Titre: Title:	Codes convolutionnels perforés de faibles taux et applications aux systèmes à concaténation sérielle
Auteur: Author:	Melita Adamovic
Date:	2000
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Adamovic, M. (2000). Codes convolutionnels perforés de faibles taux et applications aux systèmes à concaténation sérielle [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8815/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/8815/
Directeurs de recherche: Advisors:	David Haccoun
Programme: Program:	Non spécifié

UNIVERSITÉ DE MONTRÉAL

CODES CONVOLUTIONNELS PERFORÉS DE FAIBLES TAUX ET APPLICATIONS AUX SYSTÈMES À CONCATÉNATION SÉRIELLE

MELITA ADAMOVIC

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE
ET DE GÉNIE INFORMATIQUE

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE ÉLECTRIQUE)
MAI 2000

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

CODES CONVOLUTIONNELS PERFORÉS DE FAIBLES TAUX ET APPLICATIONS AUX SYSTÈMES À CONCATÉNATION SÉRIELLE

présenté par: ADAMOVIC Melita

en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de:

M. PIERRE Samuel, Ph.D., président

M. HACCOUN David, Ph.D., membre et directeur de recherche

M. BELZILE Jean, Ph.D., membre

REMERCIEMENTS

Le travail présenté dans ce mémoire n'aurait pu être mené à terme sans le soutien de nombreuses personnes que je me dois de remercier.

Je tiens tout d'abord à exprimer ma gratitude à mon directeur de recherche, Dr. David Haccoun, pour le soutien constant qu'il m'a apporté tout au long de ce travail. Ses conseils m'ont permis d'aller au delà des difficultés rencontrées durant le projet.

J'aimerais aussi remercier tous mes collègues de travail qui ont su créer une ambiance particulièrement agréable et intéressante.

Je tiens finalement à remercier mes frères, mes parents et tous mes amis pour leur soutien pendant ces années.

RÉSUMÉ

Les codes convolutionnels représentent une des classes de codes correcteurs d'erreur parmi les plus utilisées dans les systèmes de communication numérique. En appliquant la technique de perforation sur le code convolutionnel de faible taux, un code convolutionnel perforé de taux de codage élevé peut être construit. Cette technique consiste à perforer (éliminer périodiquement), selon une règle prédéfinie par la matrice de perforation, les symboles codés provenant du codeur de faible taux de codage. Tout en préservant la structure simple du codeur de faible taux de codage, un code convolutionnel perforé apporte un avantage quant à la largeur de bande requise par rapport à ce dernier. L'utilisation des codes convolutionnels perforés conduit à une grande flexibilité du système, car par le simple changement de matrice de perforation, on peut varier le taux de codage sans porter atteinte à la complexité du codeur et du décodeur. L'ensemble des codes de taux de codage élevés obtenu à partir du même code d'origine de faible taux introduit la notion de codage à taux variable.

Ce travail porte sur la détermination d'ensemble de codes convolutionnels perforés de taux de codage croissants et compatibles, ainsi que sur l'implantation des codes convolutionnels perforés dans un système concaténé en série.

On introduit la notion de codes convolutionnels perforés à des taux de codage dits compatibles comme étant des codes à taux variables auxquels on ajoute la propriété de compatibilité suivante: la forme du patron de perforation du code de taux R=b/V demeure inchangée pour tous les codes dont le taux de codage est supérieur à b/V.

Après une brève présentation des principaux paramètres de la recherche que nous menons, nous abordons l'étude sur la détermination d'ensemble de codes perforés de taux croissants ($1/6 < R < 7/8$) et compatibles à partir du code d'origine de taux $R=1/6$, et ceci pour deux longueurs de contraintes : $K=7$ et $K=8$. Des codes perforés obtenus ainsi présentent de très bonnes performances, malgré le fait que la démarche prise ne soit pas une méthode de recherche optimale, car le choix des patrons de perforations est restreint par la propriété de compatibilité.

L'analyse des performances d'un système constitué de deux codeurs et de deux décodeurs concaténés en série est effectuée. Cette analyse montre l'importance du choix des codes, de la longueur de séquence de l'information, mais surtout de l'algorithme de décodage utilisé dans un tel système. Le décodage appliqué dans ce système simulé est le décodage non-itératif, basé sur l'algorithme SOVA. L'amélioration de performances que cet algorithme de décodage apporte par rapport à l'algorithme de Viterbi est présentée. Ces performances ont tendance à augmenter davantage lorsque les codes récursifs systématiques sont utilisés ou lorsque le rapport signal à bruit est élevé.

Les performances d'un système concaténé utilisant le décodage non-itératif basé sur l'algorithme SOVA restent pourtant significativement plus modestes que celles introduites par les systèmes basés sur le décodage itératif («turbo»). En même temps, la complexité d'un tel système reste beaucoup plus petite que celle d'un système dit «turbo».

ABSTRACT

Convolutional codes are among the most used error correcting codes in digital communication systems. Applying the puncturing technique to a low coding rate convolutional code, a high-coding rate punctured convolutional code is obtained. Puncturing technique consists in puncturing (periodic elimination) code symbols produced by a low coding rate code following the rule represented by puncturing pattern. The obtained punctured code preserves a simple structure of the originating low-rate code while keeping the benefits introduced by high-rate codes, such as required bandwith. The use of punctured codes implies a high system flexibility, since a coding rate can vary by simple change of puncturing pattern, without any increase in either encoder or decoder compexity.

This work addresses the research for the catalog of punctured codes having increasing and compatible coding rates, as well as the implementation of punctured convolutional codes in a serially concatenated system.

Variable-rate coding is obtained if all punctured rates of interest are derived from the same low-rate code. Variable-rate coding is further specialized into rate-compatible coding by adding the restriction that all the code symbols of the higher punctured codes are required by lower-rate codes. Starting from the best known codes of constraint lengths K=7 and K=8 and coding rate R=1/6, a catalog of the best punctured codes of rates $R = \frac{1}{\nu}$, $2 \le \nu \le 5$ and $R = \frac{\nu - 1}{\nu}$, $3 \le \nu \le 8$ have been constructed. All these codes are rate-compatible over the entire range of coding rates of interest, $\frac{1}{6} \le R \le \frac{7}{8}$. Obtained punctured codes yield very good performances despite the fact that the used approach is

not an optimal one, since the choice of puncturing patterns is limited by introduced compatibility requirement.

Performance analysis of a serially concatenated system, formed of two encoders and two decoders is performed. The analysis showed the importance of the choice of encoder, information bloc length and, the most of all, decoding algorithm used in presented system. Decoding algorithm applied to the simulated system is a non-iterative one, based on algorithm SOVA. The improvement that SOVA algorithm brings into the system, compared to Viterbi algorithm is presented. It is shown that performances of presented system based on SOVA algorithm are increasing when recursive systematic codes are implemented, or when a signal to noise ratio increases.

Performance of serially concatenated system using a non-iterative decoding algorithm based on SOVA are still significantly below the ones obtained when an iterative (turbo) decoding is applied. On the other hand, the complexity of presented system is still much lower than the one using a turbo decoding technique.

TABLE DES MATIÈRES

Chapitre 3
CODES CONVOLUTIONNELS PERFORÉS

Chapitre 4
CONCATÉNATION SÉRIELLE DE CODES CONVOLUTIONNELS PERFORÉS

LISTE DES FIGURES

LISTE DES TABLEAUX

CHAPITRE 1

INTRODUCTION

Afin de transmettre et de recevoir les données numériques, les systèmes de communication numérique sont généralement constitués d'un émetteur, du canal de transmission et d'un récepteur. L'émetteur du système transmet les symboles auxquels s'ajoute le bruit provenant du canal de transmission. Le récepteur, quant à lui, produit une décision sur la séquence reçue en fonction du type de codage utilisé. Le bruit introduit par le canal affecte la décision prise par le récepteur en introduisant des erreurs au niveau de la séquence transmise par l'émetteur.

Un moyen efficace pour corriger les erreurs introduites dans la séquence transmise dans le canal de transmission est connu sous le nom de codage correcteur d'erreurs. Le principe du codage consiste à ajouter, selon des règles prédéfinies, une redondance dans la séquence d'information avant de la transmettre. Cette redondance permettra au détecteur de prendre une décision plus fiable sur l'information transmise. La mesure de la redondance est appelée le taux de codage, et représente le rapport entre le nombre de bits d'information à l'entrée du système et le nombre de symboles effectivement transmis dans le canal.

Les codes convolutionnels constituent la classe de codes correcteurs d'erreurs parmi les plus répandues et les plus utilisées dans les systèmes numériques. Comme leur pouvoir de correction d'erreurs dépend en grande partie du code utilisé, beaucoup de recherches ont été effectuées afin de développer des codes convolutionnels puissants et flexibles. Parmi

les applications des codes convolutionnels, celles basées sur les faibles taux de codage (R=1/V) sont les plus répandues. Le pouvoir de correction d'un code convolutionnel peut s'exprimer en terme de la performance du code en question. La détermination des performances des codes convolutionnels peut être conduite selon deux grandes approches qui sont l'étude des propriétés algébriques des codes convolutionnels et les simulations par ordinateur d'un modèle du système de communication. Ces deux approches d'étude des performances des codes convolutionnels sont utilisées dans ce travail.

À partir des codes convolutionnels, une classe de codes connue sous le nom de codes convolutionnels perforés [7] a été construite. C'est cette dernière classe de codes qui est utilisée dans ce mémoire. Les codes convolutionnels perforés sont les codes de taux de codage élevé (R=b/V) dont l'application devient très avantageuse lorsque la largeur de bande disponible est faible et lorsqu'une certaine flexibilité du système s'impose. Un code convolutionnel perforé est obtenu par l'élimination périodique des symboles de sortie d'un codeur convolutionnel de faible taux ($R_0=1/V_0$). La règle de cette élimination périodique est représentée par une matrice, appelée matrice de perforation. Le code perforé ainsi obtenu dépend à la fois du code d'origine de faible taux et du patron de perforation utilisé. Les codes convolutionnels perforés apportent une grande flexibilité au niveau du système de communication utilisé car, à partir d'un même code de faible taux de codage, on obtient des codes à taux de codage variables par simple changement des patrons de perforations, sans porter atteinte à la complexité du processus de décodage.

Grâce aux puissants algorithmes de décodage développés pour les codes convolutionnels, ces derniers sont souvent mis en oeuvre dans les systèmes nécessitant des faibles probabilités d'erreur, tels que les systèmes de communications sans fil, les communications par satellite, etc...

Parmi les algorithmes de décodage les plus répandus dans le codage convolutionnel, on

retrouve l'algorithme de Viterbi. Ce dernier se définit comme un algorithme optimal à maximum de vraisemblance [5].

Différentes modifications de l'algorithme de Viterbi sont proposées afin d'adapter le processus de décodage à des systèmes concaténés. Parmi ces algorithmes, on retrouve l'algorithme «maximum à postériori» (MAP) [6],[17] et l'algorithme SOVA («*Soft-Output Viterbi Algorithm*») [12]. Les deux algorithmes mentionnés, ajoutent par rapport à l'algorithme de Viterbi, une mesure de la fiabilité de la décision prise par le décodeur. Cette mesure de fiabilité permet d'obtenir une amélioration des performances d'un système concaténé constitué de deux codeurs/décodeurs.

Ce mémoire est structuré de la façon suivante:

Le chapitre 2 décrit les notions liées aux systèmes de transmissions numériques, telles que le codage convolutionnel, le spectre des distances d'un code convolutionnel ainsi que le décodage des codes convolutionnels. En ce qui concerne le codage convolutionnel, deux types de codage sont présentés et analysés: le codage convolutionnel non systématique et le codage récursif systématique. La notion des codes catastrophiques est ensuite brièvement mentionnée. Enfin, le décodage à maximum de vraisemblance est introduit en portant une attention particulière au décodage de Viterbi.

Le chapitre 3 est consacré aux codes convolutionnels perforés [7]. Plus particulièrement, le principe de codage et de décodage des codes perforés est présenté, et leurs performances étudiées. De plus, la notion de codes perforés à taux compatibles est introduite. Par la suite, la recherche des bons codes perforés à taux croissants et compatibles, obtenus à partir du code d'origine de taux $R_0=1/6$, est présentée. Cette recherche est basée sur l'étude des propriétés de ces codes, telles que la distance libre, le spectre du code et le profil de distance des colonnes. À partir de ces paramètres, les bornes supérieures des codes sont calculées. Un ensemble de «meilleurs» codes perforés à taux croissants ($1/6 < R < 7/8$) et compatibles est alors déterminé en examinant les performances de ces derniers, pour tous

les taux désirés et pour les deux longueurs de contrainte: K=7 et K=8. Finalement, pour un même taux de codage, les performances de l'ensemble de codes perforés ainsi obtenues sont comparées aux performances d'autres types de codes perforés plus connus.

Au chapitre 4, l'analyse d'un système concaténé, constitué de deux codeurs et de deux décodeurs concaténés en série est effectuée. Plus précisément, l'implantation des codes convolutionnels perforés dans un tel système est examinée. L'étude des performances du système concaténé est réalisée en simulant par ordinateur le modèle du système concaténé. Ainsi, les performances d'un tel système sont mesurées en fonction des différents paramètres qui le constituent. L'algorithme de décodage SOVA (*«Soft-Output Viterbi Algorithm»*) est appliqué au système, et les performances ainsi obtenues sont comparées à celles de l'algorithme de Viterbi.

Finalement, la conclusion de ce mémoire est présenté au chapitre 5.

CHAPITRE 2
CODES CONVOLUTIONNELS

Les systèmes de communication numériques modernes exigent des performances de plus en plus fiables. Une des façons d'augmenter la fiabilité de la communication est de transmettre les bits de redondance associés au signal d'information numérique émis, de sorte qu'au récepteur certaines erreurs provenant du canal puissent être corrigées. On parle donc de procédure de codage correcteur d'erreurs dont les codes convolutionnels représentent une forme très puissante. Ce chapitre décrit les notions de base du codage convolutionnel ainsi que les notions de décodage et de performances d'erreur.

2.1. SYSTÈME DE COMMUNICATION NUMÉRIQUE CODÉ

Les systèmes de communication numérique codés sont fréquemment représentés par une chaîne de transmission telle qu'illustrée à la figure 2.1.

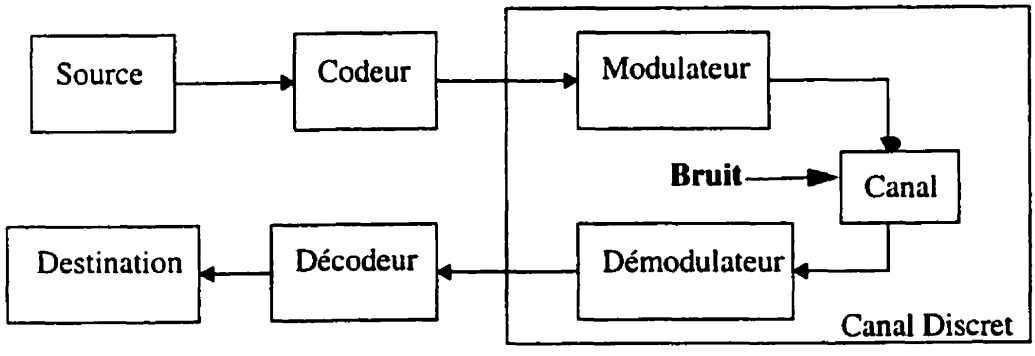

Figure 2.1 Modèle d'un système de communication numérique codé

Dans un système numérique, le modulateur, le canal et le démodulateur sont regroupés pour former un modèle théorique appelé canal discret. Quand la séquence à la sortie du

canal dépend seulement de la séquence à l'entrée et non pas des états précédents du canal, le canal discret est dit sans mémoire (en anglais: DMC ou *Discrete Memoryless Channel*). C'est ce modèle de canal qui sera utilisé tout au long de ce travail.

Le modèle le plus simple du canal DMC est le canal binaire symétrique (CBS) illustré à la figure 2.2. Pour ce canal, les symboles d'entrées sont binaires et équiprobables et les symboles de sorties sont binaires. La probabilité de transition du canal, notée p, représente la probabilité que la valeur d'un symbole est changée due au bruit ajouté au niveau du canal. La valeur de p dépend du rapport signal/bruit, du type de bruit et du type de modulation.

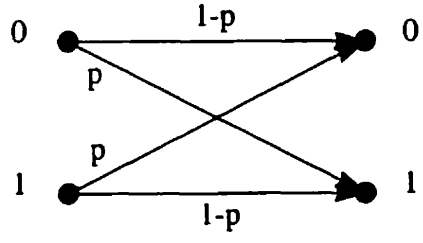

Figure 2.2 Canal binaire symétrique (CBS)

Quant aux autres blocs de notre diagramme de la figure 2.1, leurs fonctions peuvent être décrites de la manière suivante : le codeur ajoute de la redondance à la séquence d'information numérique émise par la source afin de mieux la protéger des perturbations introduites dans le canal. Cette redondance est utilisée par le décodeur dans le but de corriger certaines erreurs en estimant la séquence la plus vraisemblable. Une fois la séquence d'information codée, elle est modulée. Le rôle du modulateur est d'adapter les symboles codés au canal physique. Le bruit ajouté à la séquence modulée dans le canal est modélisé par un bruit blanc gaussien additif (BBGA) qui possède la propriété d'affecter de façon indépendante les symboles transmis. C'est un processus aléatoire dont la fonction de densité de probabilité est gaussienne de moyenne nulle et de variance $N_0/2$. À partir de la séquence

bruitée sortant du canal, le démodulateur prend une décision sur les symboles transmis.

2.2. PRINCIPE DE CODAGE CONVOLUTIONNEL

Un codeur convolutionnel est constitué d'un registre à décalage formé de K cellules, de V additionneurs modulo-2, d'un ensemble de connexions entre les additionneurs et les cellules du registre à décalage et du commutateur à V positions. Chaque codeur convolutionnel est caractérisé par les paramètres suivants:

- Taux de codage R =b/ V

où b représente le nombre de bits simultanés à l'entrée du codeur et V le nombre de symboles codés à la sortie du codeur

- Longueur de contrainte du codeur K

où K représente le nombre de cellules du registre à décalage.

Afin de mieux comprendre le fonctionnement du codeur convolutionnel ainsi que l'influence de ces paramètres sur la performance du système, prenons comme exemple un codeur convolutionnel simple représenté par la figure 2.3.

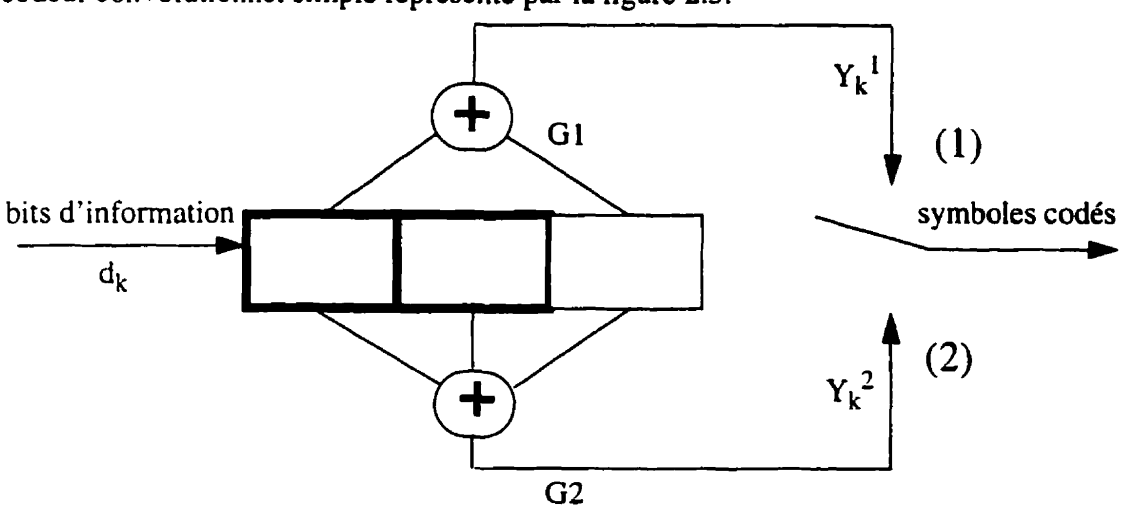

Figure 2.3 Structure d'un codeur convolutionnel

Comme on peut le constater d'après la figure 2.3, le nombre d'additionneurs dans cet exemple est V=2, le taux de codage devient R=1/2 car chaque bit à l'entrée du codeur produit deux symboles codés à sa sortie et la longueur de contrainte (nombre de cellules du registre à décalage) est K=3.

L'ensemble des connexions entre le registre à décalage et les additionneurs modulo-2 est représenté par des vecteurs générateurs. Ce sont les vecteurs générateurs du codeur qui déterminent les règles selon lesquelles les bits de redondance doivent être ajoutés aux bits d'information à transmettre. Les vecteurs générateurs G_i peuvent être exprimés par :

$$G_i = \{g_{i1}, g_{i2}, g_{i3}, \ldots, g_{iK}\}, \; i=1, \ldots, v \qquad (2.1)$$

La composante g_{ij} vaut 1 si la connexion entre la j-ème cellule du registre et le i-ème additionneur modulo-2 existe et vaut 0 sinon.

Pour l'exemple de la figure 2.3, les générateurs valent : $G_1 = (101)_2$, $G_2 = (111)_2$. Les générateurs sont très souvent exprimés sous forme octale, ce qui nous amène (dans l'exemple considéré) à la notation suivante: $G_1=5$, $G_2=7$. C'est cette dernière notation qu'on utilisera tout au long de ce travail.

Le codeur fonctionne de la manière suivante: avant que le processus d'encodage ne débute, le contenu du registre à decalage est initialisé à zéro. Les bits d'information arrivent à l'entrée du codeur de façon continue. Le bit d'information à l'entrée du codeur est injecté dans le registre à décalage et, en utilisant les vecteurs générateurs du codeur, les V symboles codés correspondants sont calculés. La séquence des symboles codés est obtenue en échantillonnant les additionneurs modulo-2 à l'aide de commutateur. Une fois

tous les bits d'information codés, une séquence de K-1 zéros est ajoutée (et codée) afin de remettre le registre à l'état initial, c'est-à-dire à l'état zéro.

Évidemment, le symbole codé ne dépend pas seulement du bit à l'entrée du codeur mais aussi du contenu des K-1 cellules du registre précédant le bit courant. La mémoire M du codeur du taux de codage R=1/V est définie par:

$$M = K-1 \tag{2.2}$$

Dans ce travail, où on considère des codes du taux de codage R=1/V, les bits d'information sont toujours considérés binaires. Alors, selon la valeur du bit d'information, le contenu des K-1 cellules du registre peut prendre une des 2^{K-1} valeurs possibles et donc :

$$\text{Nombre d'états du codeur} = 2^{K-1} = 2^{M} \tag{2.3}$$

Soit d_k le bit d'information à l'entrée du codeur à l'instant k. Les symboles codés Y_k^i (i=1, ..., V) peuvent alors s'écrire sous la forme:

$$Y_k^i = \sum_{j=0}^{K-1} g_{ij} d_{K-j} \pmod 2 \qquad i = 1, \ldots, V \tag{2.4}$$

La séquence de sortie est donc une combinaison linéaire des entrées présentes et passées. Cette séquence peut s'exprimer sous la forme du produit de convolution de la séquence d'entrée et de la réponse impulsionnelle du codeur (réponse à l'entrée 1000...); d'où le nom des codes convolutionnels.

2.3. REPRÉSENTATION DES CODES CONVOLUTIONNELS

Dans cette section, on introduit les représentations en diagramme d'états et en treillis d'encodage d'un code convolutionnel.

Diagramme d'états

Un diagramme d'états du code convolutionnel représente la relation entre les symboles codés, les bits d'information et les états du codeur. Cette représentation est rendue possible par le fait que le codeur ne peut prendre qu'un nombre fini d'états. La figure 2.4 représente le diagramme d'état du codeur de la figure 2.3.

Les noeuds du diagramme d'état représentent les états du codeur et le résultat de transition entre deux états repésente les v symboles codés qui correspondent au bit d'information à l'entrée du codeur. Chaque branche de la figure 2.4 représente les symboles codés correspondant au bit d'information indiqué entre crochets.

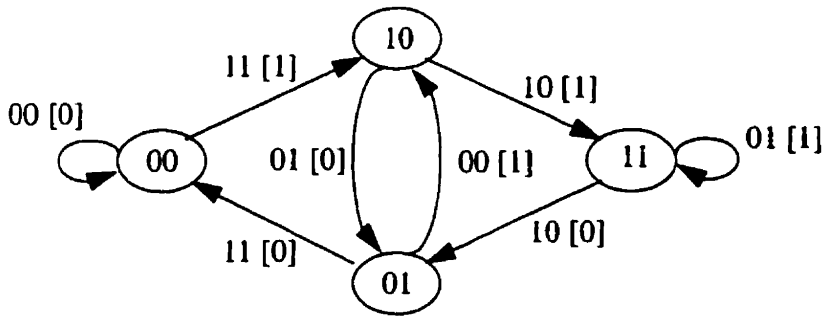

Figure 2.4 Diagramme d'état du codeur de la Figure 2.3 (K=3, R=1/2)

Le diagramme d'état permet d'analyser et de déterminer les performances d'un code parti-

culier. Par exemple, la fonction de transfert du code utilisée pour déterminer les distances de Hamming de tous les mots de code par rapport au mot de code $\underline{0}$ = (0,0,...0) peut être obtenue à partir du diagramme d'états du code. L'inconvénient est que cela devient beaucoup trop complexe lorsque K augmente et cette méthode n'est utilisée que pour des codes de faible longueur de contrainte. D'autre part, le diagramme d'état n'explicite pas la relation entre la dynamique d'encodage et la profondeur dans la séquence des bits d'information. C'est pour cette raison qu'on utilise plus souvent le treillis d'encodage pour représenter un code convolutionnel.

Treillis d'encodage

Un treillis d'encodage est une représentation du codeur convolutionnel qui tient compte du fait que le nombre d'états du codeur est fini ainsi que de la propriété de convergence. Cette dernière caractéristique traduit le fait que lorsque deux séquences d'information sont identiques pendant au moins (K-1) bits consécutifs, alors le codeur se trouve dans un état identique pour les deux séquences. Le treillis est constitué des noeuds représentant les états du codeur en question et des branches reliant les noeuds du treillis, représentant les transitions entre les états du codeur. Par rapport au diagramme d'état, la représentation en treillis introduit une information de plus, celle du temps. La notion du temps joue un rôle très important, comme on le verra par la suite, lors du décodage. La longueur du treillis représente la profondeur dans la séquence des bits d'information (notion du temps) et sa largeur, le nombre d'états du codeur. Le treillis du codeur convolutionnel de la figure 2.3 (K=3, R=1/2, G_1=5, G_2=7) est représenté par la figure 2.5, où les états sont identifiés sur la colonne de gauche.

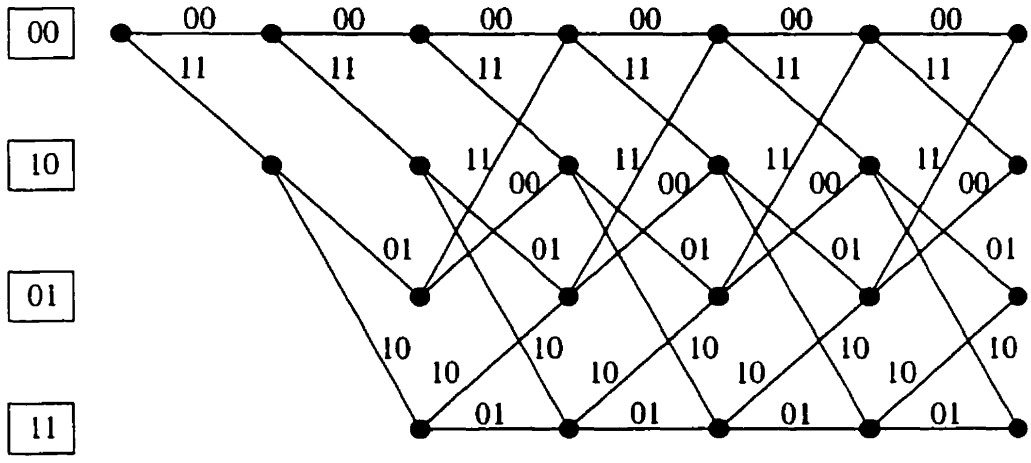

Figure 2.5 Représentation en treillis du code convolutionnel de la Figure 2.3

Comme on peut le constater à la figure 2.5, le treillis du code de taux de codage R=1/V a toujours deux branches sortant de chaque état. Dans le cas des taux R=1/V, il y a un bit à l'entrée du codeur à chaque instant et par conséquent, deux possibilités pour le prochain état. En général, pour le taux de codage R=b/V on a b bits à l'entrée du codeur à chaque instant, ce qui est équivalent à 2^b possibilités pour l'état suivant ou encore, 2^b branches arrivant à chaque noeud et 2^b branches sortant de chaque noeud du treillis.

2.4. PROPRIÉTÉS DE DISTANCE DES CODES CONVOLUTIONNELS

Une des caractéristiques les plus importantes d'un code est sa longueur de contrainte K, car le pouvoir de correction du code augmente avec K. Les performances d'un code convolutionnel dépendent aussi des propriétés suivantes : la distance libre d_{free} du code et le spectre de distance de Hamming. Le poids de Hamming (W_H) d'un mot de code est égal au nombre de symboles 1 dans ce mot de code. La distance de Hamming d_H entre deux mots de code est égale au nombre de symboles où ils diffèrent. La fonction de distance des

colonnes (FDC) d'ordre n, $d_c[n]$, d'un code convolutionnel est le poids minimal d'un mot de code calculé sur les n premières branches dont la première est non nulle:

$$d_c[n] = \min d_H(X_n^{(U)}, X_n^{(V)}), \qquad X^{(U)} \neq X^{(V)} \qquad (2.5)$$

FDC est une fonction non décroissante avec n dont la valeur maximale atteinte est notée d_{free}.

La distance libre d_{free} représente la distance minimale entre deux mots de code de longueur infinie qui ont leur première branche différente :

$$d_{free} = \lim_{n \to \infty} dc[n] \qquad (2.6)$$

La distance libre est une des mesures principales du pouvoir de correction d'erreur du code convolutionnel en conjonction avec un décodage à maximum de vraisemblance.

Le spectre des distances de Hamming représente la distribution du nombre de mots de code de longueur indéterminée ayant un poids de Hamming donné. Dans le cas des codes convolutionnels, un spectre de N raies est défini comme l'ensemble des chemins de poids inférieur ou égal à N qui divergent de l'état 0 et y reconvergent plus loin dans le treillis. Le spectre d'un code est composé de trois termes :

- d : le poids de Hamming
- a_d : le nombre de mots de code de poids d
- c_d : le nombre de bits d'information 1 correspondant à tous les mots de code de poids d.

Chaque triplet $\{d, a_d, c_d\}$ compose une raie du spectre. Le spectre d'un code est très souvent représenté sous la forme de tableau. Les 10 raies de spectre du code : K=3, R=1/2,

G1=5, G_2=7 sont fournies dans le tableau 2.1.

Tableau 2.1 Spectre des distances de Hamming du code convolutionnel, K=3, R=1/2, G_1=5, G_2=7

d	a_d	c_d
5	1	1
6	2	4
7	4	12
8	8	32
9	16	80
10	32	192
11	64	448
12	128	1024
13	256	2304
14	512	5120

Le poids correspondant à la première raie du spectre représente d_{free}. Dans le cas représenté à la figure 2.3, la distance libre du code est égal à 5, il existe un seul chemin à cette distance et 1 bit en erreur sur ce chemin; il y a 2 chemins à la distance 6 et 4 bits en erreur en tout sur ces chemins, etc.

Le spectre d'un code peut être obtenu soit en utilisant la fonction de transfert [26], ce qui devient trop complexe lorsque K augmente, soit par exploration dans le treillis d'encodage. Cette dernière méthode est utilisée afin de concevoir le logiciel [18] qui effectue le calcul du spectre du code en question. Une fois le spectre calculé, on l'utilise pour déterminer la borne supérieure sur les performances du code, c'est-à-dire la borne sur la probabilité d'erreur du code convolutionnel.

2.5. BORNE SUPÉRIEURE SUR LA PROBABILITÉ D'ERREUR PAR BIT

Pour les canaux sans mémoire (DMC), la probabilité d'erreur par bit pour l'ensemble des codes est bornée par [22]:

$$\overline{P_b} < \frac{2^{-K(R_0/R)}}{\left(1-2^{-(R_0/R-1)}\right)^2} \qquad , R < R_0 \qquad (2.7)$$

où K est la longueur de contrainte du code, R son taux de codage. R_0 est un paramètre qui ne dépend que du canal et s'écrit:

$$R_0 = -log_2\left(\sum_y \left[\sum_x P(x)\sqrt{P(y/x)}\right]^2\right) \qquad (2.8)$$

où x est le symbole codé transmis, y le symbole reçu et *P(y/x)* la probabilité de transition du canal.

L'équation (2.7) décrit la chute exponentielle de la probabilité d'erreur en fonction de la longueur de contrainte du code convolutionnel. Théoriquement, cette probabilité peut être arbitrairement faible si K est suffisamment grand, à la condition que le taux de codage R soit inférieur à R_0 (pour les valeurs de R supérieurs à R_0, la borne ne garantit plus l'amélioration des performances). D'autre part, l'effort de calcul de l'algorithme de Viterbi croît exponentiellement avec K. Par conséquent, il n'est pas possible en pratique d'obtenir une probabilité d'erreur arbitrairement faible par simple augmentation de la longueur de contrainte.

Quant à l'influence des paramètres K et R sur les performances d'un code convolutionnel, on peut conclure que la probabilité d'erreur décroît exponentiellement avec K et croît avec

R . Notons qu'en augmentant R, le nombre de bits de redondance décroît.

2.6. CODES RÉCURSIFS SYSTÉMATIQUES

Un codeur convolutionnel récursif systématique (CRS) est obtenu à partir du codeur convolutionnel qui lui est équivalent au niveau de la longueur de contrainte et des générateurs. Les deux propriétés de cette classe de codes, comme le nom l'indique déjà sont la récursivité et la propriété systématique. La récursivité consiste en une boucle de retour qui est ajoutée dans le codeur. Quant à la propriété systématique, elle se manifeste par le fait que les bits d'entrée sont toujours retrouvés intacts dans la séquence de symboles codés à la sortie du codeur.

Selon Forney [9], il existe pour chaque code non systématique, un code systématique avec une boucle de retour, possédant les mêmes propriétés de distance.
Afin d'examiner la construction des codes récursifs et systématiques (CRS), on commence par fournir la représentation polynomiale d'un code convolutionnel.
Une représentation équivalente du codeur de la figure 2.3 est illustrée par la figure 2.6.

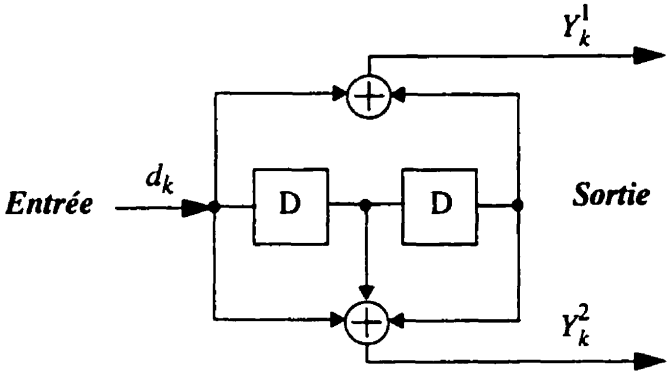

Figure 2.6 Autre représentation du code R=1/2, K=3, G_1=5, G_2=7.

Cette représentation permet de mettre plus en évidence l'aspect mémoire du code. Le registre à décalage à K cellules correspond à K-1 opérateurs de retard D; et on définit alors l'état du codeur à un instant k par la valeur des bits en mémoire à cet instant. Le nombre d'états que peut prendre le codeur est donc $M=2^{K-1}$.

L'opérateur de retard D représenté à la figure 2.6 peut être utilisé comme indicateur de temps. Dans ce dernier cas, l'équation (2.4) devient:

$$Y^i(D) = G_i(D)\,d(D) \qquad i = 1, \ldots, V \qquad (2.9)$$

où $G_i(D)$ sont les polynômes générateurs en D du code, définis par

$$G_i(D) = \sum_{j=0}^{K-1} g_{ij} D^j \qquad i = 1, \ldots, V \qquad (2.10)$$

Le codeur de la figure 2.6 est repris à la figure 2.7

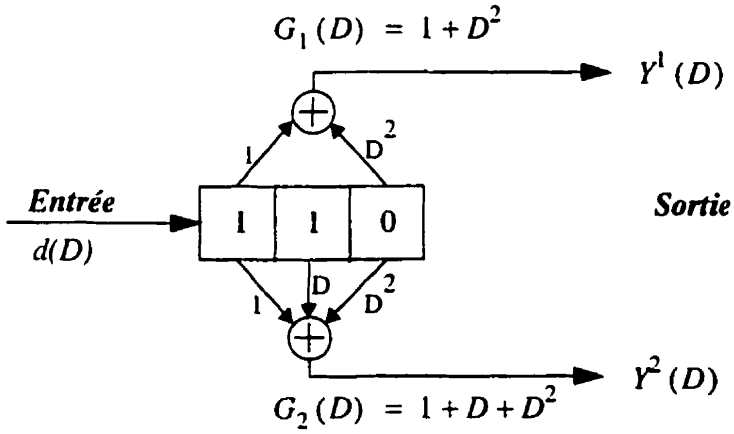

Figure 2.7 Représentation polynomiale du code R=1/2, K=3, G_1=5, G_2=7

Pour obtenir la version systématique du code de la figure 2.7, l'une des deux sorties $Y^i(D)$, i=1, 2, doit être égale à $d(D)$. Pour ce faire, on peut diviser les termes de l'équation (2.9) soit par $G_1(D)$ soit par $G_1(D)$. On choisit $G_1(D)$ dans cet exemple. On obtient alors

$$\bar{Y}^1(D) = d(D) = \bar{G}_1(D)\,d(D) \tag{2.11a}$$

$$\bar{Y}^2(D) = \frac{G_2(D)}{G_1(D)}d(D) = \bar{G}_2(D)\,d(D) \tag{2.11b}$$

avec $$\bar{Y}^i(D) = \frac{Y^i(D)}{G_1(D)} \qquad i = 1, 2 \tag{2.12}$$

Les polynômes générateurs du code ainsi obtenu sont alors

$$\bar{G}_1(D) = 1 \tag{2.13a}$$

$$\bar{G}_2(D) = \frac{G_2(D)}{G_1(D)} \tag{2.13b}$$

Dans toute la suite, nous adoptons la notation $G = (\bar{G}_1, \bar{G}_2)$, soit pour un code systématique $G = (1, \bar{G}_2) = (1, G_2/G_1)$.

En introduisant le polynôme A(D) défini par

$$A(D) = \frac{d(D)}{G_1(D)}, \tag{2.14}$$

les équations (2.11a) et (2.11b) deviennent

$$\bar{Y}^1(D) = d(D) \tag{2.15a}$$

$$\bar{Y}^2(D) = G_2(D)\,A(D) \tag{2.15b}$$

Le code ainsi construit, dont les sorties sont définies par les équations (2.15a) et (2.15b), est systématique et récursif. Le caractère récursif apparait plus clairement en ramenant l'équation (2.14) dans l'espace temporel. Elle devient alors

$$d_k = \sum_{j=0}^{K-1} g_{1j} a_{K-j} \tag{2.16}$$

En tenant compte du fait que toutes les opérations sont faites modulo 2, et en faisant l'hypothèse que g_{10}=1, le symbole a_k peut s'exprimer récursivement en fonction des symboles a_{K-j} (j=1, 2, ..., K-1) et du symbole d_k:

$$a_k = d_k + \sum_{j=1}^{K-1} g_{1j} a_{K-j} \tag{2.17}$$

Par conséquent, dans le cas d'un code CRS, ce sont désormais les symboles a_k qui sont contenus dans le registre à décalage du codeur. A partir des relations (2.15a), (2.15b) et (2.16), les sorties Y^1 et Y^2 du code CRS à un instant k peuvent s'écrire sous la forme

$$Y_k^1 = d_k = \sum_{j=0}^{K-1} g_{1j} a_{K-j} \tag{2.18a}$$

$$Y_k^2 = \sum_{j=0}^{K-1} g_{2j} a_{K-j} \tag{2.18b}$$

La construction du code CRS à partir du code CNS s'est donc faite en conservant les mêmes séquences génératrices, et en substituant simplement les symboles a_k aux symboles d_k.

Un codeur CRS équivalent au codeur de la figure 2.3 est représenté à la figure 2.8. Comme la figure 2.8 l'indique, les bits d'information à l'entrée du codeur représentent une des deux sorties du codeur. Cette sortie du codeur est appelée la sortie systématique.

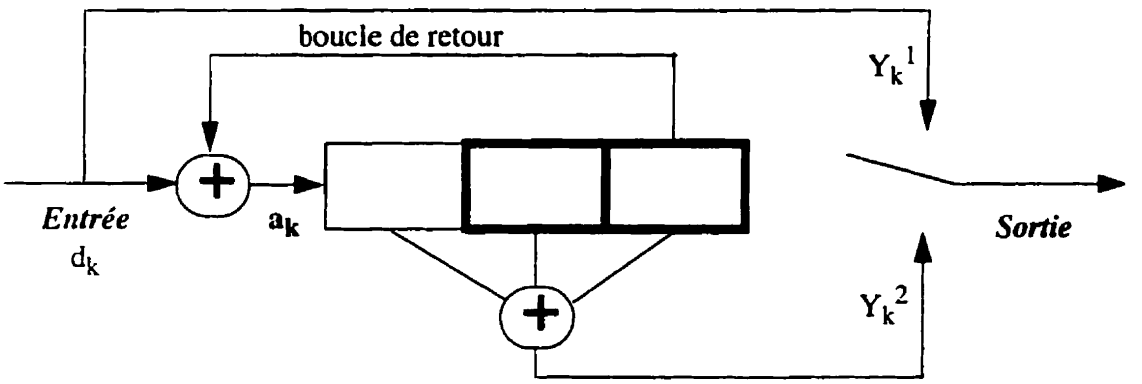

Figure 2.8 Structure d'un codeur récursif systématique (K=3, R=1/2)

A l'arrivée d'un bit d_k, celui-ci est additionné modulo 2 aux cellules du registre selon les connexions de l'additionneur à l'entrée du codeur. Le bit a_k résultant de l'addition est alors inséré dans le registre à décalage, et la sortie Y_k^2 calculée.

En comparant les structures des deux codeurs équivalents, c'est à dire le codeur convolutionnel non-systématique et le codeur récursif systématique (figures 2.3 et 2.8 respectivement), on remarque que le vecteur générateur du code diffère d'un cas à l'autre. Comme on a pu le constater dans la section 2.2, le vecteur générateur correspondant au code convolutionnel non-systématique représenté à la figure 2.3 peut être exprimé comme suit:

$$\underline{G}_{\text{convolutionnel}} = (G_1, G_2) = (5, 7) \qquad (2.19)$$

Tandis que le vecteur générateur du codeur récursif systématique illustré à la figure 2.6 vaut:

$$\underline{G}_{\text{récursif systématique 1}} = (1, 7/5) \qquad (2.20)$$

Le principe de fonctionnement du codeur CRS est le même que celui du codeur convolutionnel non-systématique (CNS) sauf que due à la boucle de retour les bits d'information ne sont pas injectés directement dans le registre à décalage (voir la figure 2.8) comme pour les codes CNS. Ainsi, le contenu de la première cellule de registre à décalage ne dépend pas seulement du bit d'information à l'entrée mais aussi du contenu des (K-1) cellules de registre. Par conséquent, il ne suffit plus d'envoyer une queue de (K-1) zéros afin de remettre le codeur en état initial. Une fois la séquence d'information transmise, les (K-1) bits de la queue doivent être calculés en fonction de l'état final du registre à décalage. Il faut noter qu'on considère les (K-1) dernières cellules du registre comme étant la mémoire du code CRS [17].

À partir de chaque codeur convolutionnel non-récursif et non-systématique de taux R=1/V

on peut obtenir V codeurs CRS en choisissant un des V additionneurs modulo-2 pour la boucle de retour. Le deuxième codeur récursif systématique (par rapport à celui de la figure 2.8) obtenu à partir du même codeur de la figure 2.3 est représenté à la figure 2.9. Dans ce cas, le vecteur générateur du code devient:

$$\underline{G}_{\text{récursif systématique 2}} = (1, 5/7) \qquad (2.21)$$

Notons qu'il n'existe jamais de connexion entre la première cellule du registre et la boucle de retour pour les codeurs récursifs.

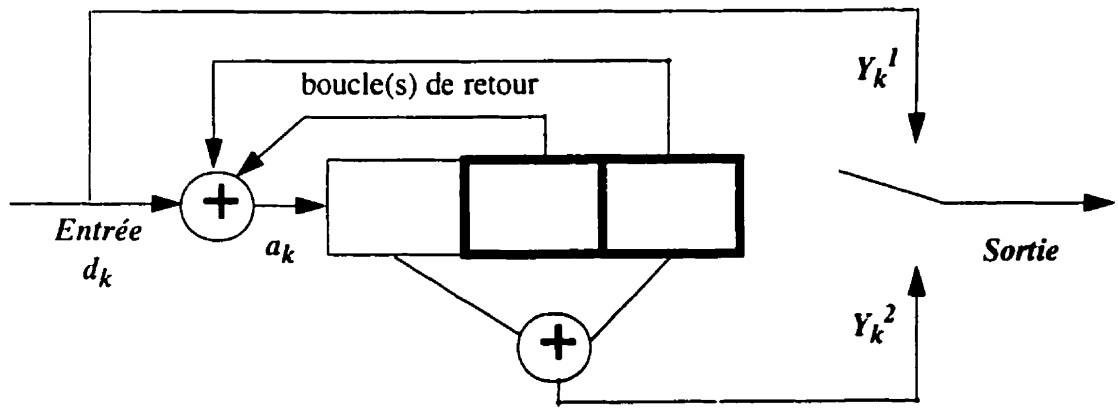

Figure 2.9 Structure du deuxième codeur récursif systématique (K=3, R=1/2)

La structure du treillis de code CRS est la même que celle du code CNS à partir duquel il a été construit, c'est-à-dire que, pour une même transition entre états, les mots de code sont identiques. Toutefois, les séquences de sorties des codes CRS et des codes CNS ne correspondent pas à la même séquence d'entrée. Cela implique qu'au niveau du spectre du code, les deux types de codes possèdent la même distance libre, les mêmes coefficients a_d mais des coefficients c_d différents. Puisque les deux types de codes ont la même distance libre, ils peuvent alors être considérés comme étant équivalents à des rapports signal à bruit élevés.

En examinant les spectres des codes, il a été démontré [26] que les coefficients c_d des codes CRS augmentent moins vite que les coefficients c_d des codes CNS. Ainsi, les performances des codes récursifs systématiques à des faibles rapports signal à bruit sont légèrement supérieurs aux codes convolutionnels non-récursifs.

Dans un système contenant deux codeurs/décodeurs concaténés en série (voir Chapitre 4) ou en parallèle, il a été prouvé que l'utilisation des codes CRS apporte une amélioration significative des performances par rapport à des codes CNS.

2.7. DÉCODAGE DES CODES CONVOLUTIONNELS

Le processus de décodage des codes convolutionnels consiste à retrouver la séquence d'information à partir de la séquence reçue. La séquence transmise est affectée par le bruit provenant du canal et, donc n'est pas connue au niveau du décodeur. C'est pour cela que le processus de décodage est beaucoup plus complexe que le processus de codage tout en lui étant inverse au niveau opérationnel.

Cette section porte sur une des techniques de décodage probabiliste des codes convolutionnels appelée décodage à maximum de vraisemblance. Le décodeur utilisant l'algorithme de Viterbi [22] qui fait partie de la classe des décodeurs à maximum de vraisemblance est un des plus utilisés dans l'industrie. On l'introduit plus loin dans la section.

2.7.1. DÉCODAGE À MAXIMUM DE VRAISEMBLANCE

Le décodage à maximum de vraisemblance est un décodage probabiliste, c'est-à-dire un processus dont l'objectif est de retracer, à partir d'une séquence reçue la séquence transmise la plus probable. Le processus de décodage revient donc à chercher le chemin dans le treillis d'encodage qui se trouve à la plus petite distance de la séquence reçue. Pour atteindre ce but, une mesure de vraisemblance est utilisée par le décodeur. Soit $\underline{U}$ la séquence transmise, $\underline{X}^{(\underline{U})}$ la séquence codée et $\underline{Y}^{(\underline{U})}$ la séquence reçue . $\underline{Y}^{(\underline{U})}$ est utilisé par le décodeur afin d'estimer la séquence transmise la plus probable ($\underline{\hat{U}}$). Ceci est illustré à la figure 2.10.

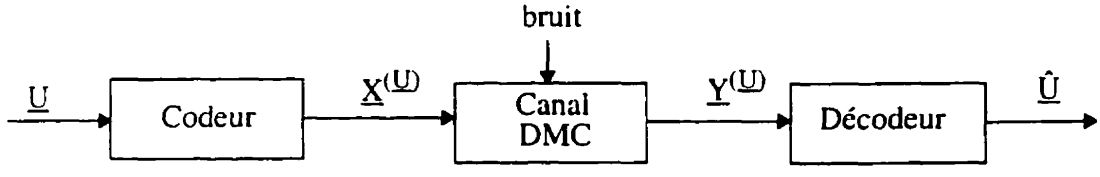

Figure 2.10 Schéma de principe de la chaîne de décodage

On définit la fonction de vraisemblance comme la probabilité conditionnelle suivante:

$$P(\underline{Y}^{(\underline{U})}|\underline{X}^{(\underline{U})}) \tag{2.22}$$

Un décodeur à maximum de vraisemblance choisira $\underline{\hat{u}}$ pour lequel:

$$P(\underline{Y}^{(\underline{U})}|\underline{X}^{(\underline{\hat{U}})}) > P(\underline{Y}^{(\underline{U})}|\underline{X}^{(\underline{U})}),\ \underline{\hat{U}}=\underline{U} \tag{2.23}$$

L'objectif du processus de décodage à maximum de vraisemblance revient alors à choisir le vecteur d'informations $\underline{U}$, en connaissant $\underline{Y}^{(\underline{U})}$, qui maximise cette fonction de vraisemblance [10]. Pour la branche i dans le treillis, la mesure de vraisemblance peut s'écrire comme suit [5]:

$$\gamma_i = \sum_{j=1}^{v} \log\left(P(y_{ij}/x_{ij})\right) \tag{2.24}$$

où x_{ij} est le $j^{ème}$ symbole de la $i^{ème}$ branche et y_{ij} est le symbole de la séquence $\underline{Y}^{(U)}$ correspondant à x_{ij}. Pour les N premières branches d'un chemin, la métrique cumulative qui est définie comme étant la somme des métriques de chaque branche faisant partie du chemin s'écrit:

$$\Gamma_N = \sum_{p=1}^{N} \gamma_p \tag{2.25}$$

Il s'agit donc de trouver le chemin de longueur N branches ayant la métrique totale la plus grande parmi tous les chemins explorés. Ce chemin correspond à la séquence d'information transmise la plus probable par rapport à la séquence reçue.

Pour un canal binaire symétrique (CBS) ayant la probabilité de transition $p < 1/2$, maximiser la métrique cumulative revient à minimiser la distance de Hamming entre les séquences $\underline{X}^{(U)}$ et $\underline{Y}^{(U)}$ pour toutes les séquences $\underline{X}$ possibles [22]. Supposons que la distance de Hamming entre $\underline{X}^{(U)}$ et $\underline{Y}^{(U)}$ soit $d_{\underline{U}}$. On a donc:

$$\log P(\underline{Y}|X^{(U)}) = \log(p^{d_{\underline{U}}} \bullet (1-p)^{N-d_{\underline{U}}}) = d_{\underline{U}}\log(p) + (N-d_{\underline{U}})\log(1-p) =$$

$$N\log((1-p) - d_{\underline{U}}\log(\frac{1-p}{p})) = -A - Bd_{\underline{U}} \tag{2.26}$$

A et B étant positifs, alors maximiser $\log(P(\underline{Y}^{(U)}|\underline{X}^{(U)}))$ revient à minimiser $d_{\underline{U}}$. Le décodage à maximum de vraisemblance sur un canal CBS consiste donc à trouver dans le treillis le chemin dont la distance de Hammming est la plus petite par rapport à la séquence reçue.

2.7.2. DÉCODAGE DE VITERBI

Comme on l'a vu dans la section précédente, le décodeur à maximum de vraisemblance effectue la comparaison des 2^N séquences transmises possibles pour une séquence reçue de longueur N afin de trouver celle qui est la plus vraisemblable.

L'algorithme de Viterbi [22] est l'algorithme de décodage à maximum de vraisemblance optimal au niveau de la performance d'erreur du code convolutionnel. L'algorithme de Viterbi utilise le treillis comme structure de données. La recherche de la séquence ayant la vraisemblance maximale est effectuée dans le treillis d'encodage en parcourant de façon exhaustive tous les états du treillis. À chaque niveau du treillis, tous les chemins sont prolongés de la façon suivante: on ajoute la valeur de la métrique de branche à la métrique cumulative du chemin en question. Parmi tous les chemins qui convergent à un état, on conserve seulement celui ayant la métrique cumulative la plus élevée. Ce chemin est appelé le chemin survivant. Cette opération est effectuée pour chaque état à chaque niveau du treillis. À la fin de l'opération de décodage, le chemin de métrique totale maximale correspond alors à la séquence transmise la plus probable. Il est évident que le décodeur de Viterbi est un décodeur à maximum de vraisemblance car à chaque étape il ne rejette que des chemins ne pouvant être meilleurs que les chemins choisis.

Pour les codes convolutionnels de taux de codage R=1/V, le décodage est effectué selon le treillis ayant 2 branches qui convergent à chaque état. Dans ce cas, à chaque niveau du treillis on a 2^{K-1} chemins survivants. Étant donné que la queue de (K-1) bits est ajoutée à la fin de la transmission des données pour remettre le codeur dans l'état initial, on peut ainsi déterminer le chemin décodé à partir des 2^{K-1} chemins survivants. Pendant le décodage de la queue, le nombre de survivants est divisé par 2 lorsqu'on passe d'un niveau du

treillis à l'autre. Cela implique qu'à la fin de la queue il ne reste qu'un seul survivant qui correspond au chemin le plus vraisemblable.

La figure 2.11 décrit les étapes du décodage de l'algorithme de Viterbi correspondant au codeur représenté à la figure 2.3 pour un canal CBS. Prenons que la séquence d'information à transmettre est : $\underline{U} = (1, 0, 1, 1, 0, 0)$ où les deux derniers bits représentent les bits de la queue (K=3) et, par conséquent la séquence codée est : $\underline{X}^{(\underline{U})} = (11, 01, 00, 10, 10, 11)$.

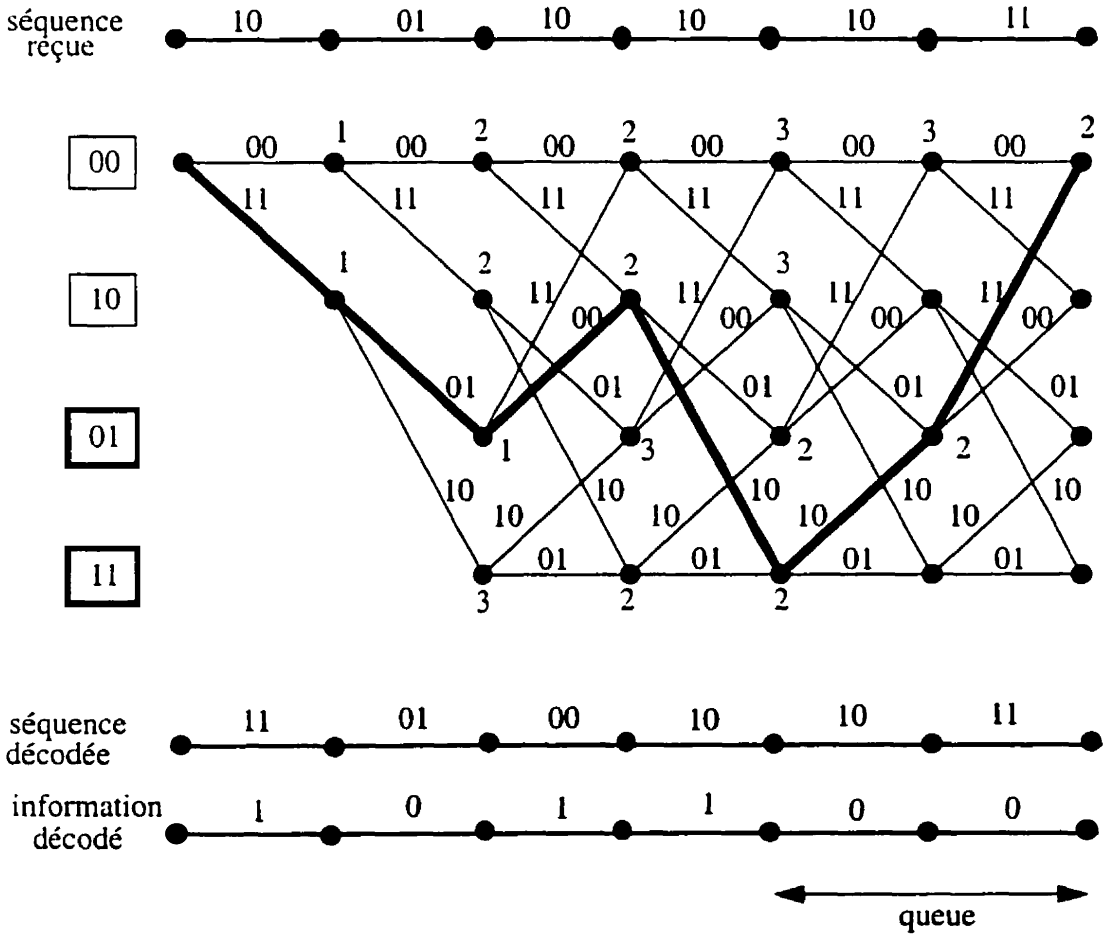

Figure 2.11 Décodage de Viterbi pour le code K=3, R=1/2, G_1=5, G_2=7

Supposons que le deuxième et le cinquième symboles aient été erronés pendant la trans-

mission à cause du bruit ajouté; donc, la séquence reçue à l'entrée du décodeur devient: $\underline{Y}^{(U)}$ = (10, 01, 10, 10, 10, 11). Le chiffre associé au noeud représente la métrique accumulée selon le chemin survivant pour le noeud (état) en question. Comme on peut le constater d'après la figure 2.11, malgré l'existance de deux symboles erronés, le message a été décodé sans aucune erreur.

Pour les codes convolutionnels de taux de codage R=b/V, le décodage est effectué selon le treillis ayant 2^b branches qui convergent à chaque état. Bien que le principe de décodage des codes de taux élevé (R=b/V) demeure le même que les codes de taux R=1/V, où à chaque niveau du treillis on compare les métriques de tous les chemins convergeant en un même état afin de trouver le survivant pour cet état, la complexité du processus de décodage des codes de taux R=b/V ne croît pas seulement avec l'augmentation de K mais aussi avec l'augmentation de b. Afin de réduire la complexité supplémentaire du processus de décodage des codes de taux élevé, les codes convolutionnels perforés [7] ont été proposés. La théorie des codes convolutionnels perforés sera présentée au chapitre 3.

Pour le code CRS, étant donné qu'il est représenté par la même structure de treillis que le code CNS à partir duquel il a été construit, il peut être décodé par les mêmes algorithmes que le code CNS.

2.8. CODES CONVOLUTIONNELS CATASTROPHIQUES

Un code convolutionnel est dit catastrophique si un nombre fini d'erreurs sur les symboles codés peut résulter en un nombre infini de bits décodés en erreur. Dans le cas du phénomène mentionné, la propagation des erreurs devient catastrophique. Pour qu'un code soit catastrophique, il suffit que dans son diagramme d'état il existe une boucle fer-

mée dont tous les symboles codés sont des zéros, à l'exception de celle qui laisse le codeur dans l'état zéro.

Au cours de l'étude des codes convolutionnels catastrophiques [16] à la Section de Communications du Département de génie électrique et d'informatique de l'École Polytechnique de Montréal sous la direction du professeur D. Haccoun, le logiciel qui détermine les codes catastrophiques a été conçu. Ce dernier outil est utilisé au long de ce travail afin de s'assurer que les codes utilisés ne soient pas catastrophiques et de rejeter ces derniers.

CHAPITRE 3

CODES CONVOLUTIONNELS PERFORÉS

Les nouvelles tendances dans le domaine du codage se dirigent vers une augmentation de la fiabilité et du taux de transmission, tout en préservant la largeur de bande utile. Cette approche peut être réalisée grâce aux codes convolutionnels à taux de codage (R=b/V) élevé, qui présentent de multiples avantages lorsqu'ils sont employés dans un canal ayant une largeur de bande limitée. Notons que le facteur d'expansion de largeur de bande est inversement proportionnel au taux de codage.

On a constaté au chapitre 2 que pour les codes à taux élevés (R=b/V), la compléxité du décodeur de type Viterbi pour un V fixé, augmente de manière exponentielle, non seulement avec la longueur de contrainte K, mais aussi avec b. Cela provient du fait que l'algorithme de Viterbi compare les métriques de tous les chemins convergeant à un état donné, afin de trouver le plus vraisemblable; or dans le treillis d'un code de taux R=b/V, il existe 2^b chemins convergeant à chaque état du treillis.

Afin d'éviter la croissance exponentielle de la complexité du décodeur avec l'augmentation du taux de codage pour les codes convolutionnels de taux élevé, les codes convolutionnels perforés ont été proposés [7].

3.1 CODAGE ET DÉCODAGE DES CODES PERFORÉS

Codage

Un code convolutionnel perforé est un code de taux élevé (R=b/V) obtenu par l'élimination périodique des symboles de sortie d'un codeur convolutionnel à faible taux (R_0=1/V_0). Ce dernier, utilisé pour générer le code perforé, est appelé codeur d'origine. La génération du code perforé se produit en gardant inchangée la structure du code d'origine grâce à un patron de perforation dont le rôle sera expliqué au paragraphe suivant (voir figure 3.1). Cette dernière caractéristique est, comme on le verra plus tard extrêmement bénéfique quant à la réduction de la complexité du décodage.

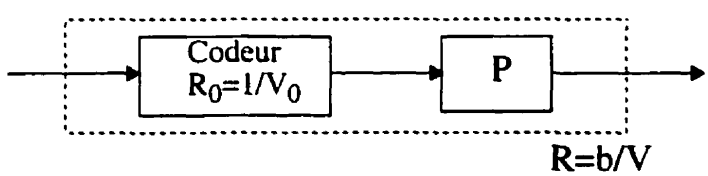

Figure 3.1 Schéma de principe d'un codeur convolutionnel perforé de taux élevé (R=b/V)

Le code perforé ainsi obtenu, dépend à la fois du code d'origine et du patron de perforation utilisé. Celui-ci est spécifié à partir d'une matrice de perforation contenant b colonnes qui représentent chacune une branche du code d'origine, et Vo rangées. Cette matrice est constituée d'éléments binaires représentant des 0 et des 1. Les 1 indiquent les positions des symboles codés à transmettre et les 0 les positions des symboles codés à perforer, c'est-à-dire à ne pas transmettre. Dans ce cas, b branches consécutives du treillis du code d'origine de taux $R_0=1/V_0$ correspondent à une seule branche du code perforé de taux R=b/V, et ce en éliminant (bV_0 - V) symboles du code d'origine.

Prenons à titre d'exemple, le code de taux de codage R_0=1/2 de la figure 2.3 comme code d'origine et la matrice de perforation suivante: $P = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$. Le code perforé ainsi résultant de cette opération sera de taux R=3/4. D'après les positions des 1 et des 0 dans la matrice de perforation, on peut conclure que les 2 symboles codés de la première branche du code d'origine ainsi que le premier et le deuxième symbole de la deuxième et de la troisième branches respectivement seront les symboles à préserver. Tous les autres symboles sont éliminés par perforation. La figure 3.2 illustre la règle de perforation des symboles codés selon la matrice de perforation P et le codeur d'origine de la figure 2.3. Les positions des symboles perforés sont notées par x.

La figure 3.2 montre qu'après chaque ensemble de 6 symboles codés, il n'en reste que 4 à transmettre après perforation. Étant donné que le taux de codage du code d'origine vaut R_0=1/2, chaque ensemble de 3 bits d'informations génère 4 symboles codés qui seront transmis. Par conséquent le taux de codage résultant est R=3/4.

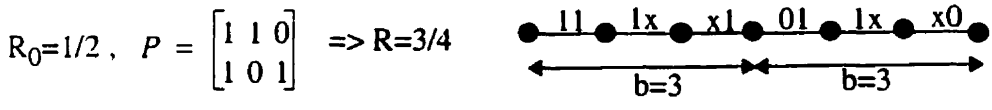

Figure 3.2 Exemple de la perforation selon la matrice de perforation P

Le principe de fonctionnement du codeur perforé de taux R=3/4 est schématisé à la figure 3.3. $\underline{X}^{(U)}$ est la séquence de symboles codés et $\underline{X}_1^{(U)}$ la séquence de symboles perforés. On remarque qu'à la toute fin de l'opération d'encodage, les 6 bits à l'entrée du codeur ont généré 8 symboles codés qui seront transmis dans le canal. Le taux de codage auquel on transmet est alors : R= 6/8 = 3/4.

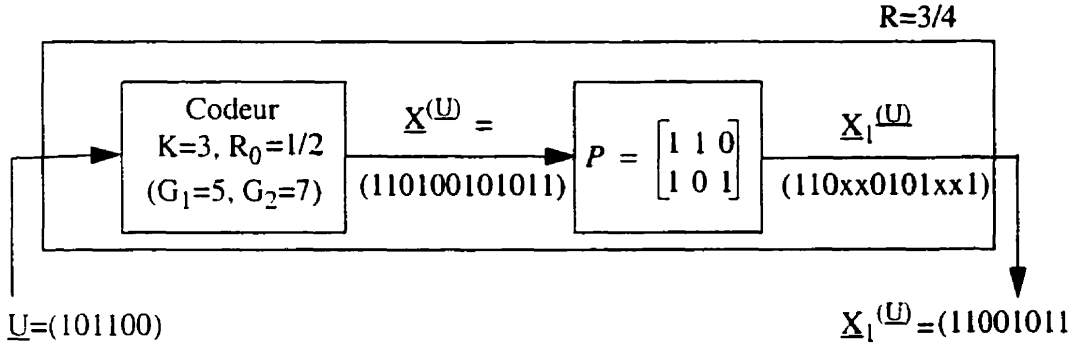

Figure 3.3 Fonctionnement d'un codeur perforé de taux R=3/4

Il est évident qu'un même code de faible taux $R_0=1/V_0$ peut générer par simple changement au niveau du patron de perforation plusieurs codes perforés de taux élevé R=b/V . On parle alors du codage à taux variable. Cette propriété qui permet d'obtenir des taux de codage différents en préservant la structure du codeur, rendent les codes perforés très pratiques.

Les codes perforés peuvent être définis à partir de deux hypothèses différentes [18] qui dicteront la manière d'utiliser ces codes en pratique. La première hypothèse considère les codes perforés comme des vrais codes de taux élevés (R=b/V), alors que pour la deuxième, les codes perforés sont des codes de faibles taux mais dont la redondance est réduite. C'est cette dernière hypothèse que l'on retiendra dans le cadre de ce projet.

Décodage

Selon la deuxième hypothèse que l'on vient de décrire, on peut affirmer avec certitude que le treillis du code perforé est de la même forme que celui du code d'origine, avec toutefois un certain nombre de symboles codés perforés. Le treillis d'un code perforé est dessiné à

la figure 3.4, où les x représentent les symboles codés perforés.

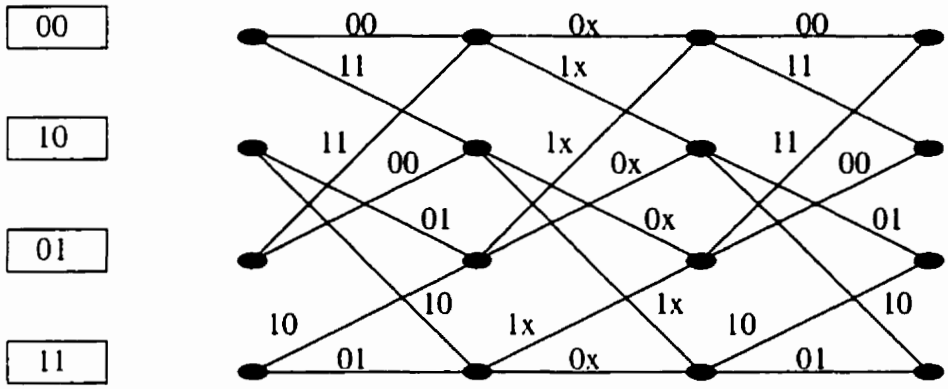

Figure 3.4 Treillis de faible taux d'un code perforé

Cette propriété des codes perforés entraîne un avantage important: il peuvent être décodés, avec des modifications mineures, selon les mêmes algorithmes que les codes d'origine de faible taux.

Les codes convolutionnels perforés ont à l'origine été proposés pour le décodage de Viterbi [7],[25]. Par la suite, ces mêmes codes ont été utilisés dans le décodage séquentiel [3]. L'emploie des codes perforés avec le décodage de Viterbi est l'approche la plus souvent utilisée dans le domaine du codage. Étant donné que cette approche fera l'objet de notre étude, il serait alors utile à ce stade de décrire les principes qui régissent le fonctionnement de l'algorithme de Viterbi pour les codes perforés de taux R=b/V:

- À l'entrée du décodeur, les positions des bits perforés sont remplies par des valeurs non-significatives selon le patron de perforation utilisé
- À chaque niveau du treillis, on compare les métriques des deux chemins convergeant à un état donné afin de trouver le survivant pour chaque état à chaque niveau

du treillis (comme dans l'algorithme de Viterbi pour les codes de taux $R_0=1/V_0$)

- Dans le calcul des métriques les bits perforés sont ignorés, c'est-à-dire que la valeur de la métrique d'une branche est obtenue en tenant compte seulement des bits non-perforés de cette branche

On peut constater que la modification apportée à l'algorithme de Viterbi pour le décodage des codes perforés se produit uniquement au niveau du calcul des métriques, alors que le principe demeure le même. Par conséquent, la complexité du décodeur de type Viterbi pour les codes de taux R=b/V ne croît plus exponentiellement avec b. Cela implique qu'en utilisant un même code d'origine, les changements des taux de codage des codes perforés influencent la complexité du décodeur uniquement en nombre V de symboles à décoder afin de récupérer les b bits transmis. Étant donné qu'à partir d'un même code d'origine on peut obtenir plusieurs taux de codage différents en changeant uniquement les patrons de perforation, il est clair que le concept du codage à taux variable n'implique pas une augmentation significative de la complexité du décodage de Viterbi. C'est cette dernière propriété qui procure aux codes perforés un grand intérêt quant à leur implantation.

3.2. PERFORMANCES DES CODES CONVOLUTIONNELS PERFORÉS

La probabilité d'erreur par bit P_b qui représente le rapport entre le nombre moyen de bits décodés incorrectement et le nombre total de bits d'information transmis est utilisée afin de mesurer les performances du code en question. Ainsi, l'amélioration des performances se traduit par la chute de la probabilité d'erreur par bit. Les performances d'un code convolutionnel dépendent de sa longueur de contrainte K et de son taux de codage R.

Afin d'évaluer rapidement les performances d'un code en particulier, une borne supérieure

sur la probabilité d'erreur a été établie. C'est cette méthode qu'on utilisera pour comparer les performances des codes perforés retenus. Comme on le verra par la suite, pour déterminer la borne union sur la probabilité d'erreur par bit, les éléments du spectre du code en question doivent être connus. Par conséquent, avant d'introduire l'expression de la borne union on s'intéressera à l'évaluation du spectre du code perforé.

3.2.1. *SPECTRES DES CODES CONVOLUTIONNELS PERFORÉS*

D'après la section 2.4, le spectre du code convolutionnel énumère pour tous les chemins qui diffèrent du chemin correct, le nombre total de bits en erreur occasionné que l'on note c_d. Pour déterminer le spectre du code, on doit explorer tous les chemins dans le treillis d'encodage. Quant aux codes perforés, la démarche reste la même, sauf que la recherche se fait maintenant au niveau du treillis du code d'origine [1] ayant, selon le patron de perforation appliqué, certains symboles de branche perforés.

Étant donné que la distance libre du code d_{free} est une des mesures des performances du code, il est nécessaire que celle-ci soit la plus grande possible pour que les mots de code soient facilement différentiables. En d'autres termes, plus la distance libre du code en question est élevée, plus la performance d'erreur du code l'est.

Comme le montrent les tableaux 3.1 et 3.2, le processus de perforation provoque une diminution de la distance libre du code d'origine. Les valeurs du tableau 3.1 représentent le spectre du code convolutionnel avec K=3 et R_0=1/2 (G_1=5, G_2=7) utilisé pour générer le code perforé de taux R=4/5. Les valeurs du tableau 3.2 représentent le spectre de ce dernier code, obtenu en utilisant la matrice de perforation $P = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$.

Tableau 3.1 Spectre du code convolutionnel K=3, R_0=1/2 (G_1=5, G_2=7)

d	a_d	c_d
5	1	1
6	2	4
7	4	12
8	8	32
9	16	80
10	32	192
11	64	448
12	128	1024

Tableau 3.2 Spectre du code perforé du taux R=4/5 obtenu à partir du code K=3, R_0=1/2 (G_1=5, G_2=7)

d	a_d	c_d
2	1	1
3	12	36
4	53	309
5	237	2058
6	1035	12031
7	4508	65754
8	19651	344656
9	85659	1755310

La première raie du spectre correspondant à la distance libre du code, on constate que le d_{free} du code d'origine (d_{free}=5) a diminué après la perforation et que, pour le code perforé de taux R=4/5 elle est égale à 2. Ceci n'est pas surprenant lorsqu'on constate que la perforation a entraîné une augmentation du taux de codage et par le fait même, une diminution

de la redondance ajoutée aux bits d'information. Cela va donc se traduire par une décroissance des performances du code perforé par rapport au code d'origine. Bégin et Haccoun [2] ont montré que la distance libre d'un code perforé est bornée par :

$$(d_{free})_{perforé} \leq \frac{1}{b}(d_{free})_{origine} \tag{3.1}$$

L'approche intuitive basée sur l'hypothèse que les bons codes convolutionnels génèrent les bons codes perforés a conduit à trouver des codes perforés ayant des bonnes performances. Par conséquant, lors de la recherche des bons codes perforés, il est plus avantageux de choisir des codes convolutionnels ayant des distances libres les plus élevées possibles comme étant les codes d'origine.

3.2.2. *BORNE SUPÉRIEURE SUR LA PROBABILITÉ D'ERREUR PAR BIT*

Pour un code particulier, il est possible de borner la probabilité d'erreur par bit en utilisant le spectre du code, ou plus précisément en connaissant la distance libre du code et les coefficients c_d. En prenant la borne union, la borne supérieure sur la probabilité d'erreur par bit pour un code de taux R=b/V est donnée par [23]:

$$P_b < \frac{1}{b}\sum_{d=d_{free}}^{\infty} c_d P_d \tag{3.2}$$

où P_d représente la probabilité de décoder une séquence erronée se trouvant à la distance d de la séquence transmise. Pour le canal binaire symétrique de probabilité de transition p, cette probabilité est donnée par :

$$P_d = \sum_{i=\frac{d+1}{2}}^{d} \binom{d}{i} p^i (1-p)^{d-i} \quad \text{, si d est impair, et} \tag{3.3a}$$

$$P_d = \sum_{i=\frac{d}{2}+1}^{d} \binom{d}{i} p^i (1-p)^{d-i} + \frac{1}{2}\binom{d}{d/2} p^{d/2} (1-p)^{d/2} \text{ , si d est pair} \tag{3.3b}$$

Dans le cas d'un canal gaussien et une modulation PSK, la valeur P_d devient :

$$P_d = Q\left(\sqrt{\frac{2dRE_b}{N_0}}\right) \tag{3.4}$$

avec:

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{-t^2/2} dt \tag{3.5}$$

où E_b est l'énergie d'un bit d'information transmis, $N_0/2$ est la variance du bruit blanc gaussien additif et R le taux de codage. L'équation (3.4) peut être exprimée en fonction de l'énergie du signal correspondant à un symbole codé (Es) en tenant compte de la relation : $E_s = RE_b$. Dans le calcul des performances des codes convolutionnels, c'est la valeur de l'énergie par bit qui sera utilisée.

3.3 CODES PERFORÉS À DES TAUX DE CODAGE COMPATIBLES

Rappelons que le principe du codage à taux variable peut être illustré par l'ensemble de

codes de taux élevés obtenu à partir du même code d'origine de faible taux, auquel on applique les changements au niveau du patron de perforation. On arrive alors à la définition des codes perforés à des taux de codage dits compatibles, que l'on peut décrire comme étant des codes à taux variables auxquels on a ajouté la propriété de compatibilité suivante: la forme du patron de perforation du code R=b/V demeure inchangée pour tous les codes dont le taux de codage est supérieur à b/V. Une illustration de la compatibilité des codes perforés est représentée à la figure 3.5. Le taux de codage du code d'origine dans cet exemple est R_0=1/2 et les codes perforés résultants sont des taux de codage croissants et compatibles, allant de R=2/3 à R=4/5.

Comme on peut le constater d'après la figure 3.5, les trois codes perforés obtenus sont des codes dont les taux de codage sont compatibles. Ceci se manifeste au niveau des patrons de perforations correspondants comme suit : parmi les éléments de la matrice de perforation P_i on retrouve la matrice de perforation P_{i-1} (les éléments de cette dernière sont encadrés en pointillé à la figure 3.5).

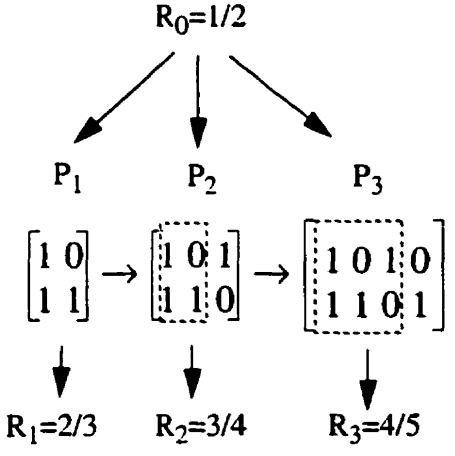

Figure 3.5 Exemple des matrices de perforation des codes perforés compatibles

3.4. RECHERCHE DE BONS CODES PERFORÉS À DES TAUX COMPATIBLES OBTENUS À PARTIR DE CODE D'ORIGINE DE TAUX R_0=1/6

À partir du meilleur code connu présenté par Daut en 1982, dont le taux de codage est R_0=1/6, nous avons construit un ensemble de codes convolutionnels perforés ayant les taux de codage croissants et compatibles allant de 1/6 à 7/8. Les longeurs de contraintes utilisées sont K=7 et K=8.

Le calcul des spectres et des performances (la probabilité d'erreur par bit en fonction du rapport signal/bruit) de tous ces codes nous a permis de déterminer le meilleur code pour chaque taux de codage, et de le comparer pour le cas d'un canal binaire symétrique avec le meilleur code perforé connu [11] du même taux de codage.

Comme on l'a expliqué précédemment, la compatibilité des codes signifie que la forme du patron de perforation du code R=b/V demeure inchangée pour tous les codes dont le taux de codage est R>b/V. Plus précisément, pour le taux de codage d'origine R_0=1/6, nous distinguons deux cas selon la forme du taux de codage R obtenu:

1) $$R = \frac{1}{m} \quad , \quad 2 \leq m \leq 6$$

Règle de perforation: afin d'obtenir la matrice de perforation du code R=1/m, nous préservons la position des bits qui ont été perforés lors de l'obtention du code R=1/(m-1), et nous perforons un symbole de plus. Cette règle est donc applicable pour les codes perforés ayant des taux de codage croissants allant de 1/6 à 1/2. Un exemple d'application de la règle citée ci-dessus est illustré à la figure 3.6. Comme on peut le constater à la figure 3.6, les positions des symboles perforés (les zéros) de la matrice P_i sont préservés dans la

matrice P_{i+1}. La matrice P_1 correspond au code d'origine, c'est-à-dire que aucun des symboles codés n'a été perforés.

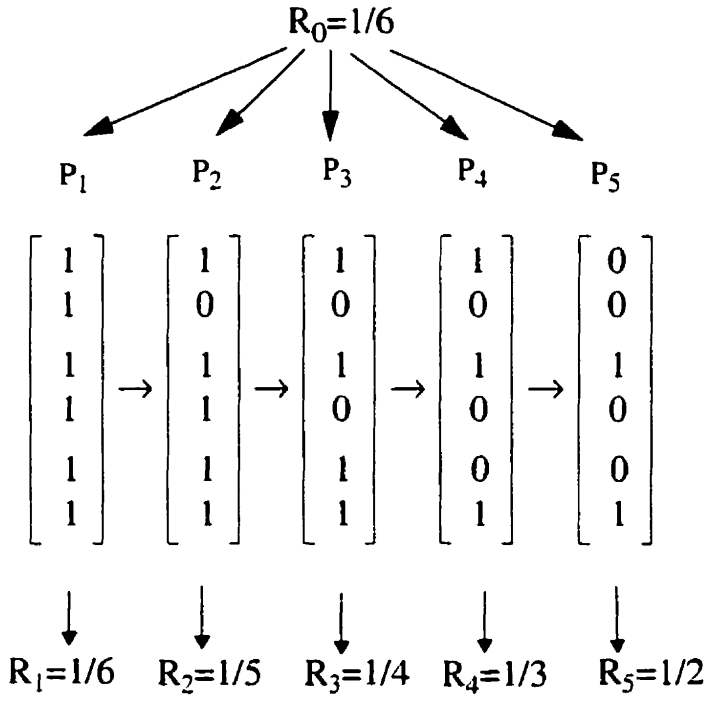

Figure 3.6 Règle de perforation pour les codes du taux de la forme R=1/m

2) $R = \frac{m-1}{m}$, $2 \le m \le 8$

Une fois le meilleur code de taux R=1/2 déterminé, et ce à partir du code d'origine de taux R_0=1/6, on voudrait obtenir des taux de codage croissants et supérieurs à R=1/2: R={2/3, 3/4, 4/5, 5/6, 6/7, 7/8}. Par conséquent, il est nécessaire d'adopter à nouveau, une règle de perforation pour préserver la propriété de compatibilité des codes perforés de taux supérieurs à R=1/2. Pour cette fin, nous avons utilisé les deux approches suivantes:

i) La position des symboles perforés lors de l'obtention du code R=1/2 est identique pour toutes les colonnes de la matrice de perforation (voir la figure 3.7); un symbole

de plus est perforé dans chacune des colonnes ajoutées.

ii) Pour obtenir le code R=(m-1)/m, nous nous servons du patron de perforation du code R=(m-2)/(m-1) et nous ajoutons une colonne de plus, dans lequelle une seule valeur vaut 1 (voir la figure 3.7).

Quelque soit l'approche utilisée, une colonne supplémentaire ayant un symbole non-perforé est ajoutée dans la matrice de perforation, afin d'obtenir un taux de codage plus élevé. C'est la règle d'ajout des colonnes qui change d'une approche à l'autre. Selon celle que l'on choisi, le nombre de matrices de perforation possibles est différent lors du passage du taux R=(m-2)/(m-1) au taux R=(m-1)/m : l'approche i) offre deux patrons de perforation possibles alors que l'approche ii) en offre six.

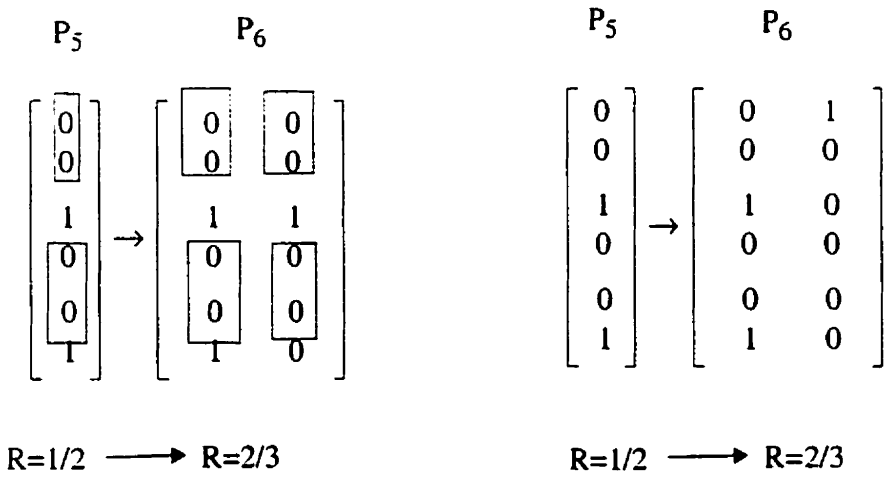

Figure 3.7 Exemples des patrons de perforation selon l'approche i) et ii) respectivement

Tout au long de notre recherche des bons codes perforés de taux de codage croissants et compatibles, on doit s'assurer que les codes obtenus ne sont pas catastrophiques. Il faut noter qu'à partir d'un code d'origine non-catastrophique on peut arriver, selon la matrice

de perforation utilisée, à un code perforé qui est catastrophique [16]. D'autre part, une fois le code catastrophique perforé, il génère un autre code catastrophique. Ainsi, il faut s'assurer que les codes obtenus, et utilisés pour générer les codes de taux plus élevés ne sont pas catastrophiques.

3.4.1 *RÉSULTATS*

L'ensemble des codes de taux croissants et compatibles allant de R=1/6 à R=7/8 est obtenu à partir du code d'origine de taux de codage R_0=1/6, pour des longueurs de contraintes K=7 et K=8.

Pour chacun des taux de codage désiré, les spectres de tous les codes possibles sont calculés et utilisés afin d'obtenir les performances des codes en question; en comparant les performances (probabilité d'erreur par bit en fonction du rapport signal/bruit) des codes du même taux de codage, on détermine le meilleur code, qui sera le «code d'origine» pour les taux de codage plus élevés.

On est en mesure de comparer ces codes et de calculer leur nombre en fonction de l'approche de perforation utilisée, afin de déterminer l'ensemble des codes de taux croissants et compatibles allant de R=1/6 à R=7/8 . Pour chacun des taux de codage de type R=1/m (m allant de 2 à 5), le nombre de codes est égal à (m+1): pour le taux de codage R=1/5, on choisit une position parmi six à perforer, pour R=1/4 une position à perforer parmi cinq est choisie, car la position du symbole perforé précédemment est préservée, etc. Ainsi, afin d'atteindre le taux de codage R=1/2 à partir de R_0=1/6, le nombre total de codes à comparer est égal à 18. Pour les codes de taux de codage supérieur à R=1/2, ce nombre, comme on l'a vu précédemment, varie en fonction de l'approche utilisée comme suit: l'ap-

proche i) offre deux possibilités pour chacun des taux de codage allant de R=2/3 à R=7/8, ce qui donne un nombre total de 12 codes à comparer, alors que l'approche ii) en offre six, ce qui donne un nombre total de 36 codes à comparer pour le même ensemble de taux de codage. Par conséquent, pour les taux de codage croissants et compatibles allant de R=1/6 à R=7/8, on arrive à un nombre total de 30 codes à comparer utilisant l'approche i), et 54 codes avec l'approche ii).

Il faut noter que le calcul effectué est exact en supposant que tous les codes obtenus sont non-catastrophiques; ceci n'est pas toujours le cas. Dans cette situation, le calcul nous donnera le nombre maximal de codes à comparer parmi lesquels, les codes que l'on trouvera comme étant catastrophiques, seront ignorés (c'est-à-dire on ne calcule ni leurs spectres ni leurs performances).

3.4.1.1. *RÉSULTATS OBTENUS AVEC K=7*

L'ensemble des codes de taux croissants et compatibles allant de R=1/6 à R=7/8 est obtenu à partir du meilleur code de longueur de contrainte K=7 et de taux de codage R_0=1/6 ; les générateurs employés sont: G_1=173, G_2=151, G_3=135, G_4=135, G_5=163 et G_6=137. Par l'évaluation et la comparaison des performances de tous les codes possibles obtenus pour les taux de codage allant de R=1/5 à R=1/2, on est arrivé à l'ensemble des meilleurs codes compatibles avec leurs patrons de perforation respectifs. Les matrices de perforation correspondants aux meilleurs codes sont illustrées à la figure 3.8. Les performances de ces derniers sont présentées à la figure 3.9 pour un canal binaire symétrique.

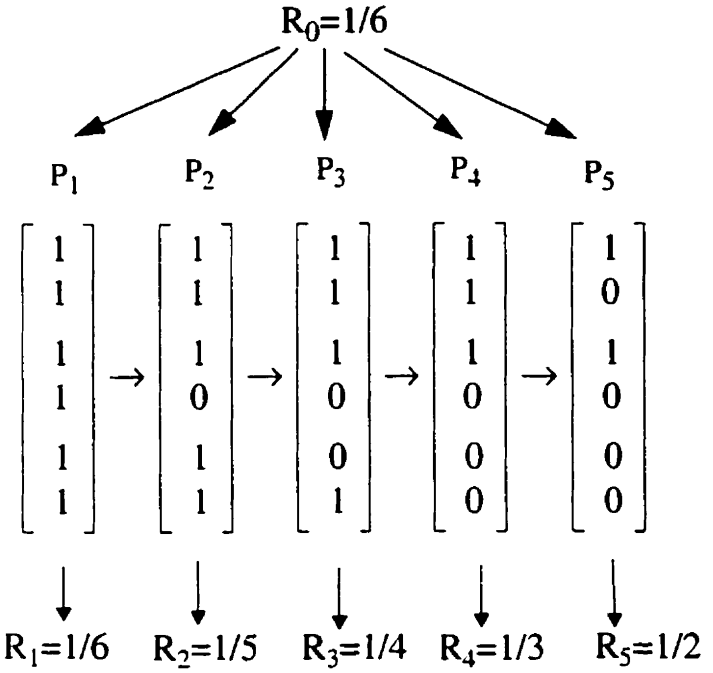

Figure 3.8 Matrices de perforation des meilleurs codes perforés à taux de codage croissants et compatibles allant de R=1/6 à R=1/2; Code d'origine K=7, R_0=1/6

RÉSULTATS AVEC L'APPROCHE i)

Les résultats obtenus pour les taux de codage supérieurs à 1/2 selon l'approche i) sont représentés aux figures 3.10 et 3.12: la figure 3.10 montre les matrices de perforation des meilleurs codes obtenus, et la figure 3.12 représente les performances de ces codes pour un canal binaire symétrique (CBS).

RÉSULTATS AVEC L'APPROCHE ii)

Les résultats obtenus selon l'approche ii) pour les taux de codage supérieurs à 1/2 sont représentés aux figures 3.11 et 3.13: la figure 3.11 montre les matrices de perforation des meilleurs codes obtenus, et la figure 3.13 représente les performances de ces codes pour le canal binaire symétrique.

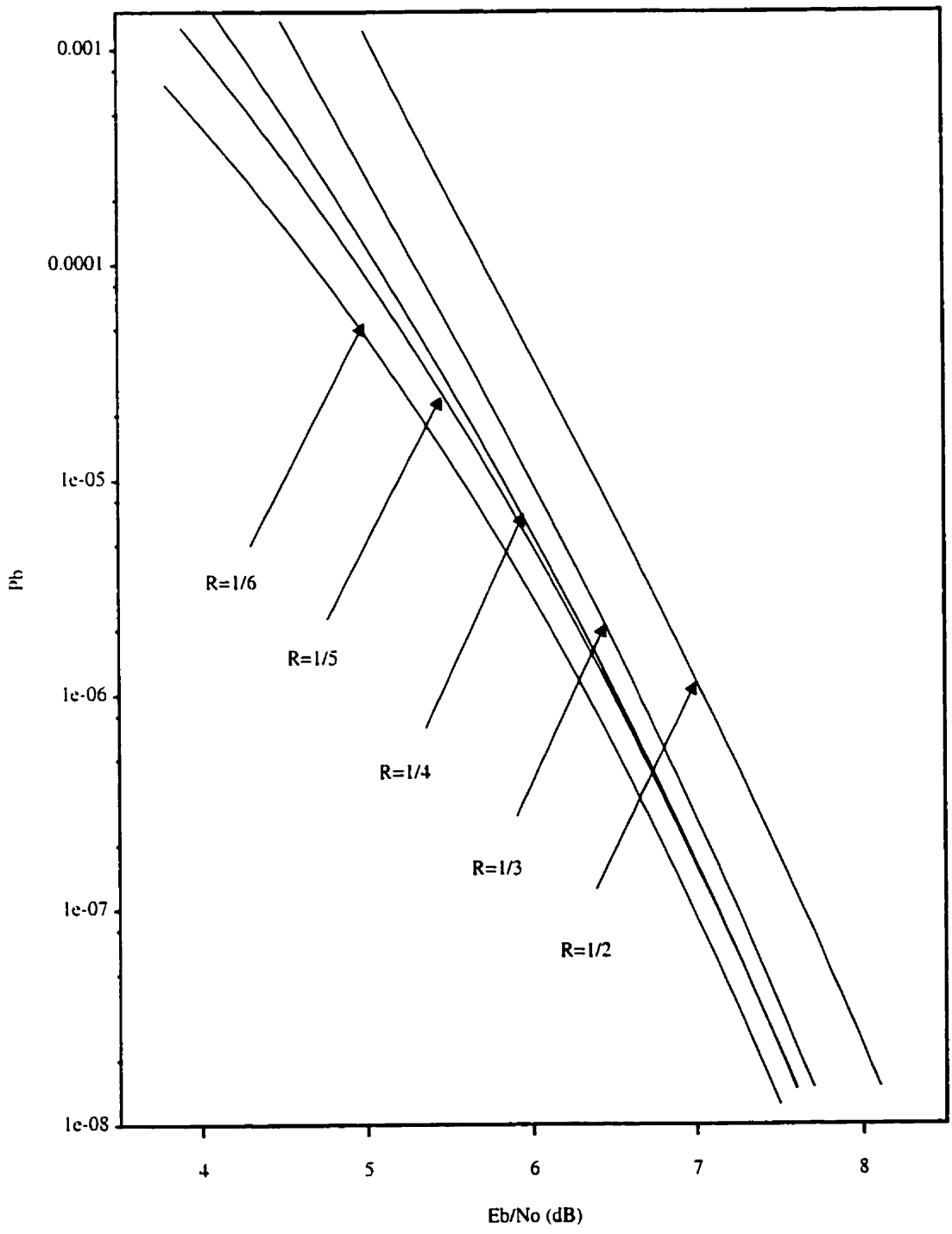

Figure 3.9 Borne sur la probabilité d'erreur par bit pour le CBS pour les codes à taux croissants et compatibles allant de R=1/6 à R=1/2; Code d'origine: K=7, R_0=1/6

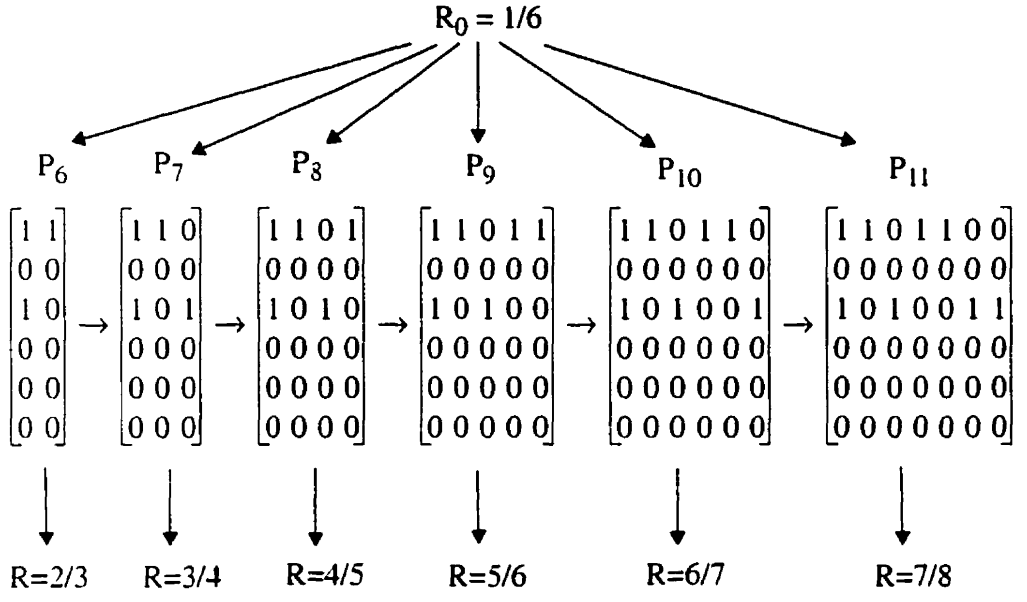

Figure 3.10 Matrices de perforation des codes perforés à taux de codage croissants et compatibles allant de R=2/3 à R=7/8; Code d'origine K=7, R_0=1/6 ; Approche i)

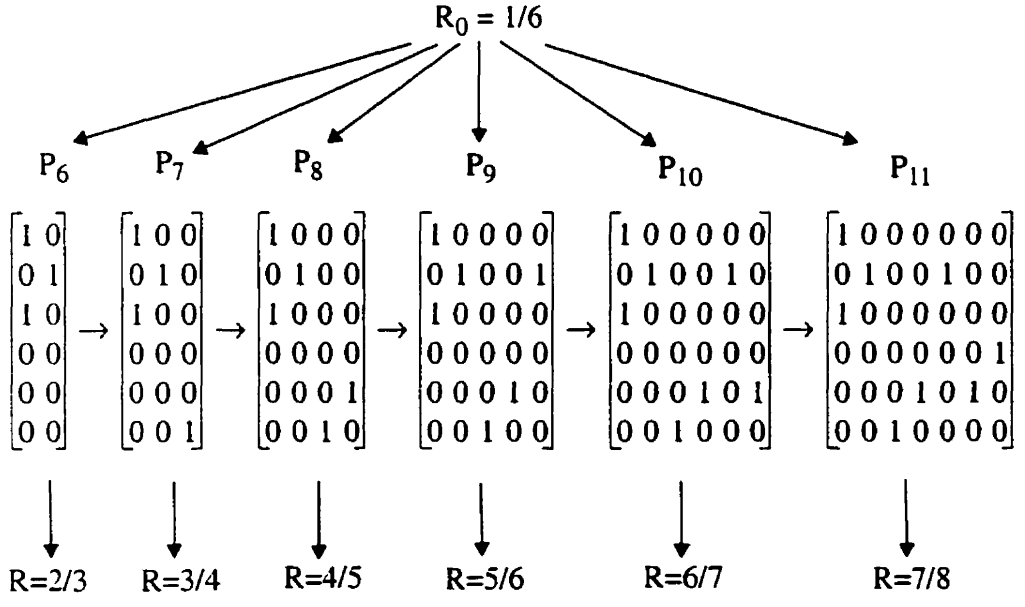

Figure 3.11 Matrices de perforation des codes perforés à taux de codage croissants et compatibles allant de R=2/3 à R=7/8; Code d'origine: K=7, R_0=1/6; Approche ii)

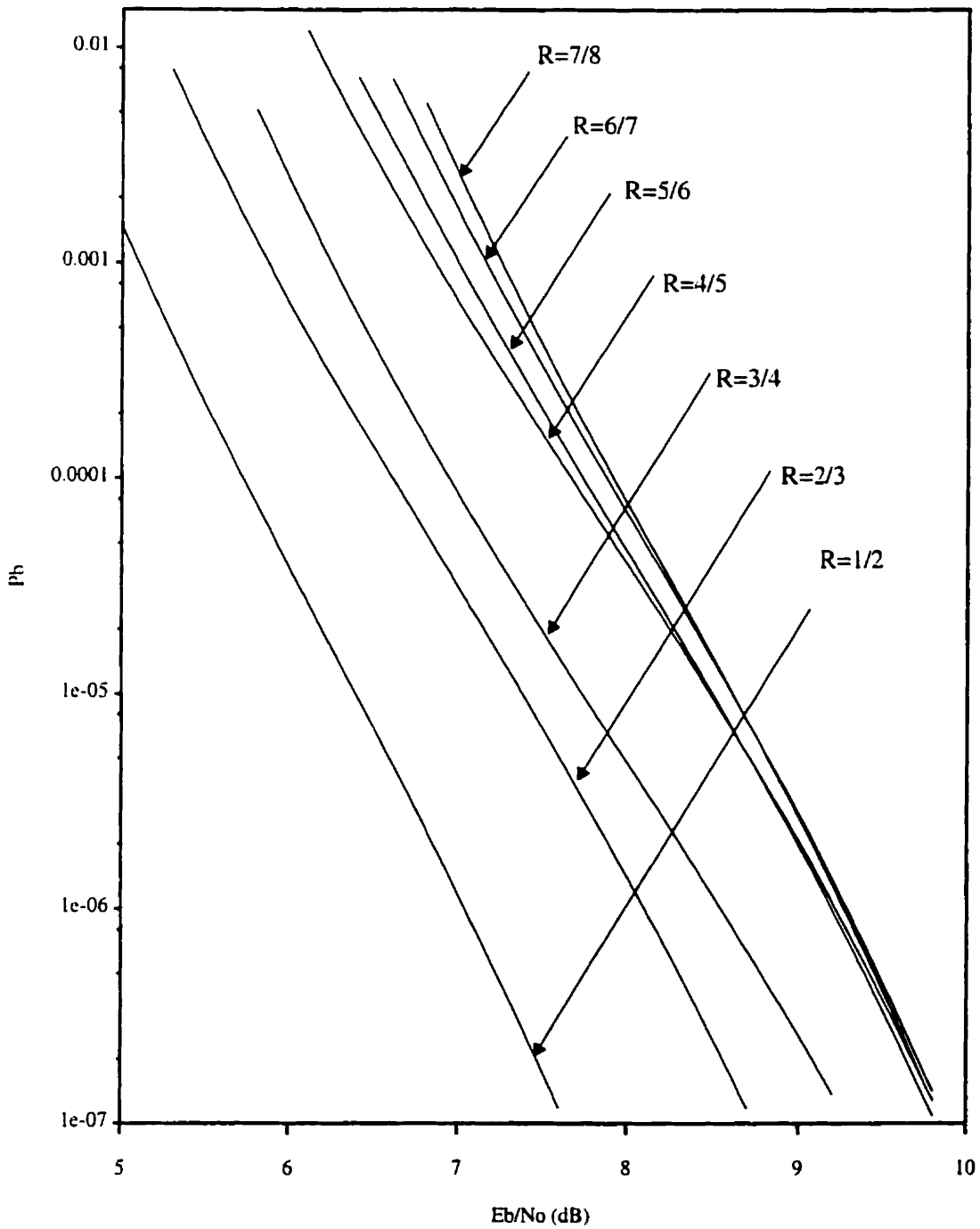

Figure 3.12 Borne sur la probabilité d'erreur par bit pour le CBS pour les codes à taux croissants et compatibles allant de R=1/2 à R=7/8; Code d'origine: K=7, R_0=1/6; Approche i)

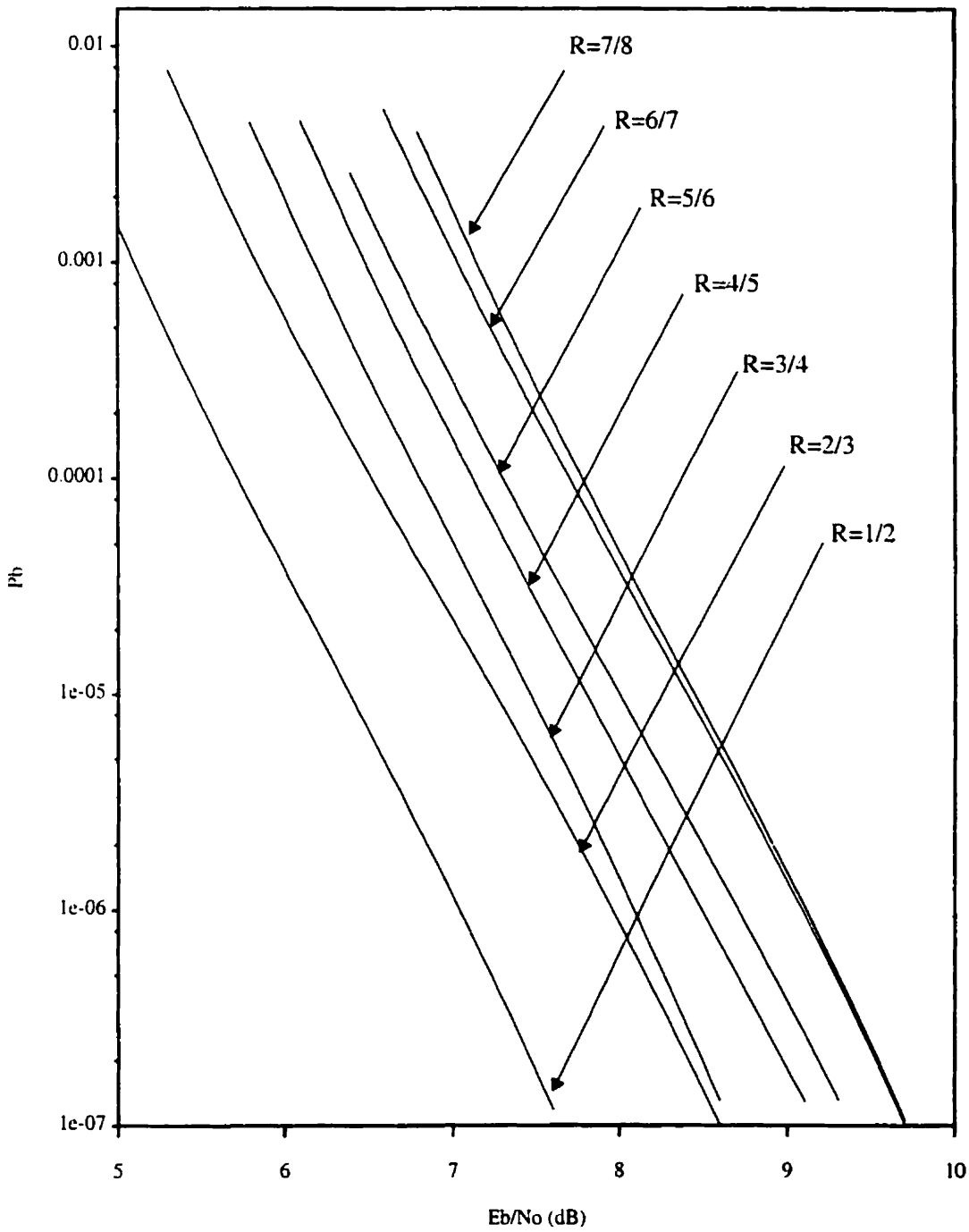

Figure 3.13 Borne sur la probabilité d'erreur par bit pour le CBS pour les codes à taux croissants et compatibles allant de R=1/2 à R=7/8; Code d'origine: K=7, R_0=1/6; Approche ii)

Aux figures 3.12 et 3.13 nous avons représenté les performances d'erreur obtenues selon les approches i) et ii) respectivement, décrites plus haut. En comparant ces résultats, il apparait que l'approche ii) permet d'obtenir des performances légèrement supérieures: le gain en puissance est d'environ 0.15 dB pour un Pb égale à 10^{-5}. Ceci était en fait prévisible, puisque lors de la recherche dans la deuxième approche, le choix du meilleur code de taux b/V se fait parmi un nombre de candidats plus grand que dans la première approche. Donc, les codes perforés obtenus selon l'approche ii) seront considérés comme étant les meilleurs, et leurs performances seront comparées aux performances des meilleurs codes perforés connus obtenus en [11].

3.4.1.2. *RÉSULTATS OBTENUS AVEC K=8*

L'ensemble des codes de taux croissants et compatibles allant de R=1/6 à R=7/8 est obtenu à partir du meilleur code de longueur de contrainte K=8 et de taux de codage R_0=1/6 ; les générateurs employés sont: G_1=253, G_2=375, G_3=331, G_4=235, G_5=313 et G_6=357.

Les matrices de perforation correspondant aux meilleurs codes de taux allant de R=1/6 à R=1/2 sont illustrées à la figure 3.14 et les performances de ces codes sont représentées à la figure 3.15 pour le CBS .

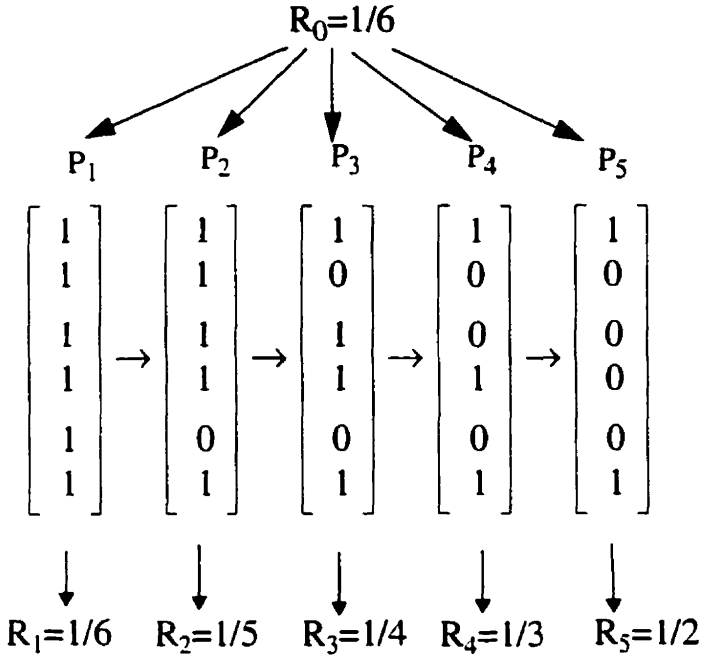

Figure 3.14 Matrices de perforation des meilleurs codes perforés à taux de codage croissants et compatibles allant de R=1/6 à R=1/2; Code d'origine: K=8, R_0=1/6

RÉSULTATS AVEC L'APPROCHE i)

Les résultats obtenus selon l'approche i) pour les taux de codage supérieurs à 1/2 sont représentés aux figures 3.16 et 3.18: la figure 3.16 illustre les matrices de perforation des meilleurs codes obtenus, et la figure 3.18 représente les performances de ces codes pour le CBS.

RÉSULTATS AVEC L'APPROCHE ii)

Les résultats obtenus pour les taux de codage supérieurs à 1/2 selon l'approche ii) sont représentés aux figures 3.17 et 3.19: la figure 3.17 illustre les matrices de perforation des meilleurs codes obtenus, et la figure 3.19 représente les performances de ces codes pour le CBS.

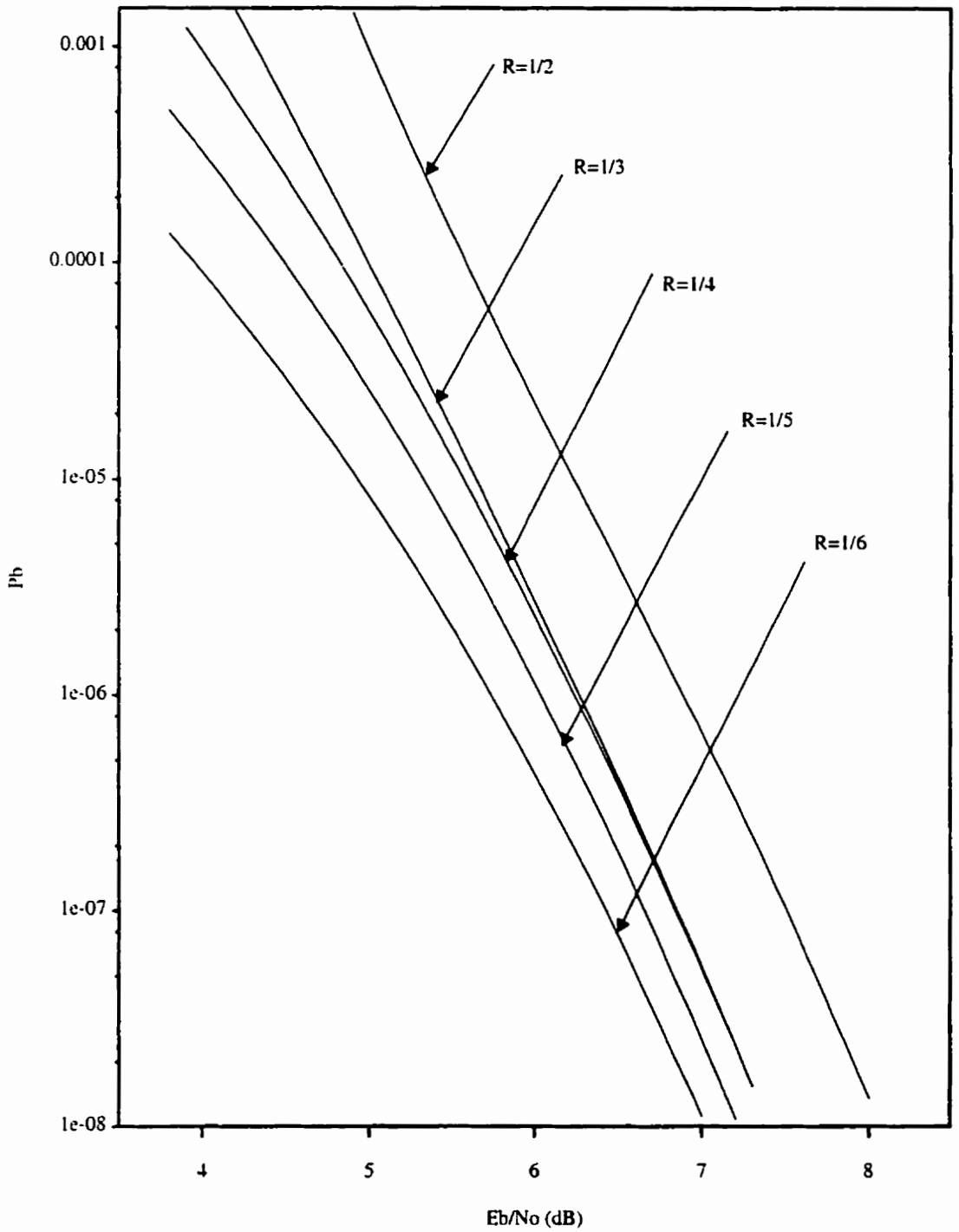

Figure 3.15 Borne sur la probabilité d'erreur par bit pour le CBS pour les codes à taux croissants et compatibles allant de R=1/6 à R=1/2; Code d'origine: K=8, R_0=1/6

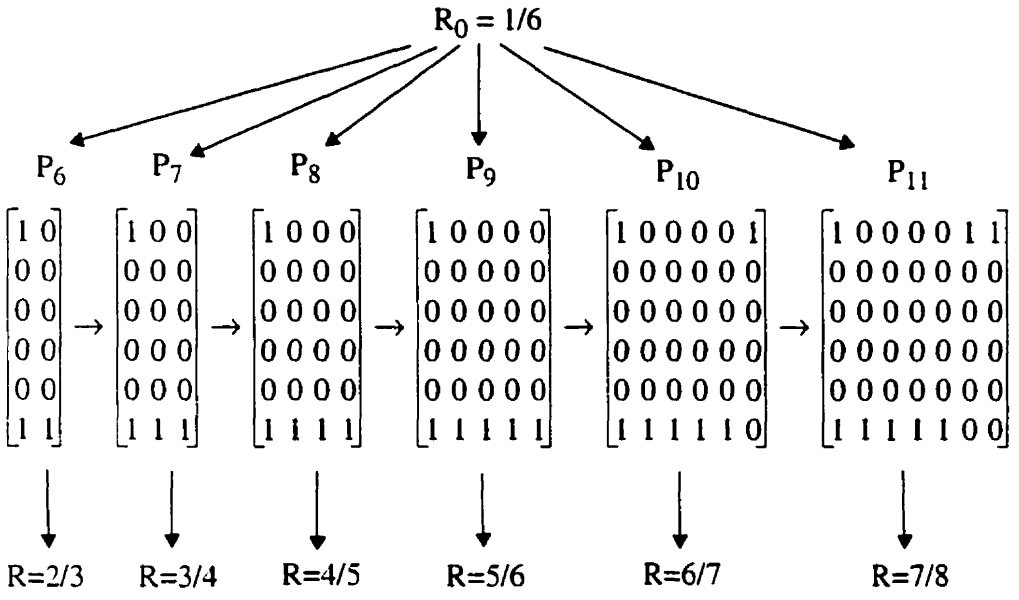

Figure 3.16 Matrices de perforation des codes perforés à taux de codage croissants et compatibles allant de R=2/3 à R=7/8; Code d'origine: K=8, R_0=1/6 ; Approche i)

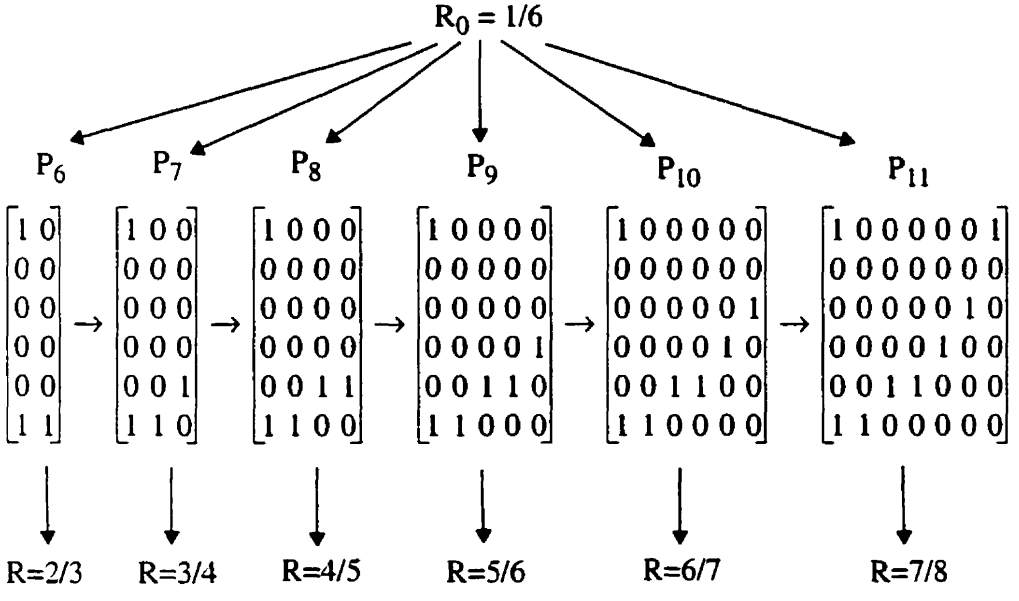

Figure 3.17 Matrices de perforation des codes perforés à taux de codage croissants et compatibles allant de R=2/3 à R=7/8; Code d'origine: K=8, R_0=1/6 ; Approche ii)

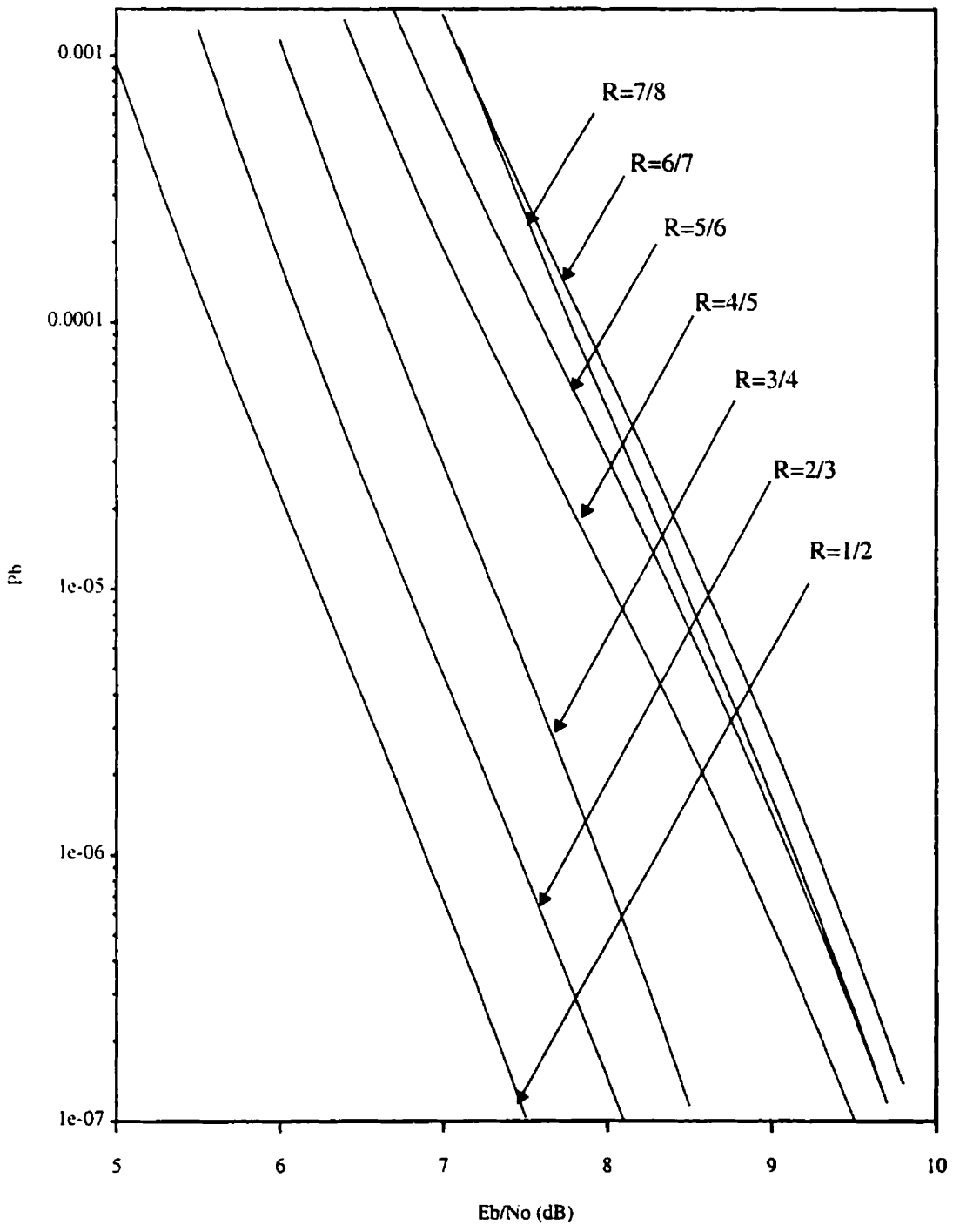

Figure 3.18 Borne sur la probabilité d'erreur par bit pour le CBS pour les codes à taux croissants et compatibles allant de R=1/2 à R=7/8; Code d'origine: K=8, R_0=1/6; Approche i)

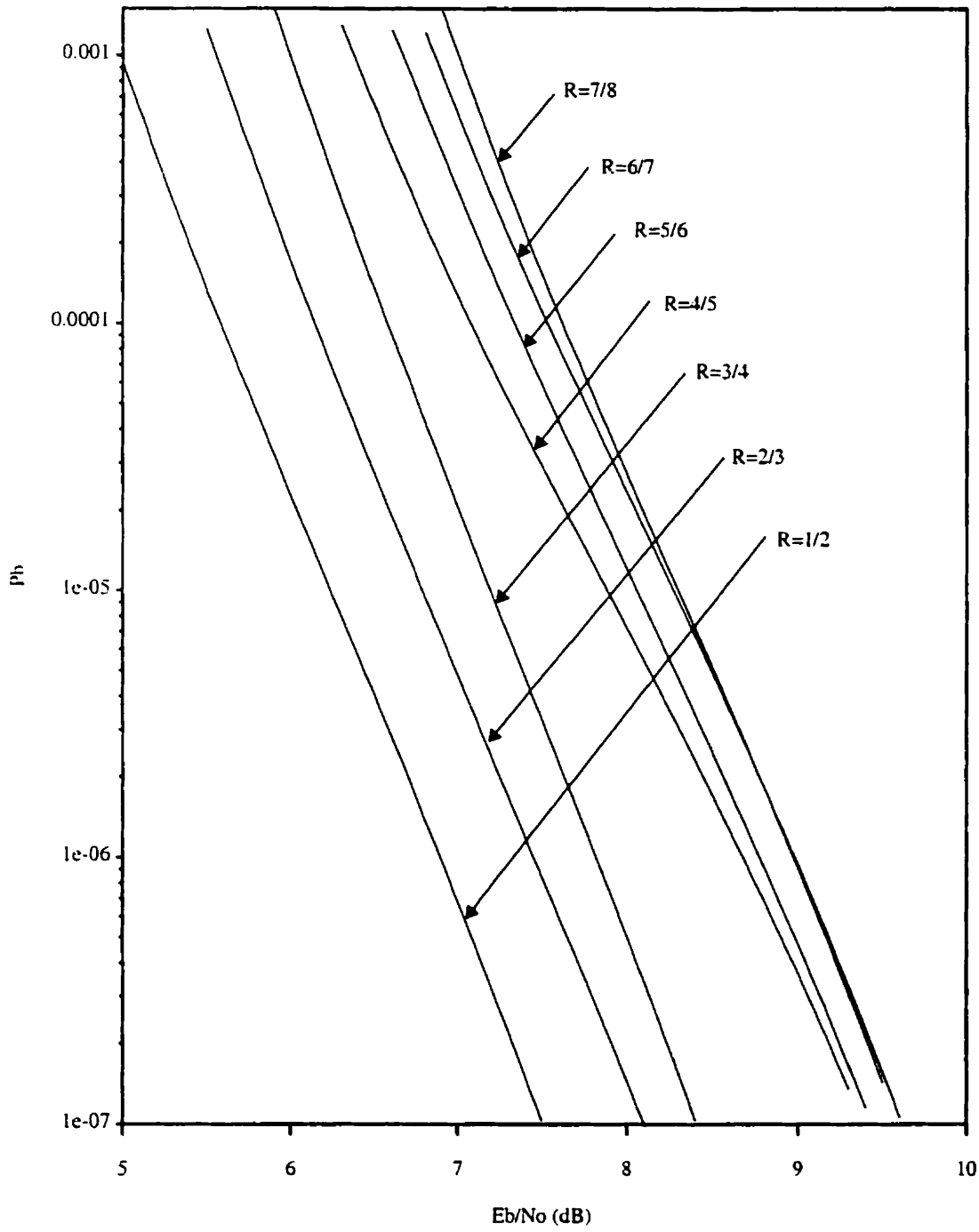

Figure 3.19 Borne sur la probabilité d'erreur par bit pour le CBS pour les codes à taux croissants et compatibles allant de R=1/2 à R=7/8; Code d'origine: K=8, R_0=1/6; Approche ii)

En comparant les résultats représentés sur les figures 3.16 et 3.18, on constate une légère amélioration des performances lorsque l'approche ii) est utilisée. Le gain que l'approche ii) apporte par par rapport à l'approche i), pour l'ensemble de codes de taux allant de 1/2 à 7/8, est d'environ 0.15 dB. Comme pour le cas K=7, les codes perforés obtenus par l'approche ii) sont considérés comme étant les meilleurs.

3.4.1.3 COMPARAISON À D'AUTRE CODES PERFORÉS CONNUS

Les bornes sur la probabilité d'erreur par bit des meilleurs codes perforés souvent cités dans la littérature [11] et obtenus à partir de code d'origine de taux R_0=1/2, sont représentées aux figures 3.20 et 3.21 pour K=7 et K=8 respectivement. Les générateurs emloyés sont : G_1=133, G_2=171 pour K=7 et G_1=247, G_2=371 pour K=8. Les performances théoriques sont toujours calculées en considérant un canal binaire symétrique en présence du bruit blanc gaussien.

À partir du code d'origine de taux R_0=1/2, la matrice de perforation donnant les meilleurs performances est déterminée pour chacun des taux de codage désirés. En d'autres termes, pour trouver le meilleur patron de perforation pour un taux de codage de type R=(m-1)/m et ce à partir du code d'origine de taux R_0=1/2, le nombre de codes à comparer devient : $\binom{2(m-1)}{m}$. Cela veut dire par exemple que pour obtenir le meilleur code perforé de taux R=4/5, il faut comparer les performances de $\binom{8}{5} = 54$ codes. Or ce nombre est largement supérieur à celui des codes compatibles où l'on avait comparé 54 codes (selon l'approche plus exigeante) afin d'obtenir l'ensemble entier des codes perforés de taux allant de 1/6 à 7/8.

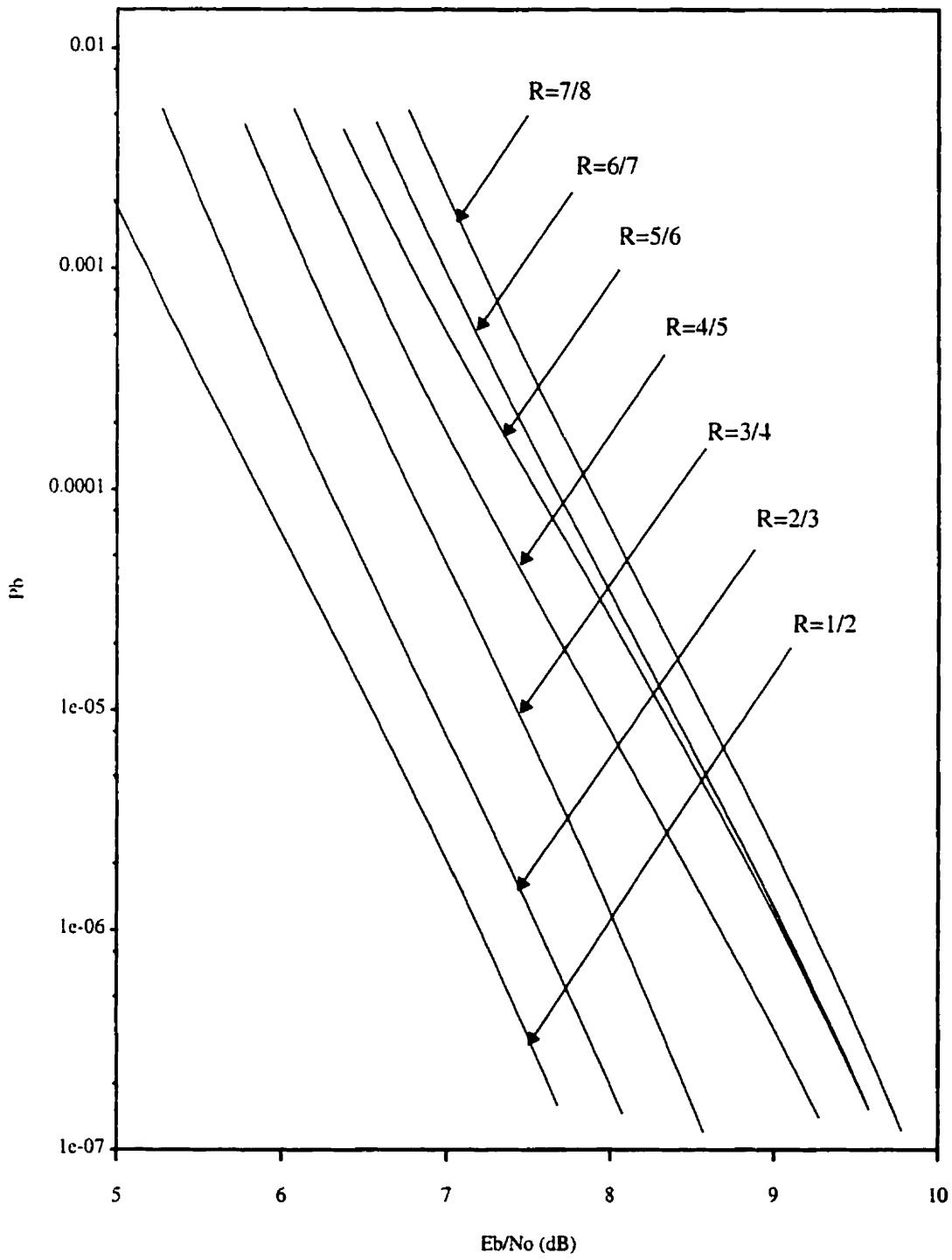

Figure 3.20 Borne sur la probabilité d'erreur par bit des codes perforés obtenus à partir du code d'origine K=7, R_0=1/2 (G_1=133, G_2=171)

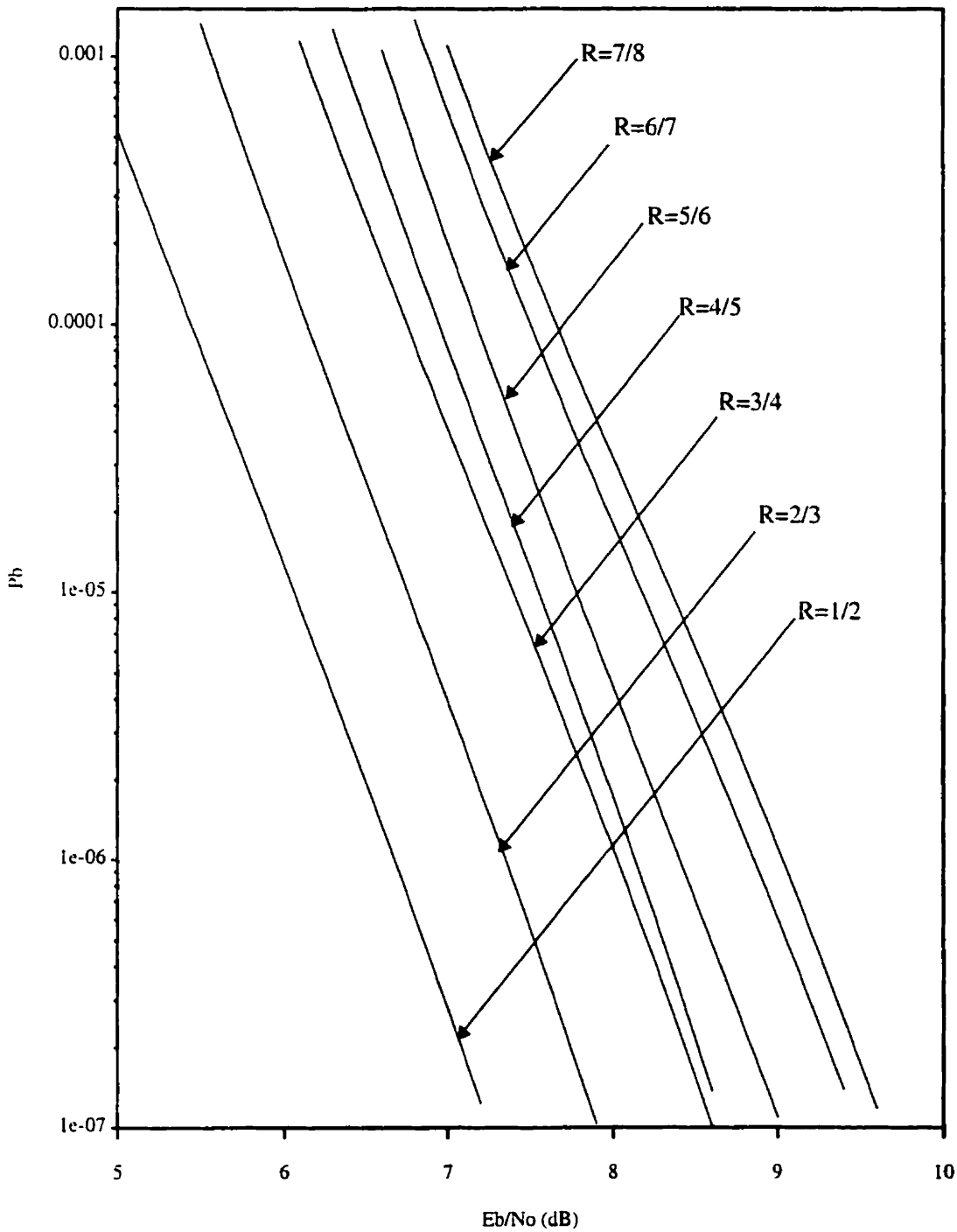

Figure 3.21 Borne sur la probabilité d'erreur par bit des codes perforés obtenus à partir du code d'origine K=8, R_0=1/2 (G_1=247, G_2=371)

La démarche que l'on vient d'exposer dans ce chapitre, c'est-à-dire la recherche d'un ensemble des codes compatibles, n'est pas une méthode de recherche optimale de codes perforés, car le choix des patrons de perforation est restreint par la propriété de compatibilité. Pourtant, les performances des codes perforés obtenus à l'aide de cette méthode sont comparables à celles des codes perforés non-compatibles [11]. Ceci peut être constaté en comparant les performances qu'on a obtenues pour les différents taux de codage avec K=7 (figure 3.13) et K=8 (figure 3.19) aux performances publiées dans [11] pour les mêmes taux de codage (figures 3.20 et 3.21).

On a extrait les courbes de performances correspondant aux taux de codage R=1/2 et R=7/8, illustrées par les figures 3.22 et 3.23 pour K=7 et K=8 respectivement.

Comme on peut le constater d'après la figure 3.22, l'ensemble des codes de taux croissants et compatibles qu'on a obtenu pour K=7, est caractérisé par de très bonnes performances: les codes des taux R=1/2 et R=7/8 ainsi obtenus, ont des performances légèrement meilleures que les codes de même taux publiés en [11].
Quant à l'ensemble des codes aux taux croissants et compatibles obtenus pour K=8, les performances de ce dernier sont toujours comparables à celles obtenues en [11]: le code de taux R=1/2 qu'on a obtenu est légèrement moins performant, d'environ 0.25 dB à $P_b=10^{-7}$, à celui de [11], alors que le code de taux R=7/8 obtenu demeure à peine supérieur à celui du même taux de codage de [11] (voir la figure 3.23).

Il est cependant important de mentionner qu'une étude similaire (pour K=7 et K=8) sur les taux de codage compris entre 1/2 et 7/8 n'offre pas toujours des résultats comparables. Certains de ces taux de codage entrainent des performances inférieures aux résultats publiées en [11].

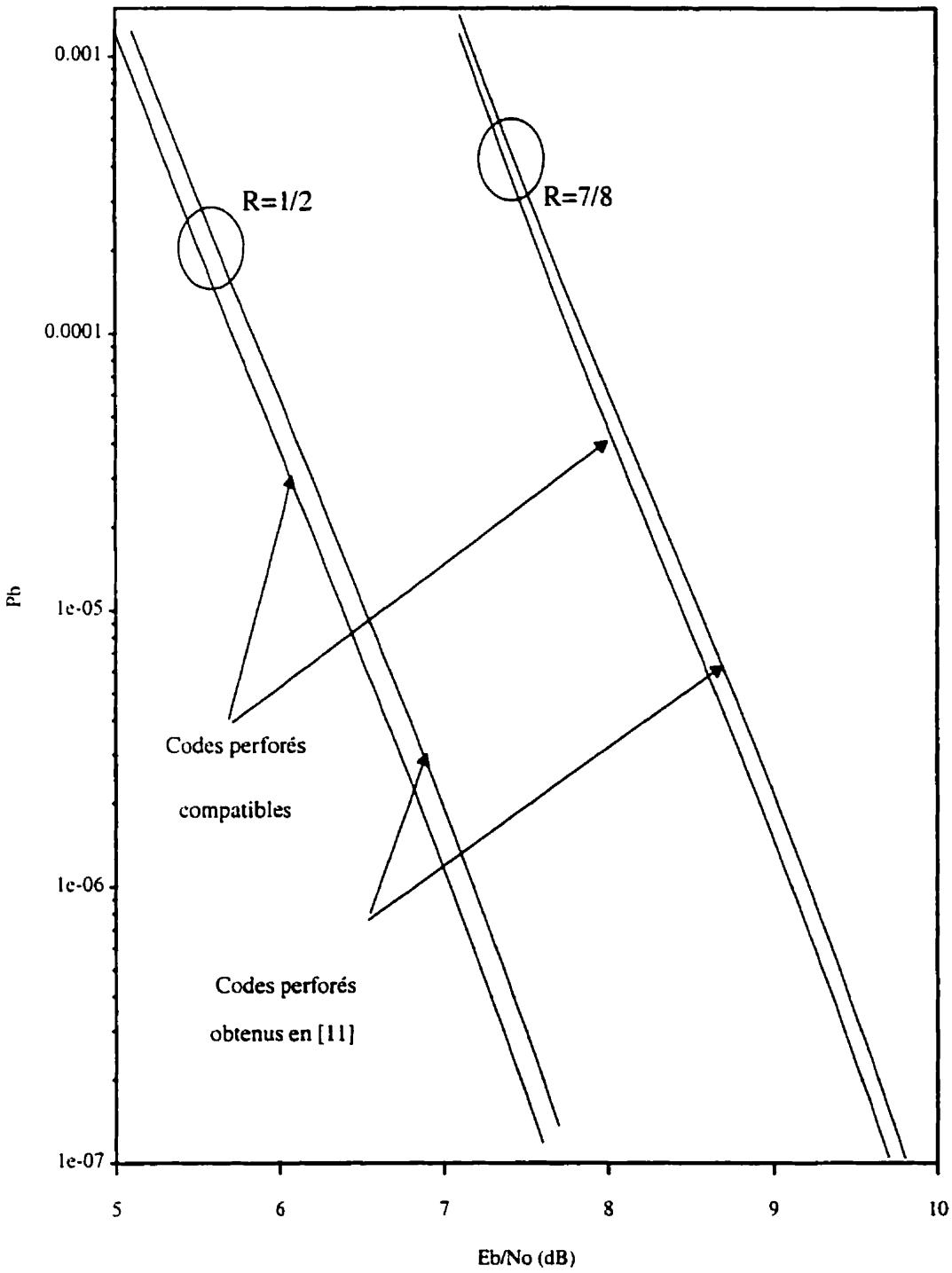

Figure 3.22 K=7: Comparaison des performances des codes perforés compatibles des taux R=1/2 et R=7/8 et des codes perforés existants des mêmes taux de codage

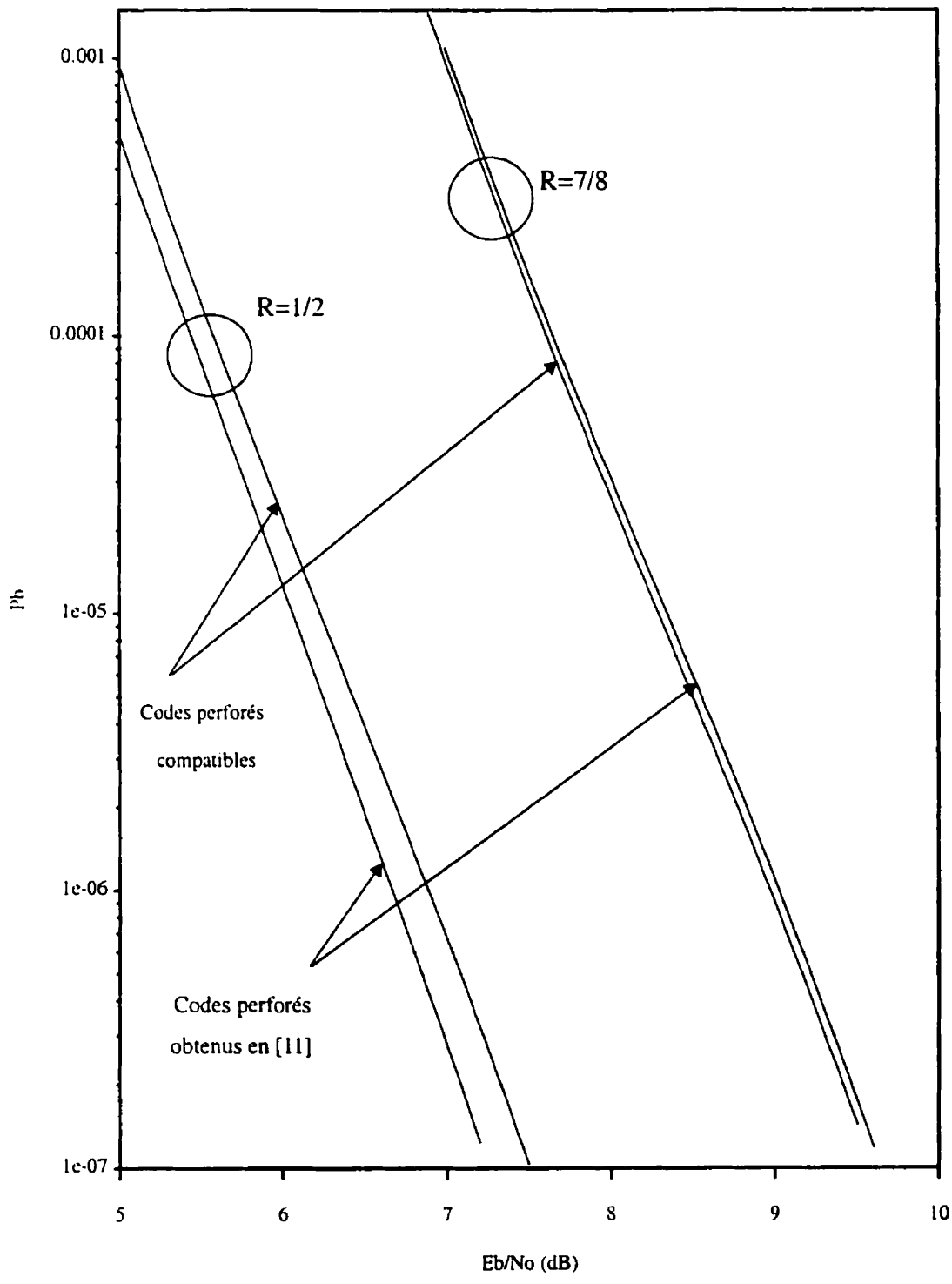

Figure 3.23 K=8: Comparaison des performances des codes perforés compatibles des taux R=1/2 et R=7/8 et des codes perforés existants des mêmes taux de codage

CHAPITRE 4

CONCATÉNATION SÉRIELLE DE CODES CONVOLUTIONNELS PERFORÉS

L'implantation des codes convolutionnels perforés dans un système concaténé en série est examiné dans ce chapitre. Ce système que l'on propose afin d'atteindre une très faible probabilité d'erreur par bit est caractérisé par sa grande flexibilité à s'adapter à des applications qui requièrent des qualités de service différentes. Cette flexibilité provient de l'utilisation des codes perforés qui, comme on a pu le constater dans le chapitre précédent, permettent de modifier facilement le taux de codage du système sans porter atteinte à la complexité du codeur ou du décodeur.

Afin d'améliorer les performances d'un système concaténé, l'algorithme de décodage SOVA (*Soft-Output Viterbi Algorithm*) est proposé dans la littérature [12]. Nous consacrerons notre étude à l'analyse des performances d'un système constitué de deux codeurs/décodeurs identiques concaténés en série dans lequel le processus de décodage est effectué selon l'algorithme SOVA. De plus, nous mettrons en relief l'amélioration des performances qu'un tel algorithme peut apporter par rapport à l'algorithme de Viterbi.

4.1. STRUCTURE DU SYSTÈME

La notion de concaténation des codes a été introduite par Forney [8]. La figure 4.1 schématise le principe de la concaténation en série de deux codes de taux de codage R1 et R2.

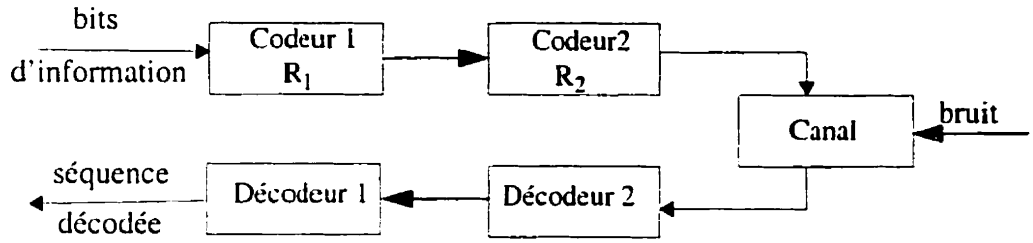

Figure 4.1 Schéma de principe de la concaténation en série de deux codes

Le taux de codage total résultant de la concaténation en série des deux codes de taux R_1 et R_2 devient alors:

$$R_{tot} = R_1 R_2 \ . \tag{4.1}$$

Traditionnellement, la concaténation en série est le résultat de deux types de codage en cascade: le codage convolutionnel de faible taux (codeur R_2) et le codage en blocs de type Reed-Solomon de taux de codage élevé (codeur R_1), communément appelé codage externe [15]. Le décodeur d'un tel système est donc constitué de deux décodeurs concaténés en série. Les erreurs à la sortie du décodeur interne (décodeur 2) étant généralement groupées, il est alors nécessaire d'insérer un délaceur entre les deux décodeurs et par conséquent un entrelaceur entre les deux codeurs [21]. Le rôle et le fonctionnement de l'entrelaceur et du délaceur seront expliqués en détail plus loin dans ce chapitre. Tenant compte des observations ci-dessus, le système représenté à la figure 4.1 peut être illustré par la figure 4.2.

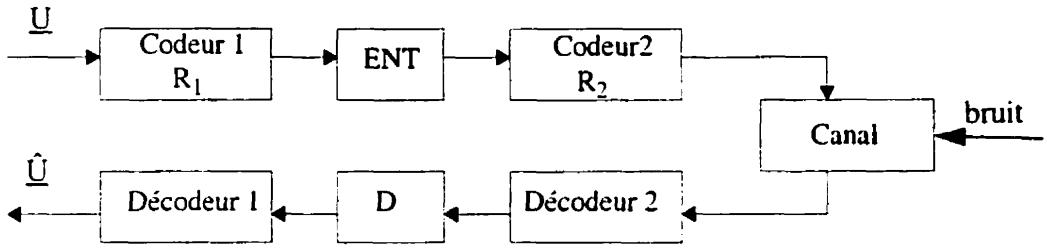

Figure 4.2 Structure du système concaténé en série avec entrelacement

Dans notre projet de maîtrise, les deux codeurs constituant le système concaténé sont des codeurs convolutionnels. Plus précisément, seuls les codes convolutionnels de taux d'origine de la forme $R_0 = 1/ V_0$ sont considérés. La raison réside dans le fait que les codes d'origine de cette forme simplifient le processus de décodage en traitant uniquement les treillis ayant deux branches sortant de chaque état. D'autre part, en utilisant les codes perforés [Chapitre 3], on sera en mesure d'obtenir des codes dont le taux de codage est de la forme R = b/ V.

L'émetteur du système est constitué d'un codeur de canal de taux $R_{tot} = R_1R_2$ suivi d'un modulateur. Les blocs correspondant au codeur et au décodeur seront décrits en détails plus loin dans le chapitre. Les bits d'information de la séquence $\underline{U}$ sont regroupés en blocs de longueurs variables. Ces bits d'information sont codés, puis modulés selon le procédé BPSK. L'information modulée est ensuite transmise dans un canal discret sans mémoire à bruit blanc additif et gaussien de moyenne nulle et de variance No/2. Au niveau du récepteur, la séquence reçue est démodulée, ensuite décodée selon l'algorithme de décodage optimal à maximum de vraisemblance afin d'obtenir une décision sur la séquence transmise ($\underline{\hat{U}}$).

4.1.1. L'ENTRELACEMENT

Les blocs notés ENT et D dans la figure 4.2 représentent respectivement l'entrelaceur et le délaceur. L'entrelacement consiste à placer dans un ordre différent et selon une règle prédéfinie les bits d'entrée. La fonction du délaceur est inverse de celle de l'entrelaceur. L'exemple le plus simple de l'entrelacement est celui de l'entrelaceur bloc représenté par une matrice de dimension MxN dont le principe de fonctionnement est le suivant: les bits rentrent ligne par ligne et sont lus colonne par colonne, comme le montre la figure 4.3.

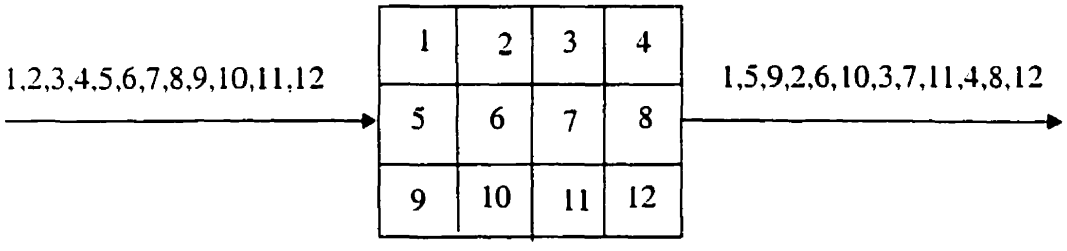

Figure 4.3 Principe de fonctionnement de l'entrelaceur bloc

Le délaceur quant à lui s'assurera que les bits reçus seront introduits colonne par colonne dans une matrice de même dimension et lus ligne par ligne afin de récupérer les symboles de la séquence transmise dans le bon ordre.

Le rôle de l'entrelaceur est de décorréler les symboles provenant des deux étapes du codage. Ceci est nécessaire dans la mesure où l'utilisation d'un modèle à bruit blanc gaussien exige que les symboles reçus soient décorrélés ou, encore, indépendants. Le délaceur a donc pour but de décorréler les erreurs provenants des deux étapes du décodage, et ce en étalant les salves d'erreurs produites à la fin de la première étape du décodage. Il est connu

qu'un décodeur utilisant l'algorithme de Viterbi peut corriger beaucoup plus facilement les erreurs isolées que les salves d'erreurs. Dans ce dernier cas, le décodeur est presque incapable de corriger des erreurs introduites dans le canal. Étant donné que dans le cas d'un canal bruité, la sortie du décodeur de Viterbi est affectée par des salves d'erreurs plutôt que par des erreurs isolées et que le système en question est composé de deux codeurs/décodeurs sériellement concaténés, il devient donc indispensable d'utiliser un entrelaceur/délaceur afin d'assurer le bon fonctionnement du deuxième décodeur (décodeur 1).

Le type d'entrelacement utilisé dans cette étude afin de simuler le système concaténé, est connu sous le nom d'entrelacement pseudo-aléatoire. L'entrelacement pseudo-aléatoire consiste à placer les bits d'entrée de façon «aléatoire» dans la liste entrelacée: pour chaque N bits du bloc, un nombre aléatoire comprise entre 1 et N représentant la position du bit dans la liste entrelacée est généré. Afin de récupérer les bits dans le bon ordre, la table de correspondance doit être connue au niveau du délaceur car, contrairement à l'entrelacement déterministe dont l'entrelacement bloc est un exemple, la règle régissant le positionnement des bits dans la liste entrelacée n'est pas connue d'avance pour l'entrelacement pseudo-aléatoire.

4.1.2. CODAGE DU SYSTÈME

Le codage du système (voir figure 4.4) consiste en deux codeurs convolutionnels concaténés de façon sérielle et séparés par un entrelaceur.

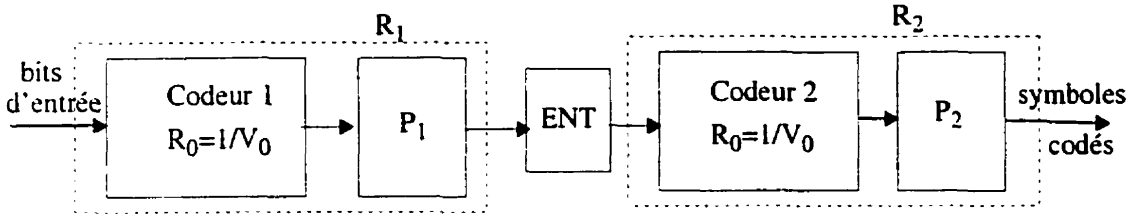

Figure 4.4 Émetteur du système concaténé

Comme on peut le constater à la figure 4.4, les deux codes perforés utilisent le même code d'origine de taux $R_0=1/V_0$. Ceci dans le but d'avoir un seul type de codeur/décodeur dans le système afin de simplifier les processus de codage et de décodage. En tenant compte des patrons de perforation de ces deux codeurs, les deux taux de codage résultants deviennent R_1 et R_2. Par conséquent le taux de codage du système entier est :

$$R_{tot} = R_1 R_2 \ .$$

Lorsque l'on tient compte des propriétés des codes convolutionnels perforés (voir Chapitre 3), on découvre que pour obtenir un taux de codage désiré du système (R_{tot}) , il suffit uniquement d'effectuer les changements au niveau des patrons de perforation utilisés . En d'autres termes, on choisit le taux de codage du système en fonction du degré de fiabilité exigé, et ce en partant d'un même code d'origine. Évidemment, l'existence d'un seul type de codeur d'origine dans le système concaténé simplifie grandement le processus de décodage. De plus, le processus de décodage est uniquement effectué selon le treillis du code d'origine dont les noeuds comportent deux branches entrantes et deux branches sortantes. Ce processus est donc indépendant du taux de codage total (R_{tot}).

Les deux types de codeurs d'origine (de taux de codage $R_0=1/V_0$) qui seront utilisés dans ce système sont le codeur convolutionnel non-systématique et le codeur récursif systématique. Le codeur récursif systématique [voir Section 2.6] utilise des générateurs sem-

blables à ceux du codeur convolutionnel qui lui est équivalent mais présente des performances légèrement meilleures à de faibles rapports signal/bruit. Et c'est précisément pour cette raison qu'il est plus avantageux d'utiliser cette dernière classe de codeurs dans les systèmes concaténés. Les codeurs récursifs sont en effet conçus dans le but d'obtenir de bonnes performances dans les systèmes de communication dont le rapport signal à bruit est faible.

Notons que les patrons de perforations choisis pour le codage récursif systématique ont la spécificité de préserver la propriété systématique du code en question. En d'autres termes, cela veut dire que les bits systématiques ne sont jamais perforés. Ceci se manifeste par l'obtention d'une matrice de perforation dont tous les bits de la première rangée valent 1. Ce choix au niveau des patrons de perforation associés aux codes récursifs systématiques, a pour but de préserver l'information transmise (les bits systématiques) dans la séquence décodée.

Si l'on désire obtenir le taux de codage total (R_{tot}) du système à partir du code d'origine R_0, on remarque qu'il existe au moins deux choix de valeurs possibles pour les taux de codage R_1 et R_2. Par exemple, à partir du code d'origine de taux R_0=1/6, le taux du codage R_{tot}=1/2 du système peut être obtenu en choisissant les patrons de perforations de la manière suivante: R_1=2/3 et R_2=3/4, ou bien R_1=3/4 et R_2=2/3. Pour le cadre de cette étude, on restreint le choix des taux de codage R_1 et R_2 à la condition: $R_2 < R_1$. La raison d'un tel choix réside dans le fait que le codeur de taux R_2, plus puissant que le codeur de taux R_1, rendra les performances du canal largement meilleures («super canal»). Dans ce cas, le codeur moins puissant (de taux R_1) sera en mesure de corriger les erreurs issues du «super canal». En effet, parmi les paramètres utilisés pour simuler le système concaténé, on retrouve la même condition ($R_2 < R_1$) au niveau des taux de codage

employés [12]. Afin de déterminer les bornes supérieures sur les performances des codes concaténés, Benedetto, Montorsi, Divsalar et Pollara [4] traitent, pour le même système, la situation inverse $(R_1 < R_2)$, mais selon une approche purement théorique.

4.1.3 MODULATEUR BPSK et le CANAL

Après avoir codé la séquence d'information, celle-ci est modulée. Le procédé de modulation utilisé est le BPSK (en anglais: *Binary Phase Shift Keying*). Le nombre de bits par symbole pour ce type de modulation est égal à 1. En appliquant la sortie X_k du deuxième codeur à l'entrée du modulateur de la figure 4.5, la sortie générée par celui-ci est:

$$X_k' = 2X_k - 1 \qquad (4.2)$$

Figure 4.5 Le modulateur BPSK

Une fois modulée, la séquence obtenue est transmise dans un canal sans mémoire, auquel s'ajoute un bruit blanc gaussien de moyenne nulle et de densité spectrale unilatérale N_0 (voir figure 4.6).

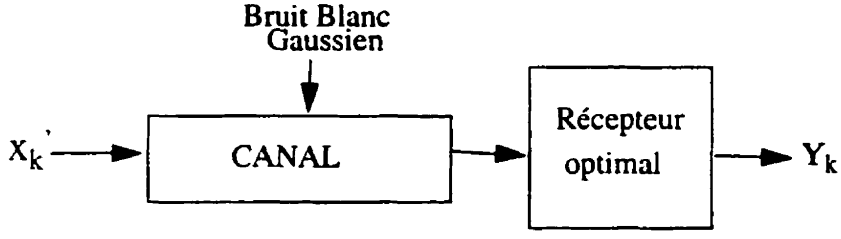

Figure 4.6 Modèle du canal à bruit blanc gaussien additif

Le bruit blanc gaussien généré par le canal est ajouté à la séquence modulée pour obtenir à l'entrée du décodeur 2 le symbole bruité suivant:

$$Y_k = X_k' + p_k = 2X_k - 1 + p_k \tag{4.3}$$

où X_k représente le symbole codé et p_k représente la composante du bruit blanc gaussien (p(x)) correspondant au symbole codé X_k. La densité de probabilité de la variable aléatoire gaussien x est exprimée par:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{x^2}{2\sigma^2})dx \tag{4.4}$$

où σ^2 est la variance du bruit blanc gaussien, et vaut:

$$\sigma^2 = \frac{1}{2R_{tot}\frac{E_b}{N_0}} \tag{4.5}$$

R_{tot} étant le taux de codage du système entier, comprenant les deux codeurs de taux R_1 et R_2, et E_b/N_0 étant le rapport entre l'énergie par bit transmis et le bruit introduit par le canal.

Pour que notre système puisse correspondre au système réel, le simulateur conçu utilise les valeurs réelles (non quantifiées) des symboles bruités (Y_k) afin d'estimer la séquence transmise. Évidemment, cela risque d'augmener le temps de calcul ainsi que la mémoire exigée par rapport à un système quantifié, car tous les calculs seront basés sur les valeurs réelles et non pas sur des valeurs entières.

4.1.4 RÉCEPTION ET DÉCODAGE DU SYSTÈME

Le récepteur du système est constitué de deux décodeurs concaténés en série séparés par un délaceur D. Le décodeur 1 est suivi d'un quantificateur comme le montre la figure 4.7.

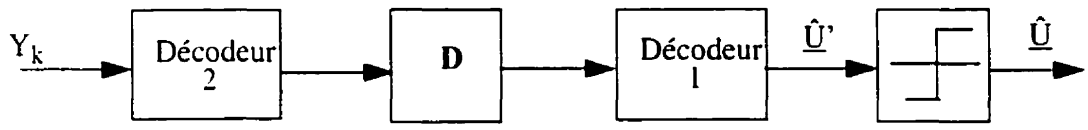

Figure 4.7 Récepteur du système concaténé

Le rôle du quantificateur est de récupérer les valeurs quantifiées à la fin du processus de décodage afin d'obtenir une estimation de la séquence transmise. Le quantificateur est utile lorsque l'algorithme SOVA (Soft Output Viterbi Algorithm) [12] est utilisé, car ce dernier produit à sa sortie non seulement les valeurs binaires comme l'algorithme de Viterbi, mais aussi les valeurs réelles. Le fonctionnement de l'algorithme SOVA sera expliqué plus en détails dans la section suivante. À ce stade de l'analyse, nous nous intéressons au rôle du quantificateur dont le fonctionnement est le suivant: si la valeur à l'entrée du quantificateur est supérieure à zéro, celui-ci décide que le bit transmis sera un 1, sinon le bit transmis est un 0. Cette opération peut être représentée par la relation suivante:

$$\hat{\underline{U}}' > 0 \Rightarrow \hat{\underline{U}} = 1, \hat{\underline{U}}' \leq 0 \Rightarrow \hat{\underline{U}} = 0 \tag{4.6}$$

Les deux décodeurs du récepteur sont des décodeurs optimaux à maximum de vraisemblance. Les deux algorithmes de décodage, soit l'algorithme de Viterbi et l'algorithme SOVA, sont alors testés et comparés. Quel que soit l'algorithme de décodage utilisé et le

taux de codage du système désiré, le décodage s'effectuera toujours selon le treillis du code d'origine de faible taux que l'on choisit afin de générer les taux de codage R_1 et R_2.

Comme on a pu le constater au Chapitre 2, le principe du décodage optimal à maximum de vraisemblance consiste à trouver en parcourant le treillis du code en question, la séquence la plus vraisemblable sachant la séquence reçue. Pour un décodeur de Viterbi fonctionnant de façon continue, la longueur minimale de la séquence décodée qui fournit un résultat fiable est souvent appellée la longueur de récupération. La longueur de récupération du décodeur dépend de sa longueur de contrainte K. Or, il a été démontré que celle-ci peut être exprimée comme suit: $L \geq 5K$. Cela veut dire que pour obtenir un résultat fiable, il suffit de décoder les symboles correspondants aux premiers (5-6)K bits d'information de la séquence reçue. On peut donc dire que l'influence des autres symboles reçus a un effet négligeable sur la fiabilité du processus de décodage.

Dans l'algorithme utilisé, le décodage s'effectue en fonction de la longueur des blocs d'information et non pas en fonction de la longueur de récupération. Cela implique qu'au niveau du récepteur, tous les symboles de la séquence reçue sont décodés afin d'obtenir une estimation de la séquence transmise.

La séquence qui ramène le codeur à l'état initial, la queue, se retrouve dans chacune des deux étapes du codage. Sachant qu'à la fin de la queue il ne reste qu'un seul survivant dans le treillis du code d'origine et que ce dernier correspond au chemin le plus vraisemblable (voir la section 2.7.2), il est donc évident que l'existence des bits de la queue simplifie le processus de décodage. Les bits des deux queues qui ne sont pas nécessairement égaux, mais ils sont connus au récepteur et, par conséquent, les symboles reçus qui correspondent aux bits de la queue ne seront pas décodés. Rappellons que les bits des deux queues des

codes convolutionnels non-récursifs sont équivalents car tous les bits valent 0, alors que les queues des codes récursifs systématiques ne sont pas nécessairement équivalents. Dans ce dernier cas les bits de chaque queue doivent être calculés en fonction de l'état final du codeur correspondant. L'augmentation de la complexité du processus de décodage avec la longueur des blocs d'information à l'entrée du système parait évidente, d'autant plus que le décodage s'effectue par l'intermédiaire des deux décodeurs optimaux concaténés en série.

Afin d'obtenir des résultats fiables et de préserver une complexité raisonnable du processus de décodage, la longueur des blocs d'information a été fixée à 500 bits dans nos simulations. Le simulateur conçu a pour longueur de contrainte maximale Kmax=6. Or, on constate que la longueur du bloc d'information choisie (500 bits) assure la fiabilité des résultats pour toutes les valeurs de K utilisées (K<6). De plus, la comparaison des performances du système sont effectuées dans les mêmes conditions (même longueur des blocs d'information), mais en choisissant des codes et des algorithmes de décodage différents. Le choix de la longueur des blocs sera justifié à l'aide de résultats numériques présentés plus loin dans ce chapitre.

4.2. ALGORITHMES DE DÉCODAGE

Comme on a pu le constater précédemment, le système examiné est un système sérielle-ment concaténé contenant deux codeurs et deux décodeurs. Les performances de ce système sont fortement influencées par les deux taux de codage R_1 et R_2 utilisés, mais aussi et surtout par l'algorithme de décodage que l'on choisit. Dans le cadre de ce travail, les deux algorithmes optimaux de décodage implantés sont l'algorithme de Viterbi et l'algorithme SOVA. Les performances de ces deux algorithmes sont alors comparées pour le

même système. L'algorithme SOVA apporte une amélioration significative des performances du système car il produit à sa sortie les valeurs non-quantifiés («soft») qui seront utilisées dans la prochaine étape du décodage. Afin de pouvoir comparer les performances de ces deux algorithmes, il est utile à ce stade de décrire brièvement le principe de fonctionnement de ces deux types d'algorithmes.

4.2.1. L'ALGORITHME DE VITERBI

Le principe de fonctionnement de l'algorithme de Viterbi a déjà été décrit au chapitre 2. En quelques mots, il s'agit d'un algorithme à maximum de vraisemblance qui parcourt tous les états du treillis afin de retrouver la séquence la plus vraisemblable.
Ce processus de décodage peut être effectué quelque soit le type de quantification utilisée (dure ou douce). Donc, à l'entrée du décodeur de Viterbi, les valeurs entrantes peuvent être soit quantifiées soit non-quantifiées. En principe, pour un canal gaussien, la séquence bruitée à l'entrée du décodeur est constituée de valeurs réelles (non-quantifiées) tandis que pour un canal binaire symétrique celle-ci sera constituée de valeurs binaires (quantifiées). Cependant, quelque soit l'entrée de décodeur, la séquence produite à la sortie du décodeur de Viterbi sera tout le temps constituée de valeurs binaires. Ceci est illustré à la figure 4.8. Ajoutons enfin que pour un tel système, la quantification douce amène une amélioration au niveau de performances d'environ 2 dB par rapport à la quantification dure.

Figure 4.8 Entrées-Sorties pour l'algorithme de Viterbi

Dans le cas d'un système concaténé en série constitué de deux décodeurs de Viterbi, l'entrée du deuxième décodeur serait donc composée de valeurs quantifiées (valeurs binaires) utilisées afin d'estimer la séquence transmise. Or, il est clair dans ce cas que l'intérêt de la quantification douce n'est plus aussi visible que dans le cas d'un système à un seul décodeur de Viterbi (au figure 4.8). Ceci sera démontré plus loin dans cette section avec valeurs numériques à l'appuie.

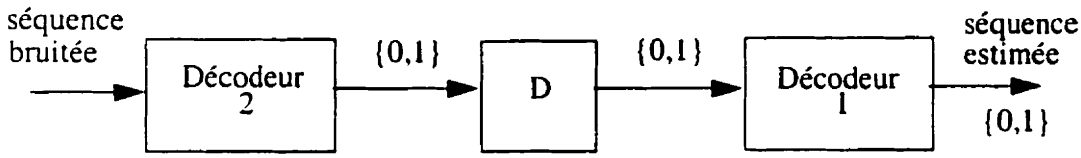

Figure 4.9 Principe de la concaténation en série de deux décodeurs de type Viterbi

D'autre part, d'après la figure 4.9 le deuxième décodeur (décodeur 1) fonctionne uniquement avec la quantification dure. Autrement dit, les entrées et les sorties de ce dernier sont également des valeurs binaires. Telle est donc la raison pour laquelle le système contenant deux décodeurs de type Viterbi ne nécessite pas l'utilisation d'un quantificateur après le décodeur 1 (voir la figure 4.7). Les valeurs obtenues à la sortie du décodeur 1 sont déjà des valeurs quantifiées.

Performances du système concaténé appliquant l'algorithme de Viterbi

Le système concaténé en série est simulé en appliquant l'algorithme de Viterbi au décodage. Ceci permet de comparer les performances d'un tel système en utilisant deux types de codeurs : le codeur convolutionnel et le codeur récursif systématique. Les résultats obtenus à la figure 4.10 correspondent à un système ayant les paramètres suivants:

- Longueur de contrainte : K=3
- Taux de codage (codeur 1): $R_1 = 1/2$
- Taux de codage (codeur 2): $R_2 = 1/2$
- Taux de codage global : $R_{tot} = 1/4$
- Entrelaceur : Aléatoire

Les résultats obtenus à la figure 4.11 correspondent à un système ayant les paramètres suivants:

- Longueur de contrainte : K=4
- Taux de codage (codeur 1): $R_1 = 2/3$
- Taux de codage (codeur 2): $R_2 = 1/2$
- Taux de codage global : $R_{tot} = 1/3$
- Entrelaceur : Aléatoire

Les résultats obtenues aux figures 4.10 et 4.11 démontrent que les codes récursifs et systématiques apportent une amélioration des performances pour un système concaténé. Cette amélioration des performances se manifeste surtout à de faibles rapports signal à bruit.

Exemple

Pour les résultats obtenus à la figure 4.10, l'amélioration apportée par le codeur récursif systématique varie entre 0.4 dB à $P_b=10^{-1}$ et 0 dB à $P_b=10^{-2}$.

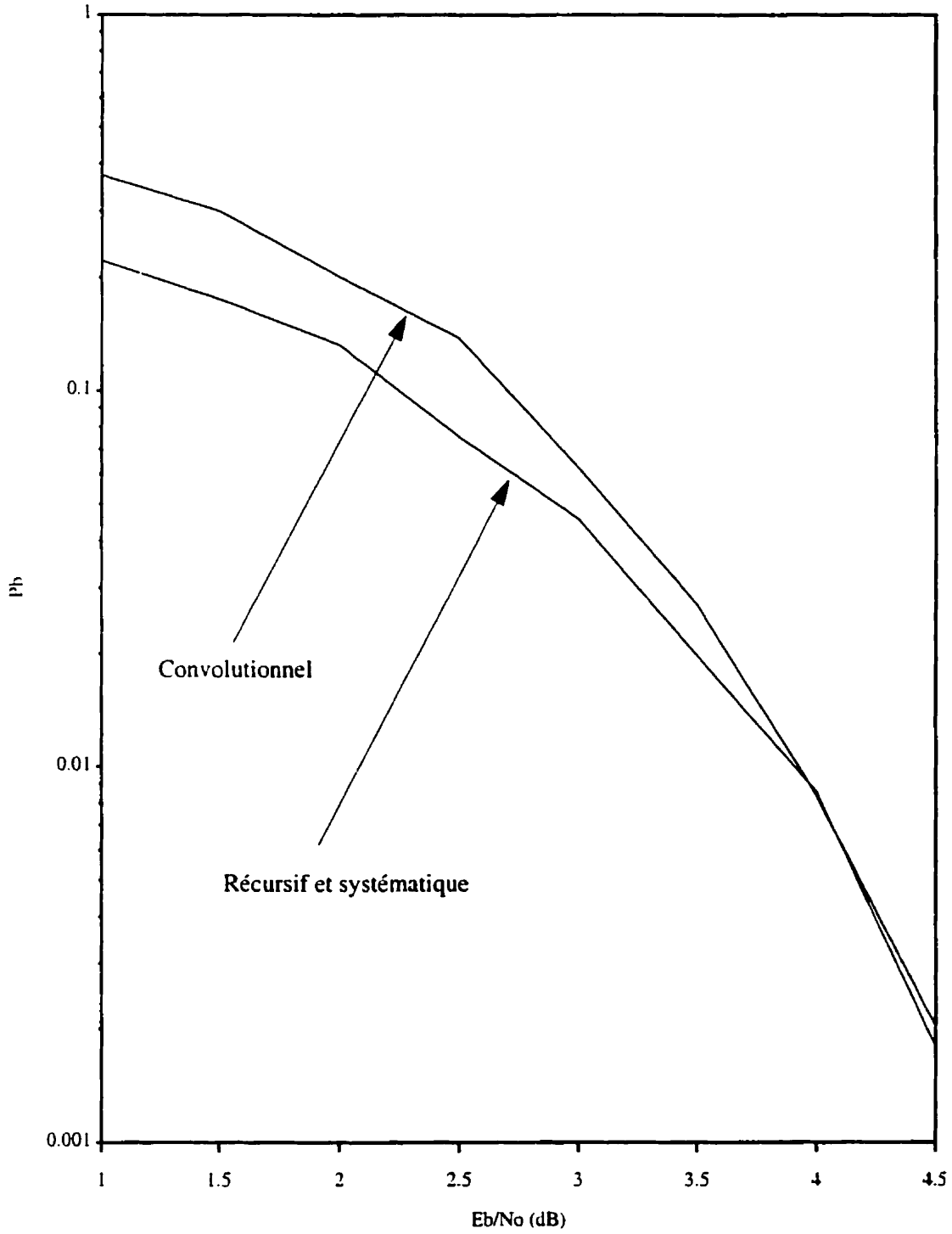

Figure 4.10 Concaténation en série de deux décodeurs de types Viterbi (K=3, R1=1/2, R2=1/2 => Rtot = 1/4)

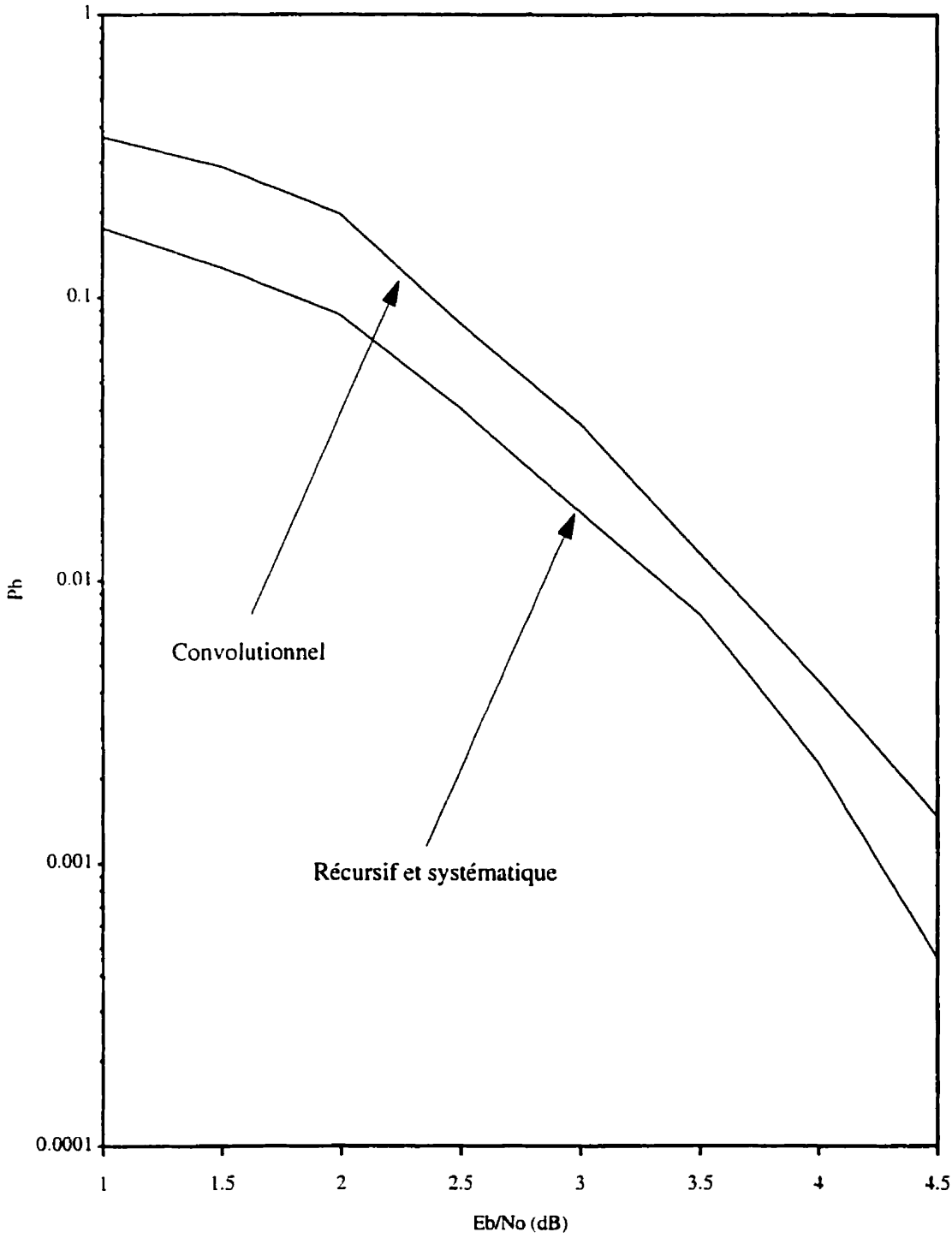

Figure 4.11 Concaténation en série de deux décodeurs de types Viterbi (K=4, R1=2/3, R2=1/2 => Rtot = 1/3)

Dans le but de comparer les performances du système concaténé utilisant l'algorithme de Viterbi avec celles des codes des taux équivalents à R_{tot} du système concaténé, les bornes supérieures sur la probabilité d'erreur de ces codes sont tracées. Soulignons ici que les bornes supérieures sont obtenues en utilisant les spectres des codes en question.

La figure 4.12 est obtenue en traçant la borne supérieure sur la probabilité d'erreur pour le code suivant: K=3, R=1/4 (G_1=5, G_2=7, G_3=7, G_4=7). La probabilité d'erreur du système concaténé obtenue à la figure 4.10 avec le codeur d'origine récursif systématique est tracée sur la même figure afin d'estimer les performances obtenues.

La figure 4.13 correspond à la borne supérieure du code K=4, R=1/3 (G_1=15, G_2=17, G_3=13). Cette borne est comparée sur la même figure avec la probabilité d'erreur du système concaténé utilisant le codeur récursif systématique de la figure 4.11.

Comme on peut le constater d'après les figures 4.12 et 4.13, l'utilisation de l'algorithme de Viterbi dans un système concaténé n'apporte aucune amélioration au niveau des performances. Bien au contraire, on constate qu'un système simple, constitué d'un codeur convolutionnel de taux équivalent à R_{tot} et d'un décodeur Viterbi a de meilleurs performances (d'environ 0.5 à 1 dB) qu'un système concaténé utilisant l'algorithme de Viterbi.

Or, il est évident que pour que la mise en oeuvre de la concaténation en série soit justifiée, les performances obtenues doivent être au moins aussi bonnes que celles du système simple utilisant un codeur/décodeur. Enfin, afin d'obtenir l'amélioration des performances d'un système concaténé, un raffinement ou une adaptation de l'algorithme de Viterbi paraît nécessaire.

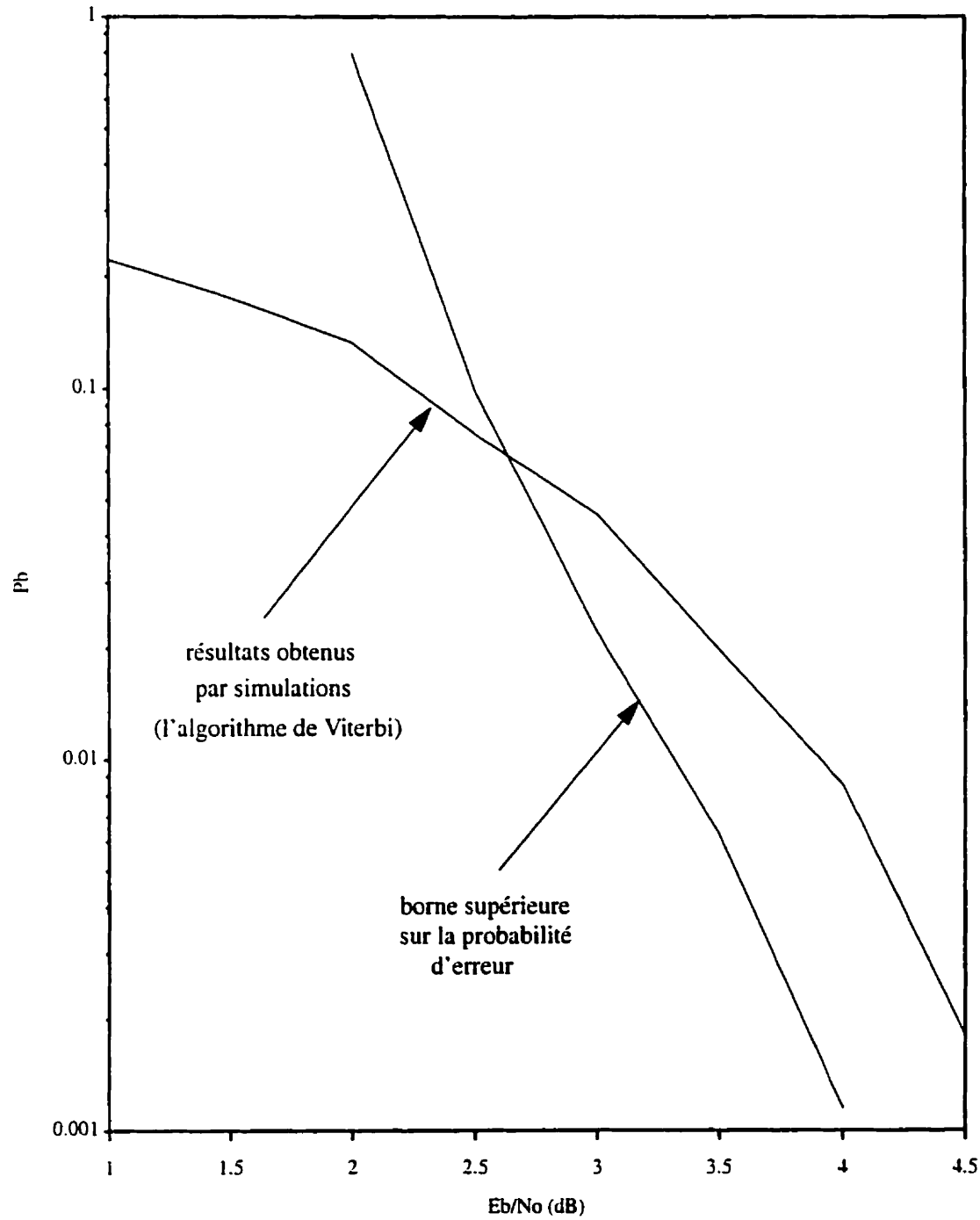

Figure 4.12 Comparaison des performances du système concaténé utilisant l'algorithme de Viterbi avec la borne union sur la probabilité d'erreur pour le même taux de codage R_{tot}=1/4 (K=3, R1=1/2, R2=1/2 => Rtot = 1/4)

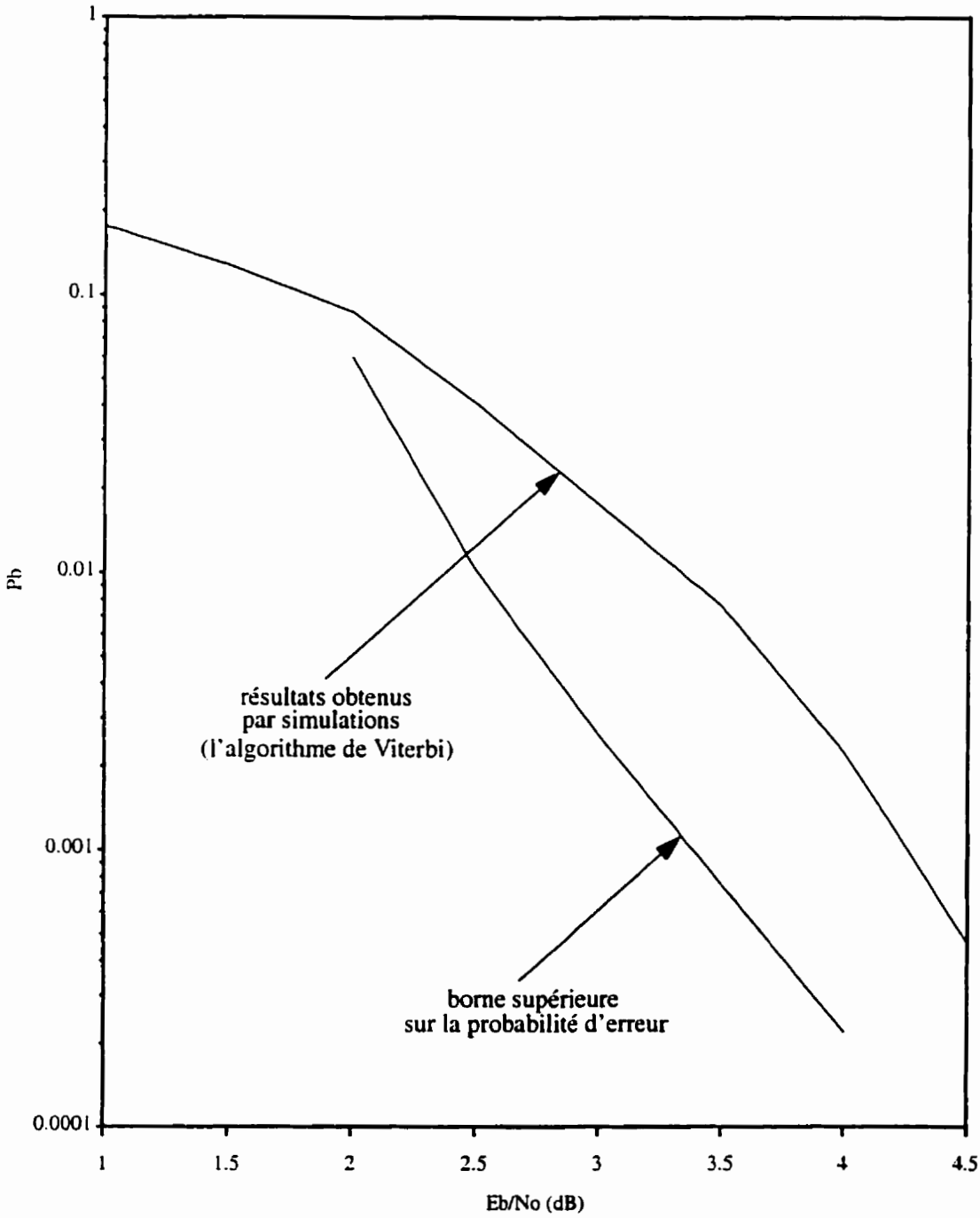

Figure 4.13 Comparaison des performances du système concaténé utilisant l'algorithme de Viterbi avec la borne union sur la probabilité d'erreur pour le même taux de codage R_{tot}=1/3 (K=4, R1=2/3, R2=1/2 => Rtot = 1/3)

4.2.2. **L'ALGORITHME SOVA** (en anglais: «*Soft-Output Viterbi Algorithm*»)

Parmi les algorithmes de décodage proposés afin d'améliorer les performances d'un système concaténé, on retrouve l'algorithme SOVA (en anglais: «Soft-Output Viterbi Algorithm») [12].

Figure 4.14 Entrées-Sorties du décodeur SOVA

En respectant la notation de la figure 4.14, le décodeur fournit une estimation û' de la séquence transmise en traitant la séquence reçue Y. Ce que l'on désire obtenir avec l'algorithme SOVA, c'est, en plus de l'estimation du symbole transmis (fourni aussi par l'algorithme de Viterbi), une estimation de la probabilité que le symbole ait été détecté incorrectement:

$$\hat{p}'_k = Prob\{\hat{u}'_k \neq u'_k | Y\} \tag{4.7}$$

Cette dernière information aide à estimer la fiabilité de la décision du décodeur. Elle sera utilisée dans la prochaine étape du décodage afin d'améliorer les performances du système concaténé.

Sachant que les algorithmes à maximum de vraisemblance produisent à leurs sorties des salves d'erreurs plutôt que des erreurs isolées, l'utilisation d'un délaceur paraît nécessaire afin d'éviter la corrélation des erreurs des symboles estimés et, par conséquent la corrélation des probabilités de détection incorrecte. Dans ce dernier cas, on peut considérer que l'algorithme SOVA fournit à sa sortie les estimations des symboles transmis ($\hat{u}_k$') avec des

probabilités d'erreurs (p_k) non-corrélées. Par conséquent, le canal au niveau du décodeur extérieur (décodeur 1) devient un canal discret sans mémoire car les symboles provenant du décodeur 2 sont non-corrélés. Comme le décodeur 1 est un décodeur à maximum de vraisemblance, la métrique utilisée afin de trouver la séquence la plus vraisemblable s'écrit:

$$\sum_k x_k^{(m)} \hat{u}_k \log \frac{1-\hat{p}_k}{\hat{p}_k} \tag{4.8}$$

où $\hat{u}_k$ est la valeur de la décision non-quantifiée (en anglais: «*hard decision*») sortant du premier décodeur et, où $x_k^{(m)}$ représente le symbole k de la séquence m, $x^{(m)}{}_k = \pm 1$.

Chaque valeur quantifiée est alors constituée du produit de la valeur $\hat{u}_k$ non-quantifiée (en anglais: «*hard decision*») et de la fiabilité de cette décision:

$$\hat{\Lambda}_k = \hat{u}_k \hat{L}_k \tag{4.9}$$

avec:

$$\hat{L}_k = \log \frac{1-\hat{p}_k}{\hat{p}_k}, \; 0 \le \hat{L}_k < \infty \tag{4.10}$$

où $\hat{p}_k$ représente la probabilité que le symbole ait été décodé incorrectement, avec $\hat{u}_k \in \{\pm 1\}$.

Les valeurs produitent à la sortie du premier codeur peuvent être utilisées dans la prochaine étape du décodage. C'est l'existence de ces valeurs quantifiées à la sortie du décodeur qui apporte une amélioration des performances du système concaténé par rapport à celles de l'algorithme de Viterbi, car elles n'indiquent pas seulement les valeurs binaires de la séquence estimée mais aussi la mesure de la fiabilité de cette décision.

Décodage itératif

Grâce à sa structure, le décodeur de type SOVA peut facilement être utilisé dans les systèmes à décodage itératif [14]. Bien que le décodage itératif améliore de façon significative les performances d'un système, celui-ci demeure beaucoup plus complexe que le décodage non-itératif et ne sera pas traité dans ce travail.

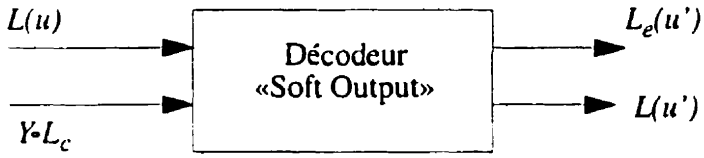

Figure 4.15 Entrées-Sorties du décodeur de type «soft-output» pour le décodage itératif

Les entrées et les sorties du décodeur sont définies comme suit:

$L(u)$ sont des valeurs à-priori,

L_c représente la fiabilité du canal,

$L(u')$ représente les valeurs quantifiées (en anglais: «soft outputs»),

$L_e(u')$ est l'information extrinsèque et

Y représente les symboles reçus.

$L_e(u')$ est le paramètre utilisé dans le décodage itératif et est définit comme suit:

$$L_e(u') = L(u') - [YL_c + L(u)] \tag{4.11}$$

Sachant que l'algorithme de décodage choisi dans ce projet est non-itératif, la sortie $L_e(u')$ ne sera jamais utilisée. Pendant le décodage itératif la valeur calculée de $L_e(u')$ est «reinjecté» dans la prochaine itération à l'entrée $L(u)$, ce qui nous permet de constater que la valeur associée à $L(u)$ est égale à zéro pour le décodage non-itératif.

Décodage non-itératif

Le schéma du décodeur utilisant l'algorithme SOVA pour un décodage non-itératif peut être illustré par la figure suivante:

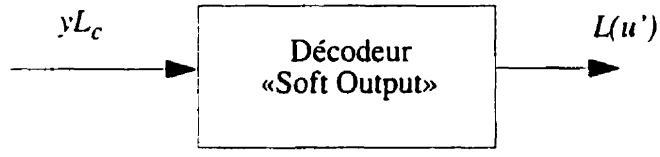

Figure 4.16 Entrées-Sorties du décodeur de type «soft-output» pour le décodage non-itératif

Le paramètre L_c indiquant la fiabilité du canal, peut être exprimé [14] en fonction du rapport signal sur bruit (E_s/N_0) et de l'amplitude des évanouissements (α) pour le cas d'un canal à évanouissements (en anglais: *fading channel*). Ceci est représenté par (4.12).

$$L_c = 4\alpha\left(\frac{E_s}{N_0}\right) \tag{4.12}$$

Dans le cas du canal gaussien (canal considéré dans ce mémoire), le paramètre α devient égal à 1.

Algorithme SOVA

Le principe de fonctionnement du décodeur de type SOVA se subdivise en deux grandes étapes. La première étape qui est similaire à celle du décodeur de Viterbi, consiste à retrouver la séquence binaire la plus vraisemblable en utilisant la règle du maximum de

vraisemblance. Cela veut dire que le chemin ayant la métrique cumulative la plus élevée est considéré comme étant le chemin survivant. Dans le cas du décodeur de type SOVA, la métrique cumulative est calculée en tenant compte de la fiabilité du canal, L_c. Elle s'exprime de la façon suivante:

$$M_k = M_{k-1} + \frac{1}{2} \sum_{i=1}^{V} L_c x_{k,i} y_{k,i} \qquad (4.13)$$

M_{k-1} représente la métrique cumulative à l'instant k-1, $x_{k,i}$ le i-ème symbole de la branche à l'instant k et $y_{k,i}$ est le i-ème symbole reçu à l'instant k. Le paramètre L_c est calculé en utilisant (4.12). Le nombre de symboles de la branche est égal à V.

La première étape du décodage consiste donc à retrouver, en parcourant le treillis du code, le chemin dont la métrique cumulative est la plus élevée, afin d'estimer la séquence transmise. La décision finale sur la séquence décodée est prise avec un délai d, où d est suffisamment grand pour que les 2^{K-1} chemins survivants convergent avec une probabilité suffisamment grande.

À la deuxième étape du décodage, le décodeur du type SOVA fournit une mesure de la fiabilité de décision de la séquence décodée. En respectant les notations de la figure 4.17, le processus d'obtention des valeurs quantifiées est illustré par l'algorithme présenté en [12] et expliqué plus loin dans cette section.
Les paramètres utilisés dans l'élaboration du principe d'obtention des valeurs quantifiées sont indiqués à la figure 4.17, et se définissent comme suit:

- d représente le délai de la décision
- d_m représente la longueur des deux chemins avant l'instant de leurs

convergence à un état donné,

- ν est la mémoire du code en question, $\nu = K - 1$
- $\hat{u}_j^z$ représente le bit décodé sur le chemin z à l'instant j ($z \in \{1, 2\}$)

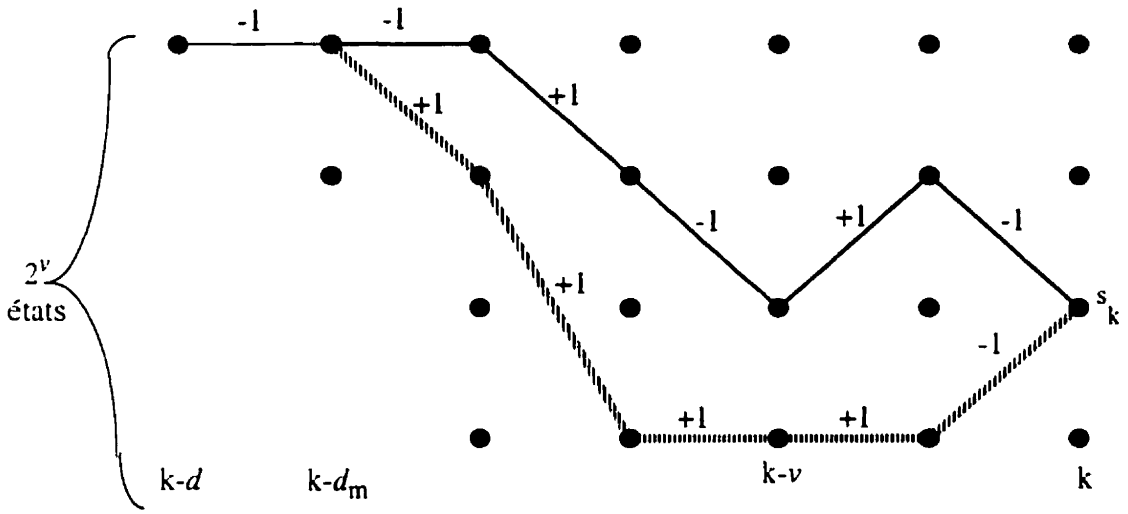

Figure 4.17 Exemple de l'algorithme SOVA

L'algorithme de décodage selon SOVA proposé en [12] se définit de la façon suivante:

- Pour chaque état s_k du treillis :

  a) Les métriques M_k et M_k' des deux chemins convergeant à l'état s_k sont calculées

  b) Le maximum entre M_k et M_k' est déterminé, et le chemin survivant ayant la métrique la plus grande est sauvegardé

  c) La relation : $\Delta_k = \left| M_k - M_k' \right|$ est effectuée

  d) La valeur quantifiée est initialisée à ∞ :

$$\hat{L}_k(s_k) = \infty$$

e) Pour j = k-ν jusqu'à j =k-d_m

Le chemin survivant est comparé au chemin concurrent;

si $\hat{u}_j^1 \neq \hat{u}_j^2$, on réinitialise la valeur quantifiée $L(\hat{u}_j)$ avec:

$$L(\hat{u}_j) \leftarrow min\,[L(\hat{u}_j), \Delta_k] \tag{4.14}$$

On peut constater à partir du fonctionnement de cet algorithme qu'une fois les bits décodés déterminés, la complexité ajoutée (par rapport à l'algorithme de Viterbi) provient du fait qu'on doit parcourir le treillis en sens inverse afin de déterminer la fiabilité de la décision.

En tenant compte que la queue des ν=K-1 bits remettant le codeur en état initial est ajoutée lors du codage, l'intervalle dans lequel les bits d'information des deux chemins de la figure 4.17 peuvent différer varie entre j=k-d_m et j=k-ν. Notons par e le nombre de bits d'information qui distinguent ces deux chemins. Par conséquent, le nombre de bits d'information qui sont identiques pour ces deux chemins est égal à d_m - e.

Or, pour un délai de décodage donné d , la complexité de calcul que l'algorithme de SOVA ajoute par rapport à l'algorithme de Viterbi devient:

- Comparaison 2^{ν} (d-ν) bits au maximum
- $2^{\nu}e$ mises à jour de L_k

4.2.3 DÉCODEUR SOVA DANS UN SYSTÈME CONCATÉNÉ EN SÉRIE

Le système concaténé en série basé sur l'algorithme de SOVA a été simulé, et ses performances sont évaluées en fonction de différents paramètres qui le constitue tel que le taux de codage et la longueur de contrainte.
Malheureusement, la grande majorité de résultats que l'on retrouve dans la littérature sont obtenus à partir d'un décodage itératif basé sur l'algorithme SOVA [13],[14]. Il y a très peu de résultats qui reflètent les performances d'un système concaténé utilisant l'algorithme SOVA non-itératif.

Les paramètres utilisés dans nos simulations sont choisis d'une manière à pouvoir comparer les résultats obtenus avec ceux de la littérature.
Par conséquent, les deux systèmes sont simulés en utilisant les paramètres suivants:

1)
- Longueur de contrainte : **K=3**
- Taux de codage (codeur 1): $\mathbf{R_1 = 1/2}$
- Taux de codage (codeur 1): $\mathbf{R_2 = 1/2}$
- Taux de codage global : $\mathbf{R_{tot} = 1/4}$
- Entrelaceur : Aléatoire

2)
- Longueur de contrainte : **K=4**
- Taux de codage (codeur 1): $\mathbf{R_1 = 2/3}$
- Taux de codage (codeur 1): $\mathbf{R_2 = 1/2}$
- Taux de codage global : $\mathbf{R_{tot} = 1/3}$
- Entrelaceur : Aléatoire

et leurs performances sont déterminées en variant les paramètres suivants:

- Longueur de blocs d'information
- Type du codeur utilisé

Les paramètres choisis en 1) correspondent à ceux présentés en [24] et les paramètres choisis en 2) correspondent à ceux dans [12], [13] et [20].

4.2.3.1 PERFORMANCES DU SYSTÈME CONCATÉNÉ EN FONCTION DE LA LONGUEUR N DES BLOCS D'INFORMATION

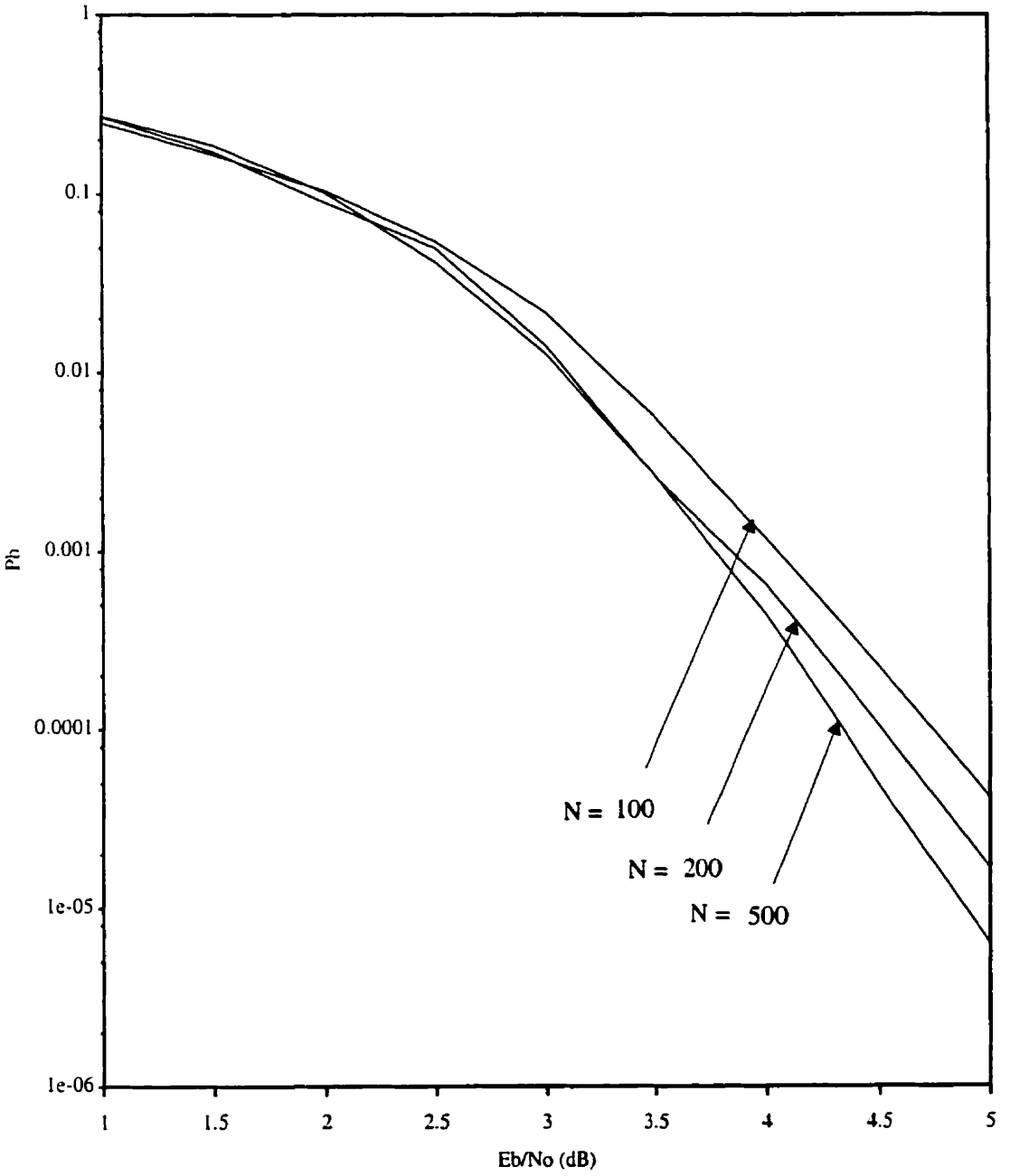

Figure 4.18 Influence de la longueur du bloc (en bits) sur la performance du système concaténé (K=3, R1=1/2, R2=1/2 => Rtot = 1/4; Codeurs convolutionnels)

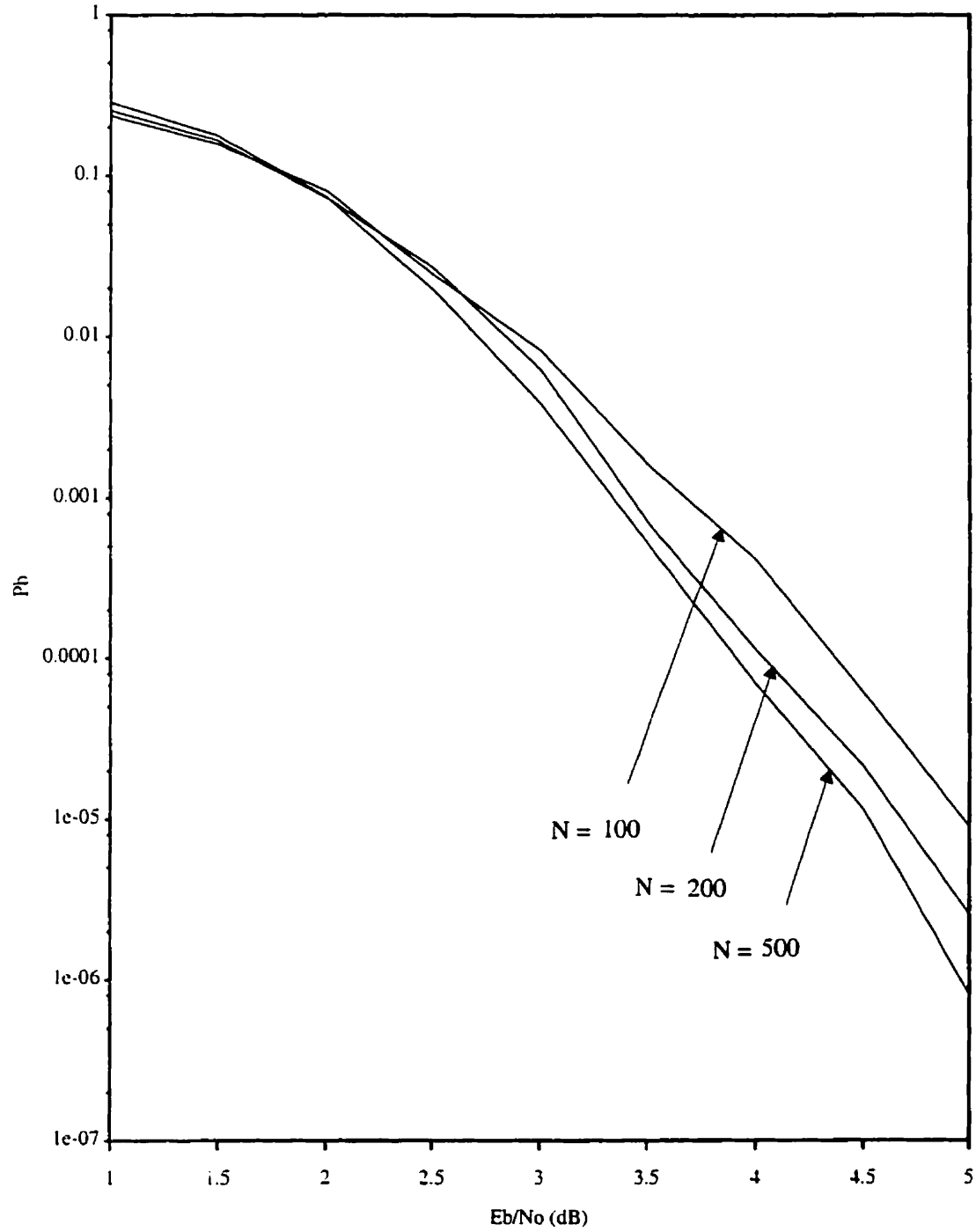

Figure 4.19 Influence de la longueur du bloc (en bits) sur la performance du système concaténé (K=4, R1=2/3, R2=1/2 => Rtot = 1/3; Codeurs convolutionnels)

D'après les figures 4.18 et 4.19, on constate qu'une longueur de bloc d'information à l'entrée du système concaténé de 500 bits apporte une amélioration des performances d'environ 0.5 dB par rapport à une longueur des blocs de 100 bits. Cette amélioration de performances provient du fait que les salves des erreurs provenant du canal sont mieux étalées quand les blocs d'une longueur de 500 bits sont utilisés. Par conséquent, c'est cette longueur de bloc que l'on utilisera tout au long de nos simulations en évitant toutefois d'augmenter significativement le délai du décodage de la séquence reçue.

4.2.3.2 PERFORMANCES DU SYSTÈME CONCATÉNÉ EN FONCTION DU TYPE DE CODEUR UTILISÉ

Comme on l'a mentionné auparavant, les deux types de codeurs (codeur convolutionnel et codeur récursif systématique) seront utilisés et leurs performances dans un système concaténé seront comparés. L'algorithme de décodage utilisé dans ce cas sera l'algorithme SOVA.

La figure 4.20 représente les performances du système concaténé, pour les deux types de codeurs : codeur convolutionnel (non-récursif) et codeur récursif systématique. Les paramètres choisis dans ce cas sont: K=3, R1=R2=1/2 => Rtot=1/4.

La figure 4.21 représente les performances du système concaténé, pour les deux types de codeurs : codeur convolutionnel (non-récursif) et codeur récursif systématique. Les paramètres choisis dans ce cas sont: K=4, R1=2/3 , R2=1/2 => Rtot=1/3.

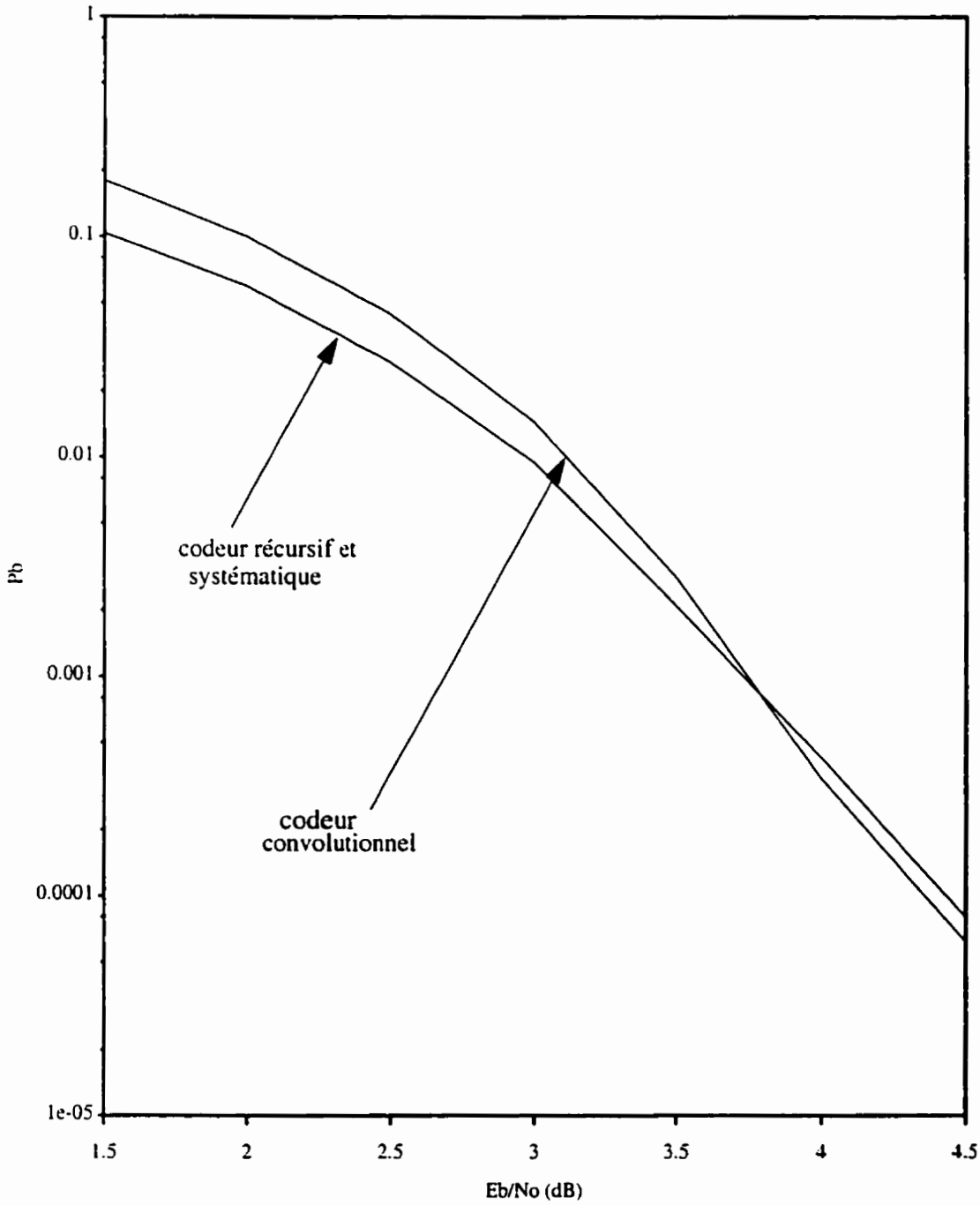

Figure 4.20 Performances du système en fonction du type de codeur utilisé; L'algorithme du décodage utilisé est SOVA (K=3, R1=1/2, R2=1/2 => Rtot = 1/4)

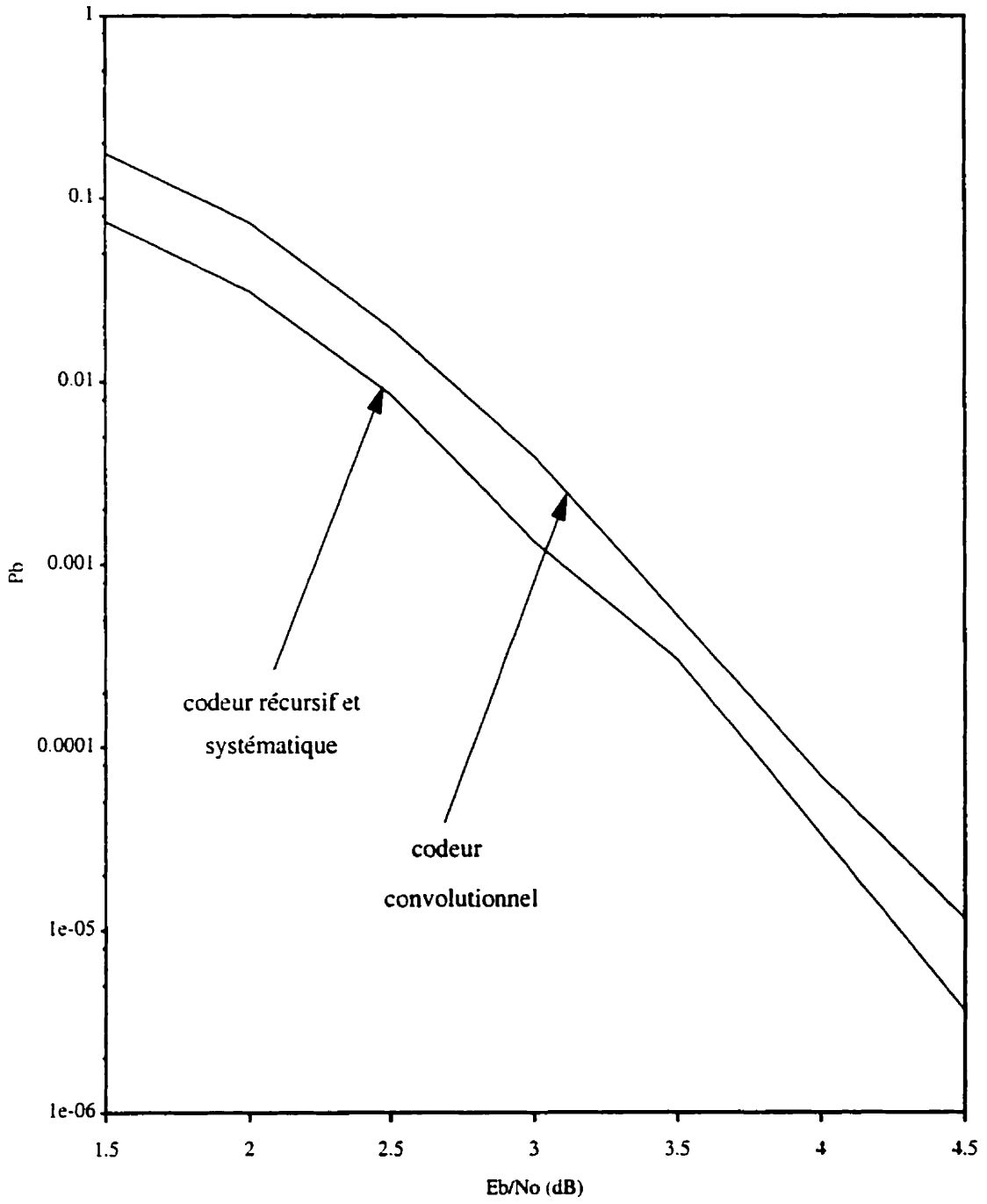

Figure 4.21 Performances du système en fonction du type de codeur utilisé; L'algorithme du décodage utilisé est SOVA (K=4, R1=2/3, R2=1/2 => Rtot = 1/3)

En examinant les figures 4.20 et 4.21, on déduit que l'utilisation des codes récursifs systématiques dans un système concaténé dont le décodage s'effectue selon l'algorithme SOVA, procurent de meilleures performances que les codes convolutionnels. Cette amélioration des performances varie entre 0 et 0.3 dB en fonction du rapport Eb/N_0 et des patrons de perforation utilisés.

Pour le reste de nos simulations, nous utiliserons des codes récursifs systématiques plutôt que des codes convolutionnels à cause de leur léger avantage sur les performances du système.

4.2.3.3 PERFORMANCES DU SYSTÈME CONCATÉNÉ : SOVA vs VITERBI

A ce stade de l'analyse, nous désirons comparer les performances lorsque les deux types d'algorithmes de décodage suivant sont utilisés : l'algorithme de Viterbi et l'algorithme de SOVA .

La figure 4.22 représente les performances du système concaténé, constitué de deux codeurs récursifs systématiques : K=3, R1=R2=1/2 => Rtot=1/4. Les performances des deux algorithmes de décodage sont tracées ainsi que la borne union du système pour le même code, soit : R=1/4 et K=3 (G_1=5, G_2=7, G_3=7, G_4=7).

La figure 4.23 représente les performances du système concaténé, constitué de deux codeurs identiquent récursifs et systématiques : K=4, R12/3, R2=1/2 => Rtot=1/3. Les performances des deux algorithmes de décodage sont tracées ainsi que la borne union du système pour le même code, soit : R=1/3 et K=4 (G_1=15, G_2=17, G_3=13).

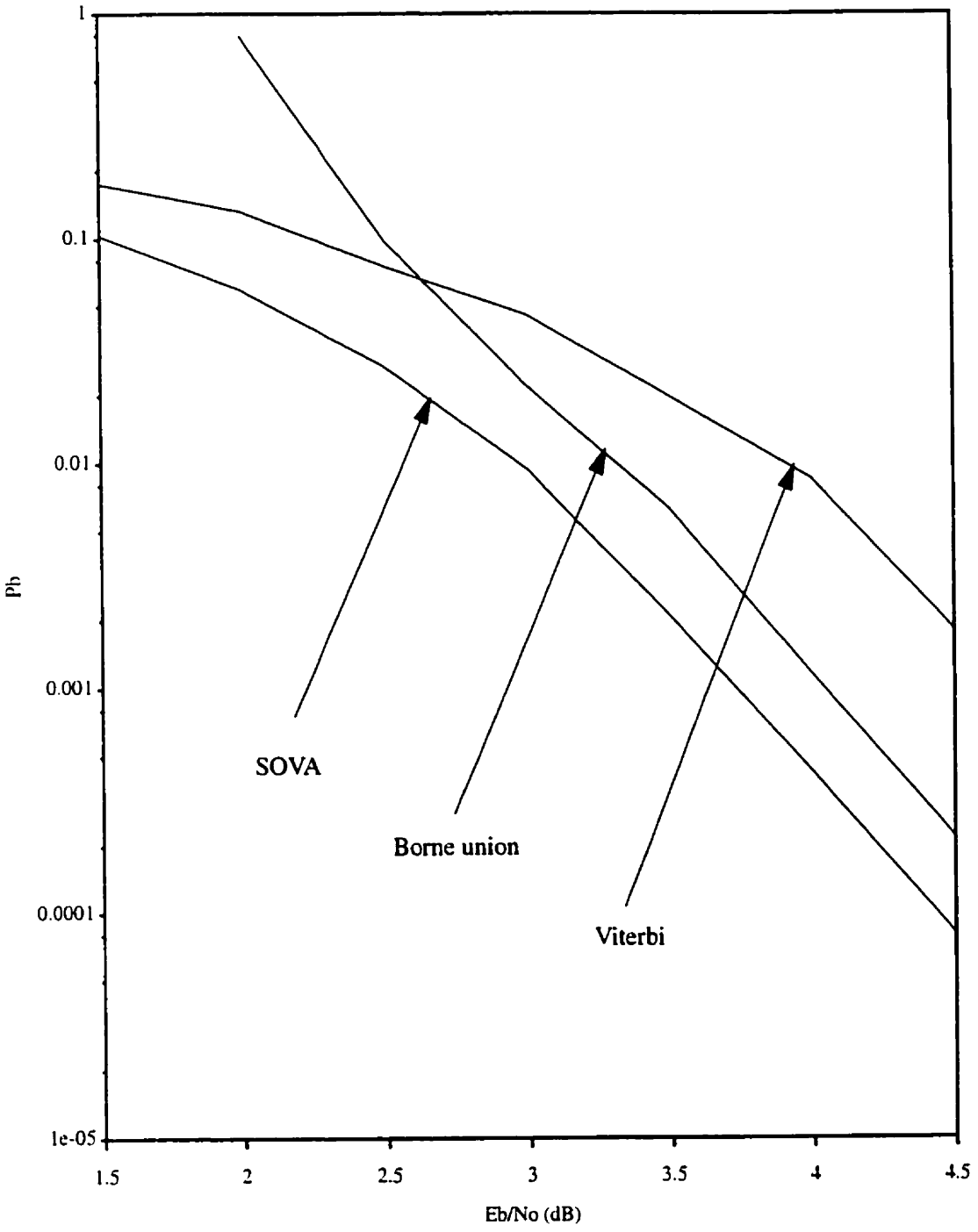

Figure 4.22 Comparaison des performances: algorithme SOVA vs Viterbi (K=3, R1=1/2, R2=1/2 => Rtot = 1/4; Codeur récursif et systématique)

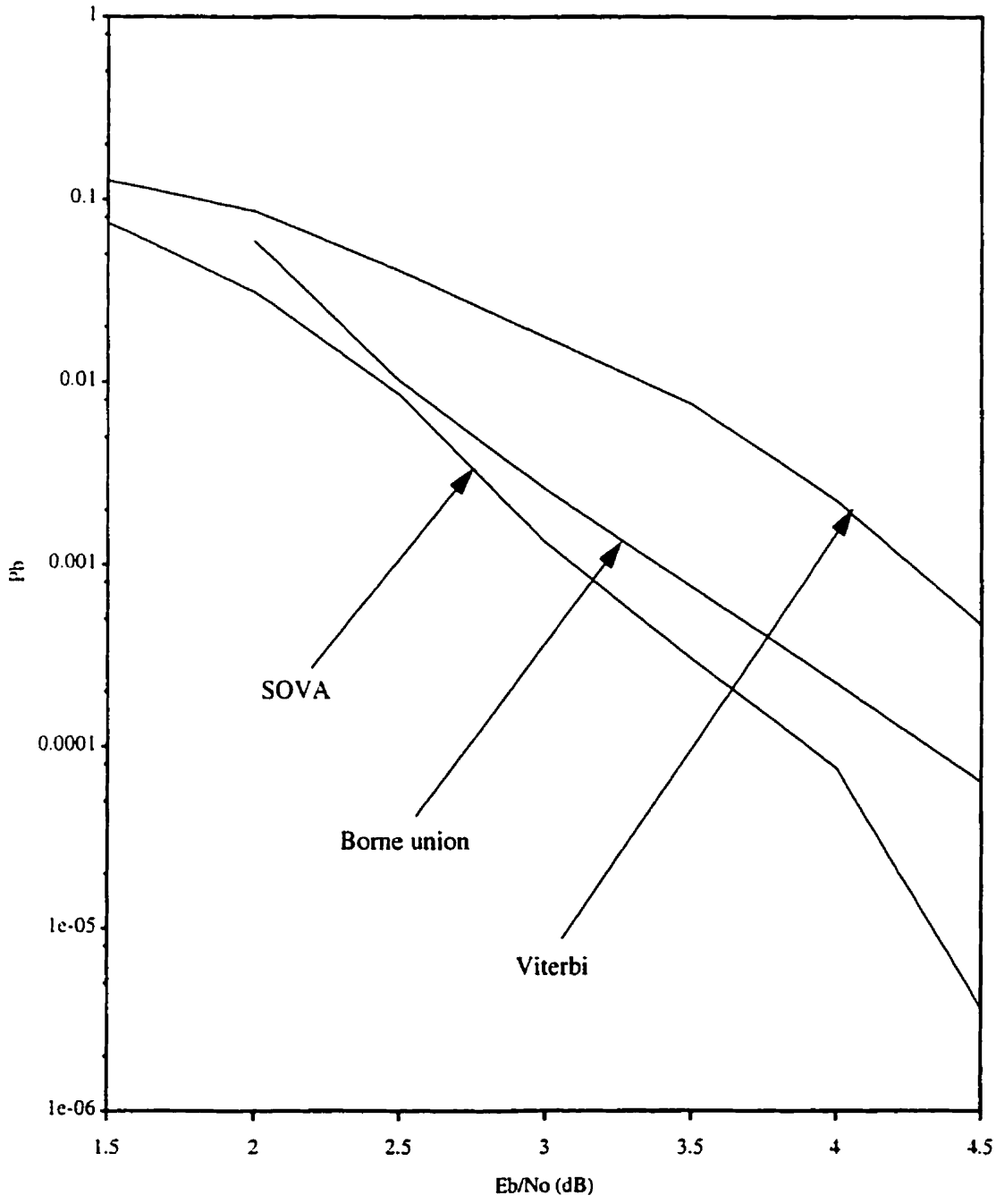

Figure 4.23 Comparaison des performances: algorithme SOVA vs Viterbi (K=4, R1=2/3, R2=1/2 => Rtot = 1/3; Codeur récursif et systématique)

Selon les résultats obtenus aux figures 4.22 et 4.23, on constate que l'algorithme SOVA apporte une amélioration des performances d'environ 1dB par rapport à l'algorithme de Viterbi. Cette amélioration provient de la présence dans l'algorithme SOVA de la mesure de la fiabilité de décision des symboles décodés.

D'un autre coté, les performances du système concaténé utilisant l'algorithme SOVA restent supérieures à la limite déterminée par la borne union pour un code de même taux R_{tot} (voir les figures 4.22 et 4.23).

En conclusion, l'algorithme de décodage SOVA procure une amélioration significative des performances d'un système concaténé en série par rapport à l'algorithme de Viterbi. Cependant, cette amélioration demeure moins importante que celle introduite par les codes itératifs, dits turbo [6],[17]. Selon les résultats obtenus en [6], la différence en termes de performances entre la première et la sixième itération peut varier entre 1.5 dB et 2.5 dB. Il ne faut pas oublier cependant, que la complexité d'un système «turbo» reste beaucoup plus grande que celle du système concaténé en série présenté dans ce travail.

Les résultats obtenus aux figures 4.22 et 4.23 sont confirmés par les simulations menées par [24] (figure 4.22) et [12],[13] et par [20] pour la figure 4.23.

4.2.3.4 RÉSULTATS OBTENUS POUR LES TAUX GLOBAL DU SYSTÈME : R_{tot}=1/2 et R_{tot}=1/3

R_{tot} = 1/2

Le taux de codage global du système désiré étant R_{tot}=1/2, les taux de codages R_1 et R_2 sont choisis comme suit:

$$R_1 = \frac{3}{4}, R_2 = \frac{2}{3} \Rightarrow R_{tot} = R_1 R_2 = \frac{1}{2}$$

Ce choix de taux de codage préserve la relation désirée entre les taux de codage (voir section 4.1.2), $R_2 < R_1$. Rappelons à ce stade que le taux de codage d'origine est toujours : $R_0 = 1/2$.

En respectant le choix des patrons de perforations [2] qui mènent aux meilleurs codes pour une longueur de contrainte donnée, on choisit les patrons de perforations suivants pour les longueurs de contraintes K=4 et K=5:

$$K = 4, P_1 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}, P_2 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

$$K = 5, P_1 = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}, P_2 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

Les patrons de perforation notés P_1 et P_2 correspondent à des taux de codage R_1 = 3/4 et R_2 = 2/3 respectivement.

R_{tot} = 1/3

Le taux de codage global du système désiré étant R_{tot}=1/3, les taux de codages R_1 et R_2

sont choisis comme suit :

$$R_1 = \frac{2}{3}, R_2 = \frac{1}{2} \Rightarrow R_{tot} = R_1 R_2 = \frac{1}{3}$$

Ce choix des taux de codage préserve la propriété désirée (voir section 4.1.2, $R_2 < R_1$.
En respectant le choix des patrons de perforations [2] qui mènent aux meilleurs codes pour une longueur de contrainte donnée, on choisit les patrons de perforations suivants pour les longueurs de contraintes K=4 et K=5:

$$K = 4, P_1 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, P_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$K = 5, P_1 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, P_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Les patrons de perforation notés P_1 et P_2 correspondent à des taux de codages $R_1 = 2/3$ et $R_2 = 1/2$ respectivement.

En examinant les matrices de perforations P_1 et P_2 pour les deux taux de codage (R_{tot}=1/2 et R_{tot}=1/3), on constate que certaines de ces matrices ne respectent pas la propriété exigée par les codes récursifs systématiques, soit que tous les éléments de la première colonne doivent être égaux à 1.
Sachant que ces matrices de perforation produisent les codes menant à de meilleures performances, les codes convolutionnels (non-récursifs) seront utilisés dans les simulations qui suivent. Il faudra pourtant souligner que les codes récursifs systématiques apportent une amélioration des performances du système concaténé à de faibles rapports E_b/N_0. Or, on est dans l'impossibilité d'utiliser ces derniers on gardant les mêmes conditions de simulations, c'est-à-dire en utilisant un seul type de codeur. Pourtant, dans tous les cas applicables - pour toutes les formes de matrices de perforations ayant tous les éléments de

la première colonne égaux à 1 - les performances du système utilisant les deux types de codeurs seront comparées. Plus précisément, c'est uniquement pour un taux de codage global R_{tot}=1/3 (K=4 et K=5) qu'on retrouve des matrices de perforation pouvant être appliquées en même temps aux codes convolutionnels et aux codes récursifs systématiques. La raison pour cela est que la propriété du patron de perforation préservant tous les bits systématiques est respectée.

On compare d'abord les performances du système concaténé en utilisant les deux types de codeurs (convolutionnel et récursif systématique), pour les cas applicables suivants:

1) La figure 4.24 représente les performances du système avec les paramètres suivants:

$$K = 4, P_1 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, P_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

2) La figure 4.25 correspond aux performances du système définit par:

$$K = 5, P_1 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, P_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Le même système est simulé en utilisant le codeur convolutionnel et le codeur récursif systématique.

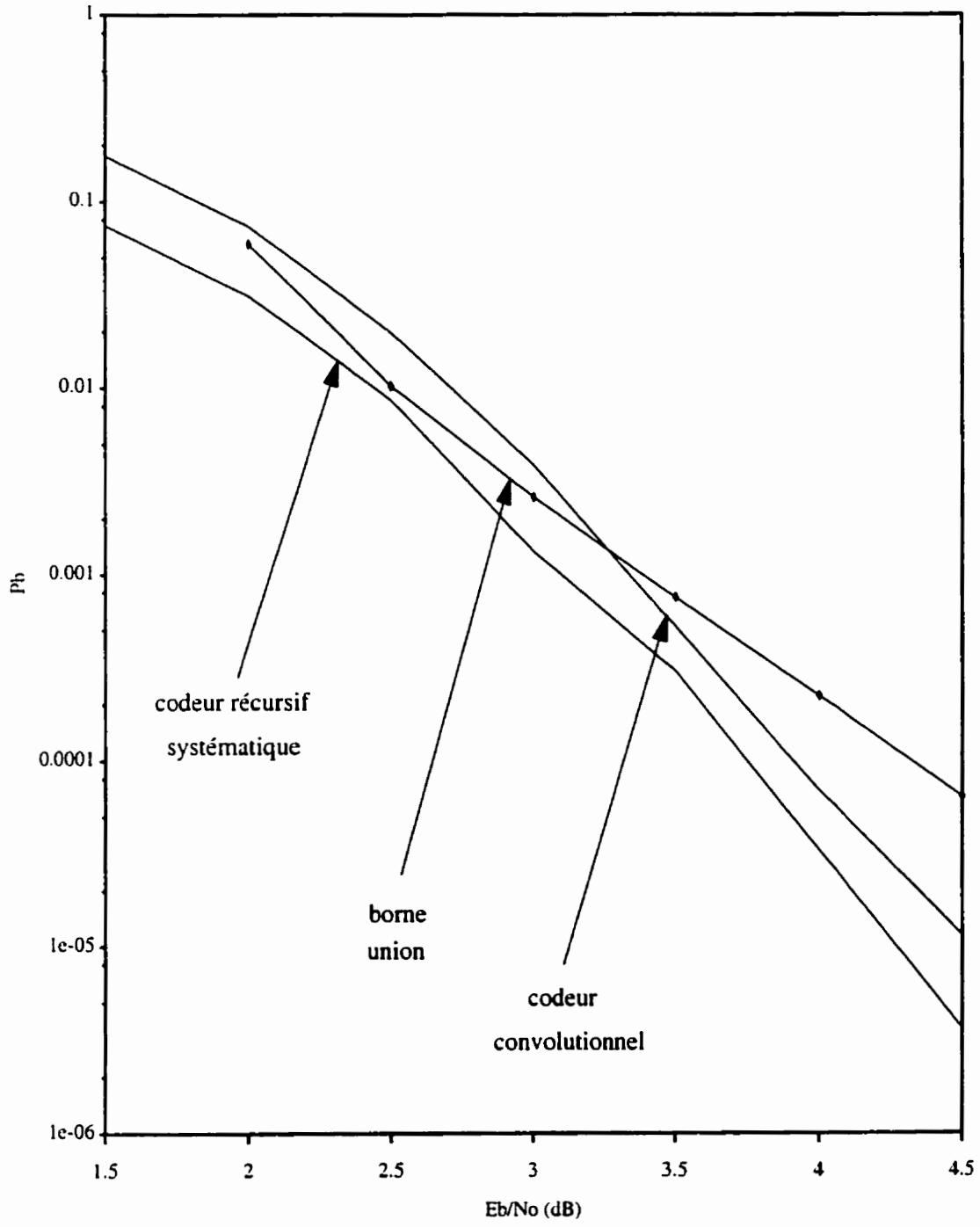

Figure 4.24 Performances du système en fonction du type de codeur utilisé; L'algorithme du décodage utilisé est SOVA (K=4, R1=2/3, R2=1/2 => Rtot = 1/3)

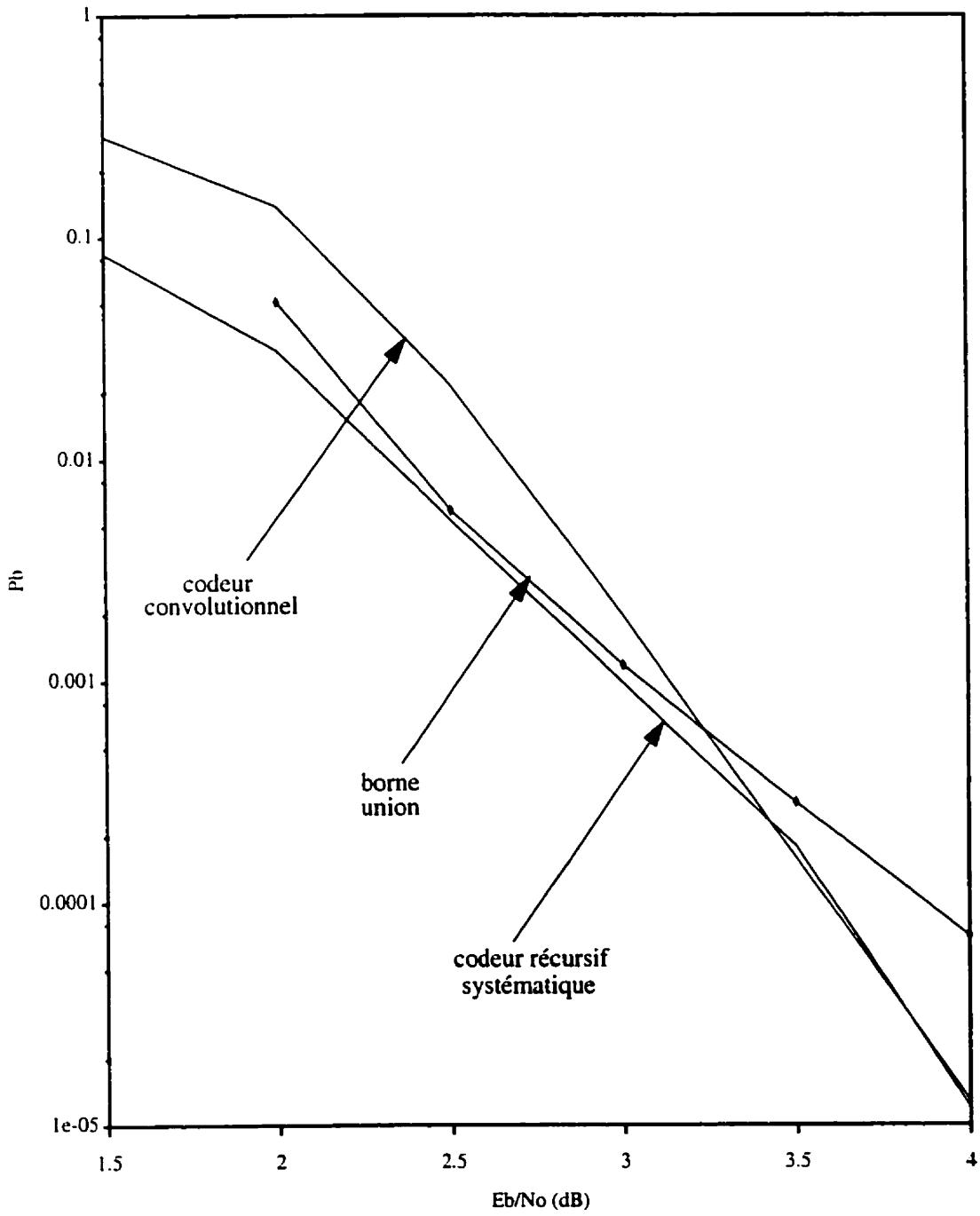

Figure 4.25 Performances du système en fonction du type de codeur utilisé; L'algorithme du décodage utilisé est SOVA (K=5, R1=2/3, R2=1/2 => Rtot = 1/3)

Il parait évident en observant les figures 4.24 et 4.25 que les codes récursif systématiques procurent de meilleures performances à des faibles rapport E_b/N_0.

La borne supérieure sur la probabilité d'erreur pour le code de taux R = 1/3 est tracée pour une longueur de contrainte K=4 à la figure 4.24 et pour K=5 à la figure 4.25.
On peut remarquer que la concaténation en série de deux codes convolutionnels ne semble pas être avantageuse à des faibles rapports E_b/N_0. Plus précisément, pour un taux de codage global du système fixé à R_{tot} = 1/3, la concaténation en série de deux codes convolutionnels avec K=4 et K=5, ne conduit à des résultats intéressants que si le rapport signal à bruit est suffisamment grand (ici, de l'ordre de 3dB).

Quant au taux de codage du système global R_{tot} = 1/2, les performances du système sont reproduites aux figures 4.26 et 4.27 pour les mêmes longueurs de contraintes K=4 et K=5 respectivement et pour les matrices de perforation qui leurs sont spécifiques.

En examinant ces courbes, on constate que l'utilisation des codes convolutionnels dans un système concaténé en série ne devient avantageux qu'à des rapports signal à bruit suffisamment grands (ici, de l'ordre de 4dB). Il faut noter que les performances d'un tel système varient en fonction des choix de la matrice de perforation et de la longueur du bloc d'information ainsi que du type de codeur utilisé. Dans tous les cas, à des faibles rapports E_b/N_0, il sera fort avantageux d'appliquer le décodage itératif [6],[17] qui apportera une amélioration significative des performances.

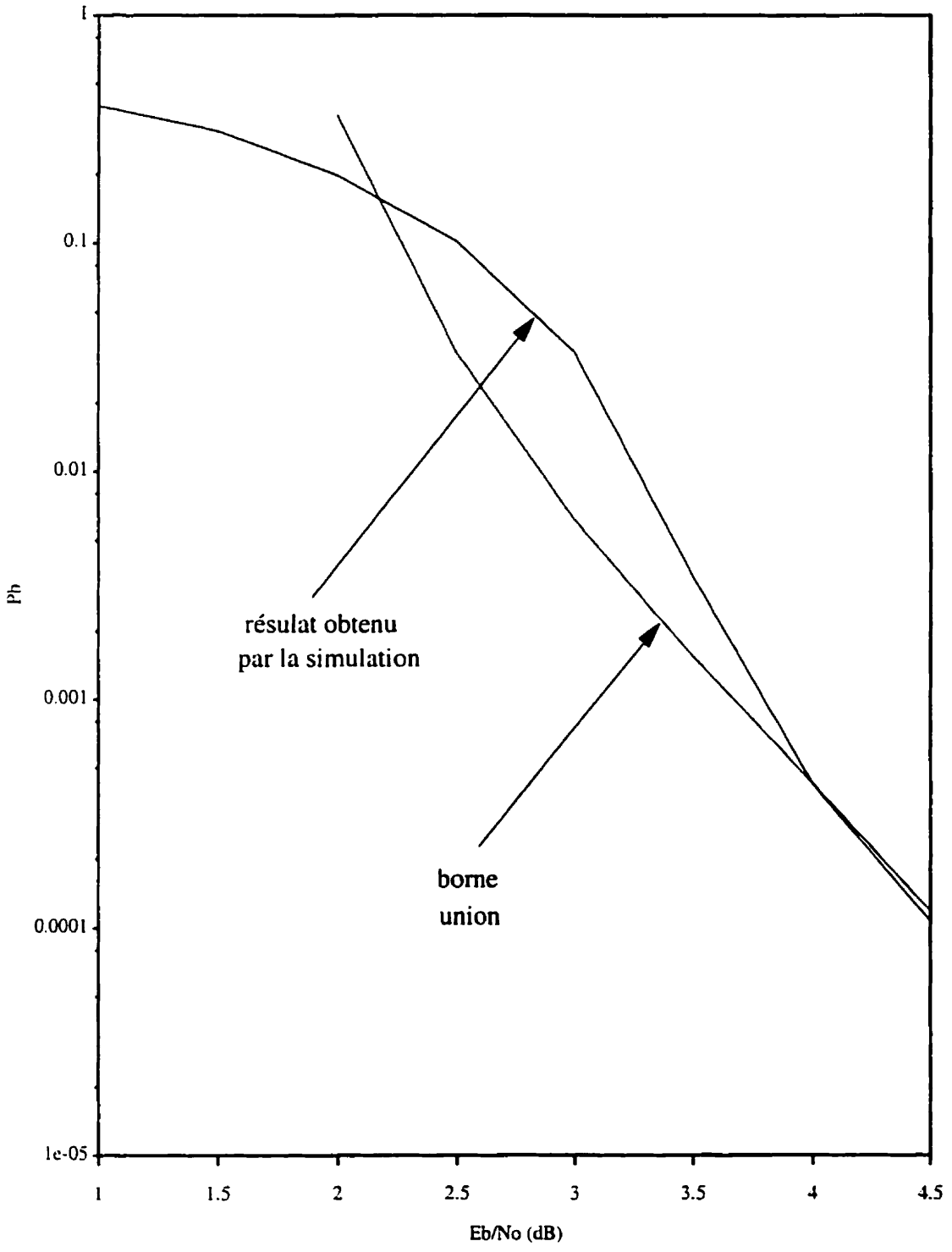

Figure 4.26 Probabilité d'erreur par bit (K=4, R1=3/4, R2=2/3 => Rtot = 1/2)

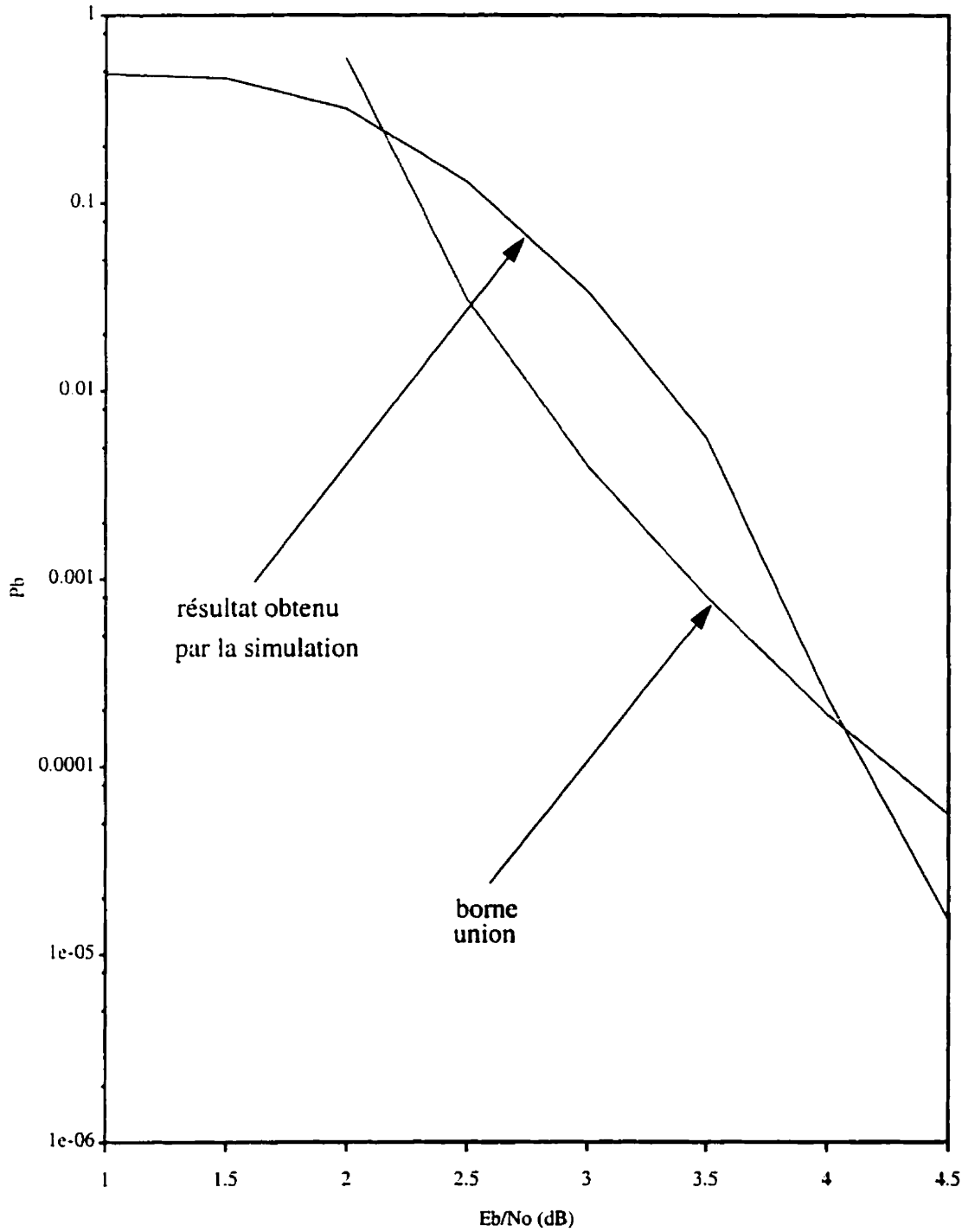

Figure 4.27 Probabilité d'erreur par bit (K=5, R1=3/4, R2=2/3 => Rtot = 1/2)

En conclusion, on peut déduire qu'un système concaténé en série utilisant l'algorithme de SOVA comme algorithme de décodage présentera les qualités et les défauts suivants:

- Système pouvant facilement s'adapter à des taux de codage différents en ne changeant que les matrices de perforation.

- Système facile à réaliser car un seul type de codeur/décodeur est requis.

- Système dont les performances sont influencées par le choix de différents paramètres comme : type de codeur (convolutionnel ou récursif systématique), longueur de bloc d'information, patrons de perforation appliqués.

- Système présentant de meilleures performances par rapport au cas où l'algorithme de Viterbi est utilisé mais reste désormais légèrement plus complexe à mettre en oeuvre que ce dernier.

De plus, les performances de ce système ont tendance à augmenter davantage lorsque les codes récursifs systématiques sont utilisés ou lorsque le rapport signal à bruit est élevé.

CHAPITRE 5

CONCLUSION ET SUGGESTIONS POUR RECHERCHES FUTURES

Ce mémoire a été consacré à la recherche d'un ensemble de codes convolutionnels perforés à des taux de codages compatibles et croissants, ainsi qu'à l'utilisation des codes convolutionnels perforés dans un système concaténé en série. Les performances du système concaténé en série ont été évaluées en fonction du type de décodeur utilisé.

Les notions liées au codage et au décodage des codes convolutionnels ont été introduites au chapitre 2. Les deux types de codage, tel que le codage convolutionnel non systématique et le codage récursif systématique ont ainsi été présentés. Le décodage à maximum de vraisemblance et en plus particulier le décodage de Viterbi ont été décrits. Cet algorithme a fait l'objet d'une étude de performance au chapitre 4.

Au chapitre 3, on a effectué une recherche de bons codes perforés à taux croissants et compatibles, obtenus à partir du code d'origine de taux $R_0=1/6$. Un ensemble de «meilleurs» codes perforés à taux croissants ($1/6 < R < 7/8$) et compatibles a alors été trouvé et leurs performances examinées pour chacun des taux désirés et pour les deux longueur de contrainte, K=7 et K=8. La méthode de recherche qu'on a proposée afin de déterminer l'ensemble des codes perforés obtenus n'est pas une méthode optimale, car le choix des patrons de perforation est restreint par la propriété de compatibilité. Pourtant, on constate que les performances de l'ensemble des codes perforés obtenus à l'aide de cette

méthode, sont comparables avec celles des codes perforés non-compatibles de mêmes taux de codage [11].

Le chapitre 4 a été consacré à l'étude du système concaténé en série en présence des codes convolutionnels perforés, dont le décodage s'effectue selon l'algorithme SOVA. Les performances d'un tel système ont été évaluées en fonction des différents paramètres qui le constituent, soient le type du codeur, la longueur de bloc d'information, les patrons de perforation utilisés, etc... Une longueur de bloc d'information de 500 bits a donc été choisie comme celle apportant une amélioration de performances, tout en évitant d'augmenter significativement le délai du décodage provenant de la concaténation sérielle. On remarque une amélioration des performances du système concaténé lorsque le codeur récursif systématique est appliqué, par rapport au cas d'un codeur convolutionnel non-systématique. Cette amélioration est plus significative à des faibles rapports signal à bruit, et elle varie de 0 à 0.3 dB en fonction du rapport signal à bruit ainsi que des patrons de perforation utilisés.

D'un autre coté, en examinant les améliorations des performances que l'algorithme SOVA dans un système concaténé apporte par rapport à la borne union pour un même taux de codage, on conclut que les performances du système concaténé ont tendance à augmenter davantage lorsque les codes récursifs systématiques sont utilisés ou lorsque le rapport signal à bruit est élevé. Plus précisément, on remarque que la concaténation en série de deux codes convolutionnels non-systématiques ne conduit à des résultats intéressants que si le rapport E_b/N_0 est suffisamment grand, de l'ordre de 3 à 4 dB - dépendant des longueurs de contraintes et des codes utilisés. Les codes récursifs systématiques choisis dans nos simulations apportent une amélioration de performances du système concaténé

par rapport à la borne supérieure pour un même taux de codage, contrairement aux codes non-systématiques, sur toute la plage de E_b/N_0.

Il faut noter que les performances d'un système concaténé semblent être fortement influencées par le choix des patrons de perforation, par la longueur de contrainte utilisée ainsi que par la variation du rapport signal à bruit considéré dans les simulations. Par conséquent, comme recherches futures, on peut suggérer d'approfondir davantage l'analyse de la concaténation sérielle. En particulier, une analyse basée sur les propriétés algébriques des codes convolutionnels constituant le système concaténé pourrait offrir une réponse plus précise sur le comportement d'un tel système. Il serait ainsi profitable, à partir d'une telle analyse de:

1) déterminer de façon algébrique la variation du rapport signal à bruit d'intérêt, à partir de la longueur de contrainte utilisée.

2) proposer les combinaisons des patrons de perforation conduisant à des meilleures performances pour une longueur de contrainte donnée.

3) considérer l'utilisation des codes convolutionnels à des taux compatibles dans un système concaténé en série.

RÉFÉRENCES

[1]. BÉGIN, G., HACCOUN, D. et PAQUIN, C., «Further Results on High-Rate Punctured Convolutional Codes for Viterbi and Sequential Decoding», IEEE *Trans. on Comm.*, vol. 38, No. 11, Nov. 1990.

[2]. BÉGIN, G. et HACCOUN, D., «High-Rate Punctured Convolutional Codes: Structure Properties and Construction Techniques», IEEE *Trans. on Comm.*, vol. COM-37, pp. 1381-1385, Dec. 1989.

[3]. BÉGIN, G. et HACCOUN, D., «Sequential decoding of punctured convolutional codes», in *Proc. 13th Biennial Symp. Commun.*, Kingston, Ont., Canada, June 1986, pp. A.3.5-A3.8.

[4]. BENEDETTO, S., MONTORSI, G., DIVSALAR, D. et POLLARA, F., «Serial Concatenation of Interleaved Codes: Performance Analysis, Design, and Iterative Decoding», *TDA Progress Report 42-126.*

[5]. BHARGAVA, V., HACCOUN, D., MATYAS, R. et NUSPL, P., *Digital Communications by Satellite*, John Wiley, New York, 1981.

[6]. BOUZOUITA N., «Sur le décodage itératif des Codes Turbo», Mémoire de Maitrise, Département de Génie Électrique et de Génie Informatique, École Polytechnique de Montréal, Montréal, Juin 1997.

[7]. CAIN, J. B., CLARK, C. et GEIST, J. «Puncured convolutional codes of rate (n-1)/n and simplified maximum likelihood decoding» IEEE *Trans. Inform. Theory*, vol. IT-25, pp. 97-100, Jan. 1979.

[8]. FORNEY, G. D. , Jr., «Concatenated Codes», Cambridge, MA : M.I.T. Press, 1966.

[9]. FORNEY, G. D. "Convolutional codes: algebraic stucture", *IEEE Trans. Inform.*

Theory, vol. IT-16, 1970, pp. 720-738.

[10]. GALLAGER, R.G., *Information Theory and Reliable Communication*, John Wiley, New York, 1968.

[11]. HACCOUN, D. et BÉGIN, G., «High-Rate Punctured Convolutional Codes for Viterbi and Sequential Decoding», IEEE *Trans. on Comm.*, vol. 37, No. 11, pp. 1113-1125, Nov. 1989.

[12]. HAGENAUER, J. et HOEHER, P., «A Viterbi Algorithm with Soft-Decision Outputs and its Applications», *Proc. IEEE Globecom Conf.* (Dallas, TX, Nov. 1989), pp. 1680-1686.

[13]. HAGENAUER, J. et HOEHER, P., «Concatenated Viterbi Decoding», *Fourth Joint Swedish-Soviet Int. Workshop on Inf. Theory*, Gotland Sweden, Studentlitteratur, Lund, pp. 29-33, Aug. 1989.

[14]. HAGENAUER, J., OFFER, E. et PAPKE L., «Iterative Decoding of Binary Block and Convolutional Codes», IEEE *Transactions on Information Theory*, Vol. 42, No. 2, pp. 429-445, March 1996

[15]. JUSTESEN, J., THOMMESEN C. et ZYABLOV V. V., «Concatenated Codes with convolutional inner codes», IEEE *Transactions on Information Theory*, Vol. IT-34, pp.1217-1225, Sept. 1988.

[16]. LAVOIE, D., «Détection des Codes Convolutionnels Catastrophiques», Rapport d'été, Département de Génie Électrique et de Génie Informatique, École Polytechnique de Montréal, Montréal, Sept. 1991.

[17]. OSSEIRAN A., «Sur le décodage des Codes Turbo», Mémoire de Maitrise, Département de Génie Électrique et de Génie Informatique, École Polytechnique de Montréal, Montréal, Septembre 1999.

[18]. PAQUIN C., «Algorithmes de Détermination de Spectres des Codes Convolution-

nels Perforés», Mémoire de Maîtrise, Département de Génie Électrique et de Génie Informatique, École Polytechnique de Montréal, Montréal, Novembre 1990.

[19]. PROAKIS, J. K., Digital Communications, McGraw-Hill Book Company, New York, 1989.

[20]. Rislow, B., Maseng, T. et Trandem, O., «Soft Information in Concatenated Codes», IEEE *Transactions on Communication*, Vol. 44, No. 3, pp. 284-2286, March 1996.

[21]. TSAI, S., «Interleaving and error-burst distribution», IEEE *Transactions on Communication*, Vol. COM-20, pp. 291-296, Jun. 1972.

[22]. VITERBI, A.J., «Convolutional Codes and their Performance in Communication Systems», IEEE *Trans. Commun. Technol.*, vol. COM-19, pp.751-772, Oct. 1971.

[23]. VITERBI, A. J., et OMURA, J. K., Principles of Digital Communication and Coding, McGraw Hill, New York, 1979.

[24]. Wang, X. et Wicker, B., Stephen, «A Soft-Output Decoding Algorithm for Concatenated Systems», IEEE *Transactions on Information Theory*, Vol. 42, No.2, pp.543-553, March 1996.

[25]. YASUDA, Y., KASHIKI, K. et HIRATA Y., «High-Rate Punctured Convolutional Codes for Soft-Decision Viterbi Decoding», IEEE *Trans. on Comm.*, vol. COM-32, pp.315-319, Mar. 1984.

[26]. ZERONG Y., «Analyse et performances des codes convolutionnels récursifs systématiques», Mémoire de Maitrise, Département de Génie Électrique et de Génie Informatique, École Polytechnique de Montréal, Montréal, Decembre 1998.