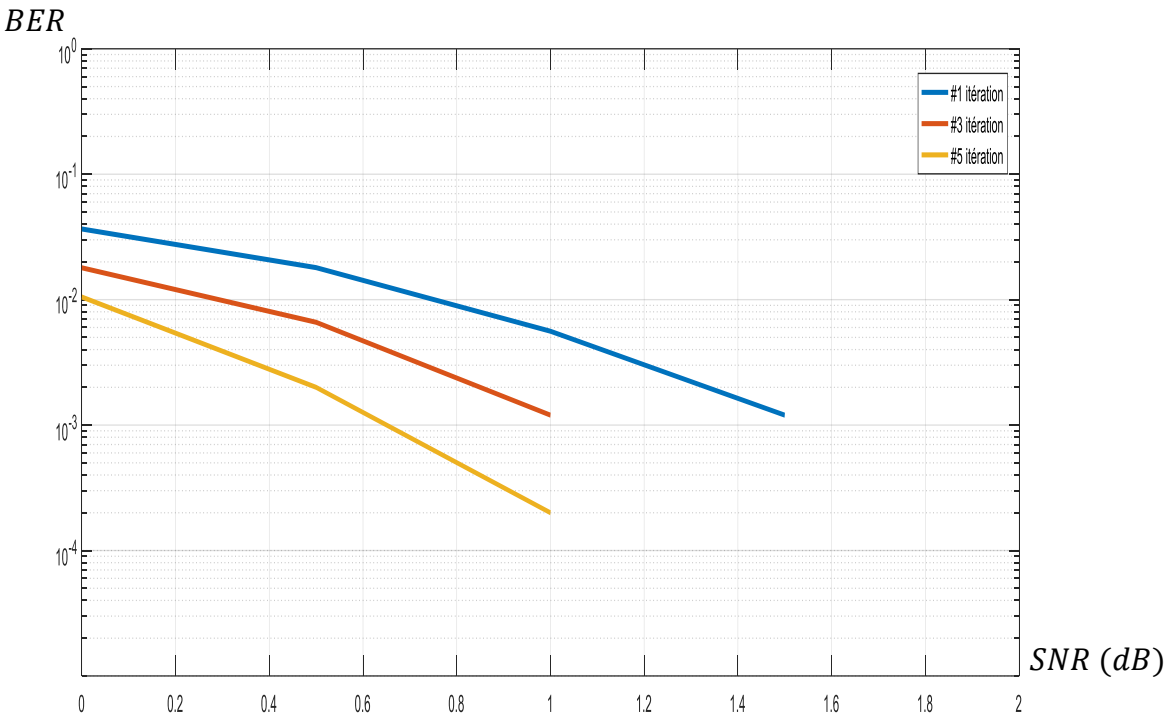


Fig. 3.17 Le BER en fonction du SNR du CTC de la figure 3.3.b pour différentes itérations.

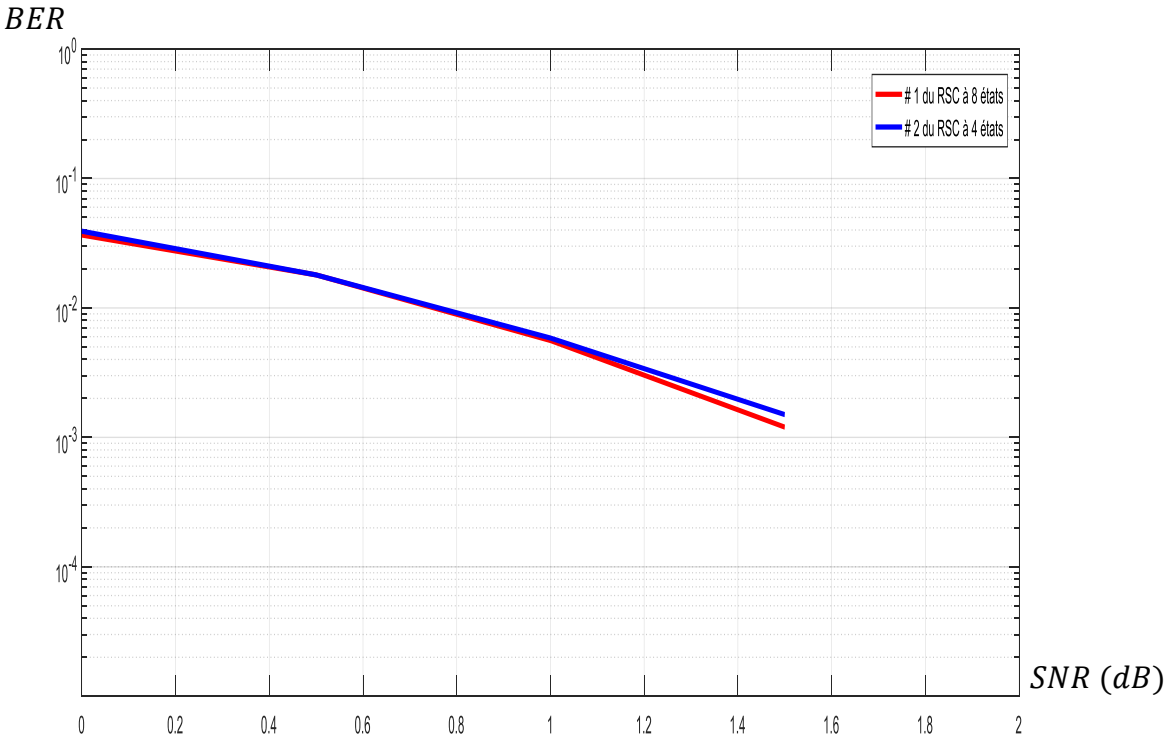


Fig. 3.18 Les BER en fonction du SNR des CTC de la figure 3.3.a pour deux itérations et CTC de la figure 3.3.b pour une seule itération.

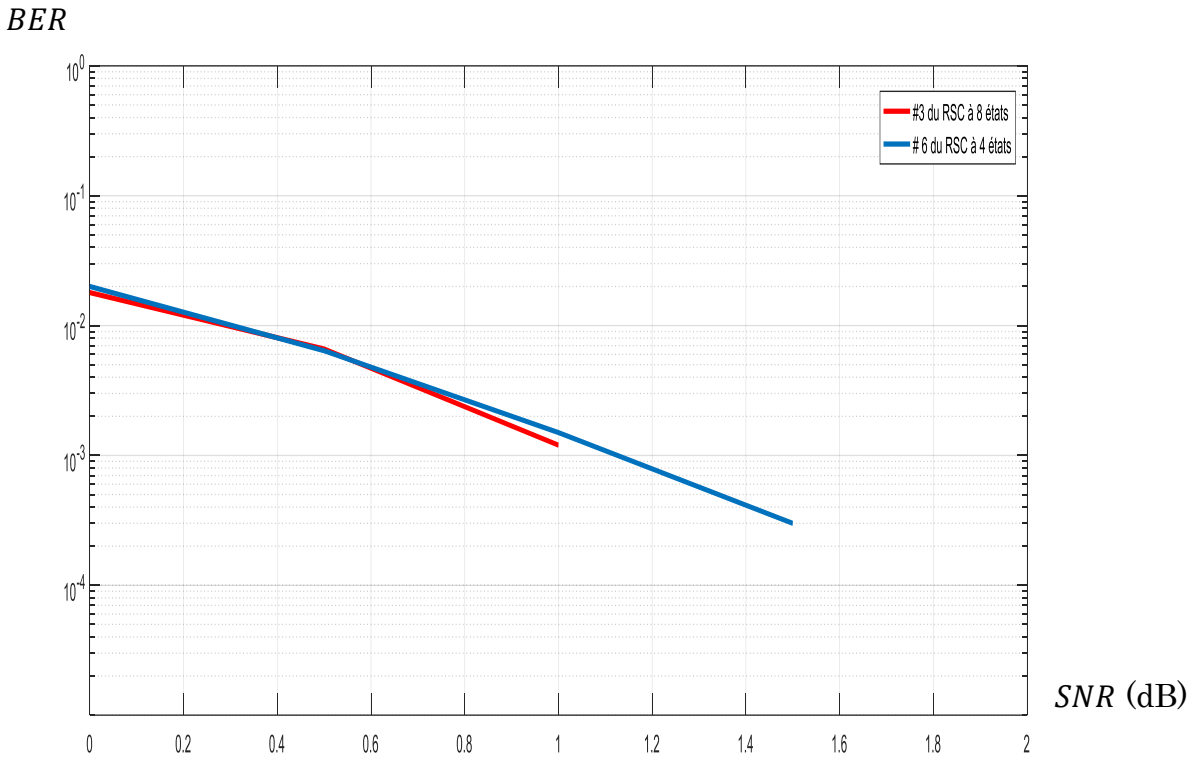


Fig. 3.19 Les BER en fonction du SNR des CTC de la figure 3.3.a pour trois itérations et CTC de la figure 3.3.b pour six itérations.

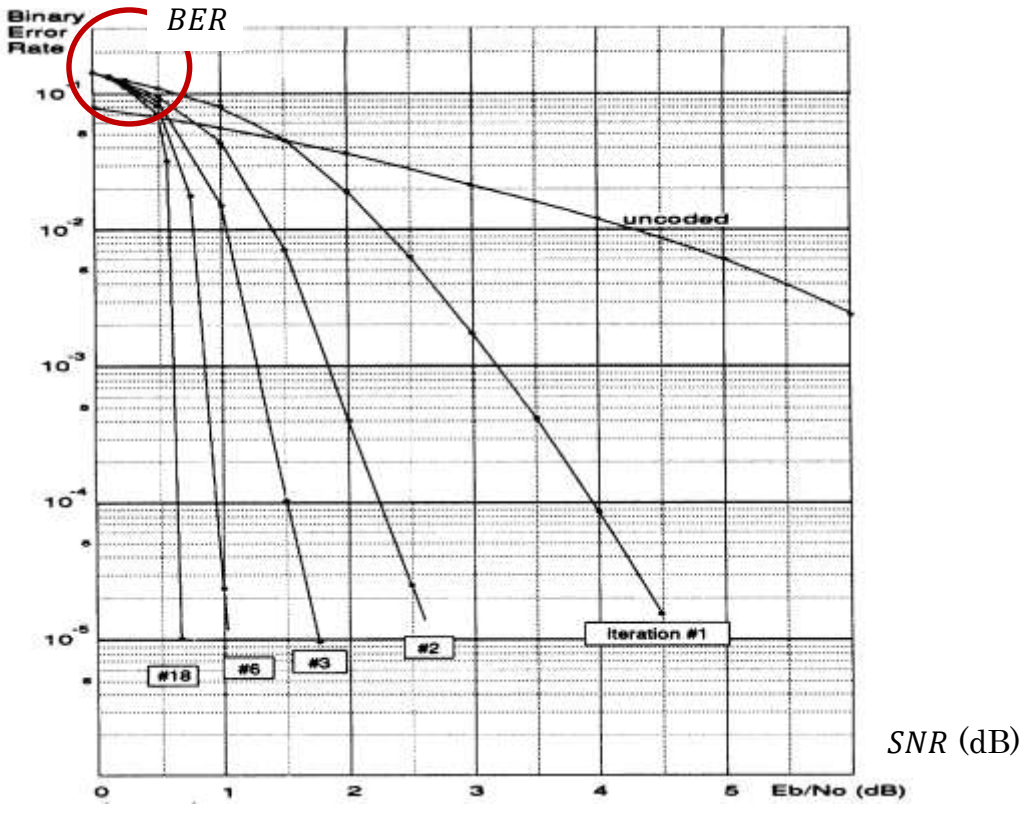


Fig. 3.20 Résultats de C.Berrou [2] pour $SNR = 0\,dB$

3.4 Conclusion

Notre simulation a permis de calculer les performances des systèmes avec CTC pour un bruit AWGGN (où le paramètre de forme est choisi = †□.□) en utilisant l'algorithme SOVA. Les résultats obtenus sont en conformité avec la théorie, et montre que les CTC sont parmi les meilleurs codes existants pour un bruit AWGGN.

La distribution gaussienne généralisée GGD est bien présenté ici, où on a présenté une méthode pratique pour générer des échantillons i.i.d suivent une telle distribution, et on a vu graphiquement en traçant les échantillons générés que, plus le paramètre de forme est proche de zéro plus le bruit est impulsif.

La simulation montre aussi, avec une comparaison entre nos résultats et ceux de C. Berrou, que l'incorporation d'un facteur de convergence convenablement choisi, dans le décodeur itératif, diminue le nombre d'itérations de la moitié pour un BER donné, corollairement, ce facteur augmente la vitesse de convergence même pour un SNR= 0 dB, et aussi à partir de l'analyse les figures, nous concluons une relation générale qui peut être prise à l'avance pour nous aider à faciliter le travail: (le nombre d'itération) X (Le nombre d'état du codeur) = Cte

Notre travail s'est limite bien sûr aux CTC parallèles avec modulation BPSK, les plus utilisées dans le domaine. Il existe bien entendu d'autres façons d'associer des codes, on peut par exemple utiliser deux codes en bloc pour coder respectivement les lignes et les colonnes d'une matrice contenant des données. Ce schéma de codage, appelé code produit, présente aussi des bonnes performances.

Notre travail s'est limité bien sûr aux turbocodes convolutifs avec bruit AWGGN à GGD symétrique en utilisant le SOVA. Notre étude peut être étendue à des bruits AWGGN à GGD asymétriques ou corrélés qui peuvent représenter des modèles de bruit pour certaines applications.

Conclusion générale

Selon le cahier de charge exposé, notre travail concernant l'étude d'un canal à bruit AWGGN avec un décodage itératif en utilisant le SOVA, a englobé deux parties : une partie théorique et l'autre une simulation.

La partie théorique a pour but de dégager les fondements théoriques de ces codeurs de canaux, à s'avoir : les théorèmes fondamentaux du codage de canal, le principe des turbocodes convolutifs, la notion de l'information extrinsèque et le principe de décodage itératif, la notion de l'algorithme de **Viterbi** a sortie souple SOVA (en donnant des exemples numériques) et de dégager aussi les autres fonctions d'un système de communications numérique, comme la modulation BPSK et la notion du bruit AWGGN à GGD symétrique. Les deux premiers chapitres ont été consacrés à cette partie.

La deuxième partie était la mise en œuvre d'une simulation permettant d'évaluer les performances des CTC, et cela en estimant le taux d'erreurs binaire (BER) de ces codes en fonction du SNR. Le bruit envisagé dans notre travail est le bruit AWGGN. Le décodage étant réalisé à l'aide d'un décodeur itératif basé sur le SOVA. La géniale idée qui consiste à incorporer un facteur $\boldsymbol{\beta}$ dans le décodeur itératif dit facteur de convergence, améliore les performances des CTC, surtout pour les faibles SNR. Au niveau de la programmation du SOVA une nouvelle idée consiste à considérer le treillis inverse comme était un treillis direct à un arrangement d'états près a été utilisée.

Nos moyens informatiques disponibles ont limité la génération de la séquence d'information pseudo-aléatoire à une longueur de $\mathbf{2^{16} = 65536\ \boldsymbol{bits}}$, et les calculs des intervalles d'incertitudes dans les courbes obtenus (temps d'exécution des programmes est très grand).

Notre simulation a permis de calculer le BER qui est le paramètre d'évaluation des CTC. L'allure des courbes obtenues correspond bien à la théorie, puisque le BER diminue avec l'augmentation du SNR, converge rapidement si on augmentation le nombre d'itérations, et ceci grâce, et ceci grâce au facteur de convergence $\boldsymbol{\beta}$ et à la sous-optimalité de l'algorithme SOVA. Cette sous-optimalité a été mis en évidence dans plusieurs littératures pour un bruit blanc additif gaussien AWGN. ainsi la comparaison entre nos résultats et ceux de C.Berrou [2] montre l'efficacité du facteur de convergence .

Puisque notre travail s'est limité aux turbocodes convolutifs avec bruit AWGGN à GGD symétrique en utilisant le SOVA, on peut l'étendre à des bruits impulsifs à GGD asymétriques ou corrélés qui peuvent représenter des modèles de bruit pour certaines applications, comme dans le développement de détecteurs multi-utilisateurs en présence de bruit impulsif basés sur la théorie de la détection localement optimale [33] [34]. Dans tel axe on peut tirer un sujet de travail de master en télécommunications pour l'année prochaine.

Bibliographie

[1] C.Berrou, codes et turbocodes, Springer Paris, octobre 2007.

[2] C.Berrou, A. Glavieux, « turbo codes : principe et application », département électronique, département signal et communications, ENST, France. 1995.

[3] T. Majoul, F. Raouafi and M. Jaïdane, « Semi-blind turbo decoding in impulsive noise channels. » IEEE , ISCCSP 2008, Malta, 12-14 March 2008, pp 810-813.

[4] F. Flitti et Y. Omarouyoub, « codes convolutifs : Etude, Simulation et valuation, », Projet de fin d'étude, ENP, département d'électronique, Sept. 1996.

[5] A. Glavieux, « codage de l'information et Modulation des signaux, » Techniques de l'Ingénieur, Traité Mesure et contrôle, R 308-1,1996.

[6] A. Spataru, Fondements de la théorie de la Transmission de l'information, Presse Polytechnique Romandes. Lausanne, Suisse, 1987.

[7] H. Sari, « Transmission des signaux Numériques, » Techniques de l'Ingénieur, Traité Electronique, E 7100, 1996.

[8] S. Haykin, Digital Communication, John Wiley & Sons, New York.1988.

[9] C. Berrou, A. Glavieux, P. Thitimajshima : « Near Shannon limit error-correcting coding and decoding : turbo-codes, » Proceedings of ICC'93, Genève, PP. 1064-1070, Mai 1993.

[10] J.B.Doré, « optimisation conjointe de codes LDPC et de leurs architectures de décodage et mise en œuvre sur FPGA », thèse de doctorat, INSA de Rennes, France, octobre 2007.

[11] B. Ferhat et A. Guemari, « Les turbocodes convolutifs avec modulation BPSK pour un canal AWGN. », Projet de fin d'étude, Université de Kasdi Merbah, département de mécanique et d'électronique, Ouaregla, Juin 2008.

[12] Y.Mori, Codage de source et de canal, Vol 6, Ed. LAVOISIER, Paris-France, 2006.

[13] V. T. Zarasoa, « Codage de sources distribué : codage symétrique et adaptatif en débit de sources corrélées », Rapport du stage, université de RENNES 1, cedex, France, Sep. 2007.

[14] J. J. Boutros, » Les Turbo Codes Parallèles et Séries : Décodage SISO Itératif et Performance ML », notes de cours, module TMC – option CST, ENST, Oct. 1998.

[15] A..H. OSSEIRAN, « On the decoding of Turbo Codes », thèse de Maître en ingénierie, Revised version, Octobre 2000.

[16] R. Fano, « A heuristic discussion of probabilistic decoding, » Information Theory, IEEE Transactions on, vol. 9, no. 2, pp. 64–74, 1963.

[17] A. J. Viterbi, « Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm, » IEEE Transactions on Information Theory , vol. IT-13, April, 1967, pp. 260-269.

[18] GD. Forney, « The Viterbi Algorithm, » Proc. of the. IEEE, vol. 61, no. 3, pp. 268-278, Mars 1973.

[19] J. Hagenauer and P. Hoeher, « A viterbi algorithm with soft-decision outputs and its applications, » in Global Telecommunications Conference, 1989, and Exhibition. Communications Technology for the 1990s and Beyon, GLOBECOM 89., IEEE, vol.3., pp. 1680–1686, 1989

[20] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, « Optimal decoding of linear codes for minimizing symbol error rate (corresp.), » Information Theory, IEEE Transactions on, vol. 20, no. 2, pp. 284–287, 1974.

[21] Robert H. Morelos-Zaragoza, The Art of Error Correcting Coding : Second Edition, JOHN WILEY & SONS, LTD, ISBN: 0-470-01558-6, England, 2006.

[22] J. G. Proakis, M. Salehi, Digital Communications, 5th edn. McGraw Hill, New York, 2008.

[23] A. Chemsa, K. Ghrissi, « Etude et évaluation des systèmes de communication numérique utilisant la modulation par codes en Treillis TCM », Projet de Fin d'Etudes, Ecole Nationale Polytechnique d'Alger, Octobre 1997.

[24] A. F. Mondragón-Torres, K. R. Narayanan and E. Sánchez-Sinencio, « Floating Gate Analog Implementation of the Additive Soft-Input Soft-Output Decoding Algorithm, » supported in part by Fulbright-CONACYT and Texas Instruments. July 5, 2001.

[25] F. H. Huang, Evaluation of Soft Output Decoding for Turbo Codes, Master thesis, Blacksburg, Virginia, May 29, 1997

[26] A. Kazem, Particules déterministes généralisées en filtrage non-linéaire : Applications défense et télécommunications, Thèse de Doctorat, l'université Toulouse III - Paul Sabatier, France, 17 septembre 2008.

[27] F. Lehmann, Les Systèmes de Décodage Itératif et leurs Applications aux Modems Filaires et Non-filaires, Thèse de doctorat, Institut National Polytechnique de Grenoble, France, Décembre 2002.

[28] M. Borda, Fundamentals in Information Theory and Coding, Springer -Verlag Berlin Heidelberg, 2011.

[29] D. Bera, T. Chakravarty and S. Chakrabarti, « Reliable Wireless Communication for Medical Devices Using Turbo Convolution Code, » Int. J. Communications, Network and System Sciences, vol. 3, pp. 703-710, August 2010.

[30] F.J. MacWilliams and N.J.A. Sloane, «Pseudo-Random Sequences and Arrays, » Proceedings of the IEEE, Vol, IT-64, PP. 1715-1728, Decem. 1989.

[31] J.C. Bic. Duponteil, JC Lmbeaux, Elements de Communications Numériques : Transmission sur Fréquence Porteuse, Tome II, Bordas et C.E.N.T.E.N.S.T, Paris, 1986.

[32] A. Larue, « Blancheur et non-gaussianité pour la déconvolution aveugle de données bruitées : application aux signaux sismiques. », thèse de doctorat, Institut national polytechnique de grenoble, Sep. 2006.

[33] A. Chemsa, Modulation avec codage itératif pour un canal à bruits nongaussiens, Thèse de doctorat en sciences, Spécialité génie électrique, université Mohamed Khider, Biskra, 2016.

[34] S. Sanchez Urrieta, K. Abed-Meraim, V. Y. Kontorovich, and M. Larra-Barron., « Locally optimum detection for DS-CDMA systems in non-gaussian noise. » In Int. Symposium on Signal Processing and its Applications (ISSPA), pages 410-413, Malaysia, Aug 2001.

[35] A. Mittal, A. K. Moorthy and A. C. Bovik : « Blind/Referenceless image spatial quality evaluator, » Signals, Systems and Computers (ASILOMAR), 2011 Conference Record of the Forty Fifth Asilomar Conference, PP. 723-727, Nov 2011.