UNIVERSITÉ DE MONTRÉAL

ANALYSE ET PERFORMANCE DES CODES CONVOLUTIONNELS RÉCURSIFS SYSTÉMATIQUES

ZERONG YU
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)
NOVEMBRE 1998

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

ANALYSE ET PERFORMANCE DES CODES CONVOLUTIONNELS RÉCURSIFS SYSTÉMATIQUES

présenté par: YU Zerong

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

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

M. CONAN Jean, Ph.D., président

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

M. GAGNON François, Ph.D., membre

A Jun et Anne

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, Professeur 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 et sa curiosité scientifique m'a poussé à approfondir mon sujet de recherche et mes connaissances.

J'aimerais aussi remercier mes collègues de travail qui ont su créer une ambiance particulièrement agréable et intéressante loin des tracas professionnels. Merci à Christian, Melita, Pierre, Afif, Jean-Émile et Naoufel.

Je tiens finalement à remercier mes proches et mes amis pour leur soutien durant tout la période de ma recherche. Je pense en particulier à mes parents, Shijun YU et Shide TAN.

RÉSUMÉ

Les codes convolutionnels forment une classe de codes correcteurs d'erreur qui offrent des performances d'erreur. Parmi les meilleurs dans ces codes, on retrouve les codes convolutionnels systématiques et récursifs. Un code convolutionnel récursif systématique peut être construit à partir de la transformation d'un code non systématique en un code systématique. Cette transformation implique que le codeur utilise une boucle de retour interne. Il possède alors la même distance libre et le même nombre de chemins erronés, mais le nombre total de bits d'information en erreur sur ces chemins est différent. De plus, des résultats de simulations montrent qu'à de faibles rapports signal à bruit, la probabilité d'erreur d'un code convolutionnel récursif systématique est plus faible que celle d'un code convolutionnel non systématique.

La perforation est une technique qui permet la construction de codes convolutionnels de taux de codage élevé à partir de codes de faible taux de codage. Les codes perforés sont obtenus par l'élimination périodique (perforation) de symboles codés produits par un codeur de faible taux de codage. Les codes perforés possèdent donc la structure simple des codes de faible taux de codage dont ils sont dérivés mais tout en ayant les avantages des codes de taux de codage élevé quant à une puissance d'émission et une largeur de bande requises inférieures à celles des codes de faibles taux. De plus, il est possible de changer le code résultant de la perforation d'un code origine, en changeant simplement la façon de perforer les symboles, ce qui permet donc d'obtenir un code à taux variable.

Une nouvelle stratégie de corrections d'erreur consiste à utiliser la concaténation parallèle de deux codeurs convolutionnels récursifs systématiques (codeurs constituants) à travers un entrelaceur. Le codage CPCC (concaténation parallèle de codes convolutionnels) constitue une technique très puissante pour la détection et la correction d'erreurs. À de faibles rapports signal à bruit, les performances d'erreur obtenues avec ces nouveaux codes sont meilleures que celles des codes constituants.

L'analyse de la performance d'erreur des codes est établie à partir de leur spectre. Le spectre est la distribution du nombre de mots de codes ayant un poids de Hamming donné.

Ce mémoire s'adresse au problème de la détermination du spectre de plusieurs types de codes convolutionnels: récursifs systématiques, récursifs systématiques perforés, (CPCC). L'approche privilégiée est celle qui consiste à déterminer le spectre par une exploration de leur structure en arbre. Des méthodes pour déterminer le spectre d'un code convolutionnel et celui d'un code convolutionnel perforé ont déjà été développées sous la direction du professeur David Haccoun à la section télécommunication de l'École Polytechnique de Montréal. Les algorithmes utilisés dans ce mémoire sont tous basés sur ces méthodes mais sont adaptés au cas particulier des codes convolutionnels récursifs systématiques et des codes convolutionnels récursifs systématiques perforés. En apportant plusieurs changements à ces algorithmes, on arrive à un algorithme qui a pour but de calculer le spectre des codes CPCC. Ces algorithmes sont programmés en langage C et sont implantés sur un système UNIX d'un réseau SUN.

L'application de ces algorithmes nous a permis d'étendre les connaissances du spectre de plusieurs types de codes tel que les codes convolutionnels récursifs

systématiques, les codes convolutionnels récursifs systématiques perforés et CPCC).

L'analyse des performances d'erreur des codes CPCC montre l'importance du choix des codes constituants et de l'entrelaceur. La comparaison des performances d'erreur de codes convolutionnels avec celles des codes CPCC montre que ces derniers permettent encore d'obtenir de meilleurs résultats lorsque les paramètres des codes constituants et de l'entrelaceur sont choisis convenablement. A toute fin, nous proposons de nouveaux codes de taux variables résultant de la perforation des codes CPCC. Les calculs des spectres et des performances de ces codes, et la comparaison des performances d'erreur nous permettent de déterminer les meilleurs patrons de perforation.

ABSTRACT

Convolutional codes belong to a class of error correcting codes which provide good error performances. From this set of codes, there exist special codes called convolutional systematic and recursive codes. A convolutional systematic and recursive code may be constructed using a transformation from a non-systematic code to a systematic code. This transformation implies that the coder uses a feedback. It has the same free distance and the same numbers of paths, but the total number of information bits is different. Furthermore, simulation results show that at low signal-to-noise ratio, the error probability of a recursive systematic convolutional code is lower than the one of a non-systematic convolutional code.

Puncturing is a technique which allows to construct the high coding rate convolutional codes from the low coding rate codes. Punctured codes are obtained by periodic elimination (perforation) of code symbols produced by a low coding rate code. The punctured codes have the same simple structure than the low coding rate codes, but they keep the benefits of high-rate codes like the energy of emission and the bandwidth required. Furthermore, by simply changing the puncturing rule, different punctured codes may be obtained from a single original code, this allows to obtain variable-rate codes.

A new strategy of error correcting consist in using a parallel concatenation of two recursive systematic convolutional codes (constituent codes) through an interleaver. Among the well known error control techniques, the PCCC (parallel concatenated

convolutional codes) coding is the most powerful one. At low signal-to-noise ratios, the error performances of this new class of codes are better than its constituent codes.

The error performances analysis of codes is usually established from their spectrum. The spectrum is the distribution of number of code words and total weight of associated information sequences depending the Hamming weight of code words.

The objective of this thesis is to determine the spectra of several types of convolutional codes: recursive systematic, recursive systematic punctured, code (PCCC). The privileged approach is based on an exploration of code trees. Methods for determining the spectrum of convolutional code and the one of punctured code were developed under the supervisor of professor David Haccoun in the telecommunication section at Ecole Polytechnique de Montréal. The algorithms used in this thesis are based on these methods, but are adapted to recursive systematic convolutional codes and to recursive systematic convolutional punctured codes. After several modification of these algorithms, an algorithm has been developed to calculate the spectrum of PCCC codes. The algorithms have been programmed in C language and implemented on a UNIX system of a SUN network.

By application of these algorithms, we have extended our knowledge of the spectrum of several types of codes such as recursive systematic convolutional code, recursive systematic convolutional punctured code, and PCCC code.

The error performances analysis of PCCC codes shows the important of choice of constituent codes and the interleaver to obtain best results. The performance comparisons between convolutional codes and PCCC codes show that the last ones still give the

advantageous solutions when the parameters of constituent codes and of interleaver are adequately chosen. Finally, variable coding rate PCCC codes are examined. The determination of the spectrum and the error performances, and the comparison between error performances allow us to determine the best puncturing pattern.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ANNEXES

CHAPITRE 1

INTRODUCTION

Dans un système de communication numérique, les symboles transmis sont souvent perturbés par du bruit provenant de plusieurs origines. Ceci peut entraîner une décision incorrecte sur les symboles reçus et par conséquent, dégrader la fiabilité du système et du message reçu.

Un moyen efficace de détecter et corriger les symboles en erreur est d'utiliser la technique de codage correcteur d'erreur. Le principe du codage consiste à ajouter, selon des règles particulière, de la redondance aux messages avant de les transmettre [1]. Une mesure de la redondance introduite par ce codage est le taux de codage R qui représente le rapport entre le nombre de bits d'information à transmettre et le nombre de symboles effectivement transmis dans le canal.

Il existe deux familles de codage: le codage en bloc et le codage convolutionnel. Le codage convolutionnel est une technique appropriée à une source qui génère l'information de façon continue. Il est bien connu que, dans un canal sans mémoire, un code convolutionnel avec décodage probabiliste peut fournir un gain de codage substantiel, permettant de s'approcher de la limite de la fiabilité prédite par la théorie de Shannon [8].

Les codes convolutionnels figurent parmi les codes les plus utilisés, notamment

pour des applications, telles que les systèmes de communications personnelles sans fil, les communications par satellite, etc., où une très faible probabilité d'erreur par bit est requise. Cela a été rendu possible grâce à l'utilisation d'un algorithme de décodage puissant et pratique appelé algorithme de Viterbi. Cet algorithme à maximum de vraisemblance est optimal et permet de minimiser la probabilité d'erreur par séquence.

Parmi les applications des codes convolutionnels, celles utilisant un faible taux de codage ($R = 1/v$) sont les plus répandues. Pour ces taux de codage, des codes optimaux sont connus [21] [22] [23] et des techniques de décodage performantes sont mises en application [2] [8].

A partir d'un code convolutionnel, nous pouvons construire un nouveau code avec une boucle de retour interne, dit code convolutionnel récursif systématique. A de faibles rapports signal à bruit, la probabilité d'erreur d'un code convolutionnel récursif systématique est plus faible qu'un code convolutionnel non systématique. Cet avantage est conservé pour l'étude des codes CPCC.

Par ailleurs, les codes convolutionnels de taux de codage élevé ($R = b/v$) peuvent assurer une bonne performance d'erreur dans un canal sans mémoire, et leur utilisation s'avère particulièrement avantageuse lorsque la largeur de bande disponible est faible. Toutefois, l'utilisation des codes convolutionnels de taux de codage élevé a été limitée en pratique parce que l'application des techniques de décodage probabiliste devient rapidement impraticable avec l'augmentation du taux de codage. Cet inconvénient peut être évité par l'utilisation des codes convolutionnels perforés.

Les codes convolutionnels perforés peuvent être générés à partir d'un code origine

de taux de codage faible ($R = 1/v$) par perforation périodique, ce qui entraîne un avantage important: ils peuvent être décodés tout comme les codes de taux de codage faible avec une modification minimale du décodeur. Les codes perforés ouvrent la voie à de nouvelles applications telles l'encodage et le décodage à taux variables.

Une nouvelle stratégie de correction d'erreur a été récemment proposée par Berrou *et al.* [15]. Cette stratégie implique une concaténation parallèle de deux codeurs convolutionnels récursifs systématiques séparés par un entrelaceur. L'évaluation des performances d'erreur a déjà été obtenue par Benedetto et Montorsi [6] pour une concaténation parallèle de codes blocs et pour celle de codes convolutionnels. Avec une autre méthode proposée par Podemski, Holubowicz, Berrou, Glavieux [7], on a aussi obtenu la borne supérieure sur la probabilité d'erreur par bit pour un canal à bruit blanc gaussien et additif et un décodeur à maximum de vraisemblance. A de faibles rapports signal à bruit, la performance de cette nouvelle classe de codes est meilleure que celle des codes constituants.

Outre celui de trouver de bons codes, un des problèmes difficiles de la théorie de codage est de déterminer la probabilité d'erreur d'un code donné, c'est-à-dire la probabilité que le message décodé diffère de celui émis par la source. Comme nous le verrons, le calcul se réduit à celui du spectre du code. L'évaluation de ce spectre constitue toutefois un problème ardu. Elle peut se faire à partir d'un diagramme d'état du code ou en utilisant une exploration dans l'arbre représentant le code. Cette dernière approche est privilégiée dans le cadre de la recherche dont nous présentons les résultats dans ce mémoire.

Des méthodes pour déterminer le spectre d'un code convolutionnel et de celui d'un

code convolutionnel perforé ont déjà été développées sous la direction du professeur David Haccoun à la section télécommunication [19] [20]. Les algorithmes utilisés ici sont tous basés sur ces méthodes mais sont adaptés au cas particulier des codes convolutionnels récursifs systématiques et des codes convolutionnels récursifs systématiques perforés. En apportant plusieurs changements à ces algorithmes, on arrive à un algorithme qui a pour but de calculer le spectre des codes CPCC (concaténation parallèle de codes convolutionnels).

Les recherches effectuées durant ce projet ont produit les contributions suivantes:

- L'évaluation par simulations sur ordinateur des spectres de distance et des bornes sur les performances d'erreur des codes convolutionnels récursifs systématiques de taux de codage $R = 1/2$ et de longueurs de contrainte K allant de 3 à 16, et de taux de codage $R = 1/3$ et de longueurs de contrainte K allant de 3 à 9.

- L'évaluation par simulations sur ordinateur des spectres de distance et des bornes sur les performances d'erreur des codes convolutionnels récursifs systématiques perforés de taux de codage R allant de 2/3 à 7/8 et de longueurs de contrainte K allant de 3 à 9.

- L'évaluation par simulations sur ordinateur des spectres de distance et des bornes sur les performances d'erreur des codes CPCC (concaténation parallèle des codes convolutionnels récursifs systématiques) de taux de codage de deux codes constituants $R_1 = R_2 = 1/2$, $K = 3$, $G = (1,7/5)$, et de longueurs de l'entrelaceur N égales à 16, 32, 64, 128, 192.

- L'évaluation par simulations sur ordinateur des spectres de distance et des bornes

sur les performances d'erreur des codes CPCC (concaténation parallèle des codes convolutionnels récursifs systématiques) perforés de taux de codage $R = 2/4, 4/6, 6/8, 8/10$ et de longueur de contrainte $K = 3$.

Ce mémoire est structuré comme suit. Le chapitre 2 décrit les caractéristiques de base de différents types du codage convolutionnel, tel que le codage convolutionnel non systématique, le codage convolutionnel récursif systématique et le codage CPCC (concaténation parallèle de codes convolutionnels). Dans ce chapitre, on y aborde aussi la structure des codeurs ainsi que les propriétés de distance des codes convolutionnels. Il sera aussi question d'une technique pour la correction ou la détection des erreurs, de l'évaluation des performances d'erreurs et des techniques de décodage.

Le chapitre 3 est consacré à l'étude du spectre. L'évaluation des performances d'erreur est effectuée à partir des relations entre le spectre et la probabilité d'erreur. Deux approches essentielles pour déterminer le spectre sont analysées: calculer avec la fonction de transfert et rechercher dans l'arbre du code.

Le chapitre 4 décrit les principes du codage convolutionnel récursif systématique, la structure du codeur. La relation qui existe entre le code convolutionnel récursif systématique et le code convolutionnel non systématique sera explicitée. Nous démontrons le spectre de distance et la probabilité d'erreur des codes convolutionnels récursifs systématiques.

Les codes convolutionnels perforés sont une classe de codes convolutionnels de taux de codage élevé $R = b/v$ qui sont obtenus à partir de codes convolutionnels de faible taux de codage $R_0 = 1/v_0$, par l'élimination périodique (perforation) de certains symboles

codés à la sortie de ce codeur de faible taux de codage. Au chapitre 5, nous présentons le principe de la perforation, le codeur, le spectre et le performances d'erreur du code perforé.

Dans le chapitre 6, nous présentons la structure de codage CPCC (concaténation parallèle de codes convolutionnels). Nous y présentons aussi le spectre, la représentation en hyper-treillis et en arbre, les performances d'erreur des codes CPCC.

Finalement, la conclusion de ce mémoire et un nombre de perspectives de recherche sont présentés au chapitre 7.

CHAPITRE 2

CODES CONVOLUTIONNELS

Les systèmes de communication numérique servent à transmettre les messages d'une source à une destination. Pour ces systèmes, différentes techniques ont été mises au point afin de protéger l'intégrité de l'information transmise dans le canal de communication. Le codage convolutionnel est l'une de ces techniques.

Le codage correcteur d'erreur est une technique permettant d'augmenter la fiabilité des systèmes de communication numérique. Son principe est basé sur la redondance de l'information. La technique de codage consiste à envoyer avec l'information utile des symboles supplémentaires appelés symboles de parité calculés selon une loi bien définie et connue du récepteur comme de l'émetteur.

Il existe deux grandes catégories de codage: le codage en bloc et le codage convolutionnel [1] [2]. Le codage en bloc consiste à coder des blocs d'information de façon indépendante. On ajoute un certain nombre de symboles de parité à la séquence d'information. Dans le codage convolutionnel, l'information est codée de façon continue. Dans ce cas, une séquence de longueur indéterminée de symboles d'information est transformée en une séquence de longueur indéterminée de symboles codés. Pour une complexité donnée, le codage convolutionnel est la technique la plus efficace pour la correction des erreurs [3].

Ce chapitre décrit les caractéristiques de base de différents types du codage convolutionnel, tel que le codage convolutionnel non systématique, le codage convolutionnel récursif systématique, le codage convolutionnel concaténé parallèle de codes convolutionnels, la structure des codeurs, les propriétés de distance des codes convolutionnels, une technique pour la correction ou la détection des erreurs, les performances d'erreurs, et la technique de décodage.

2.1 SYSTÈME DE COMMUNICATION NUMÉRIQUE

Le modèle d'un système de communication numérique avec codage et décodage est illustré à la figure 2.1. Dans ce modèle une source transmet un message à une destination dans un canal qui ajoute du bruit blanc gaussien.

D'une façon générale, le codeur associe à toute séquence $\underline{U}$ de b bits d'information une séquence $\underline{X}$ de v symboles codés, telle que v soit plus grand que b. Le bruit qui affecte le canal de transmission est généralement représenté par un processus aléatoire blanc et gaussien dénoté $\underline{N}$. Le récepteur reçoit la séquence bruité $\underline{Y} = \underline{X} + \underline{N}$. La tâche du décodeur est de minimiser la probabilité d'erreur lors de l'estimation de $\underline{U}$ par $\underline{\hat{U}}$. Les valeurs de $\underline{Y}$ sont habituellement quantifiées à la sortie du démodulateur. Les définitions de ces valeurs seront présentées à la section 2.3.

Si les symboles transmis dans le canal sont statistiquement indépendants, le canal est dit sans mémoire (en anglais Discrete Memoryless Channel ou DMC).

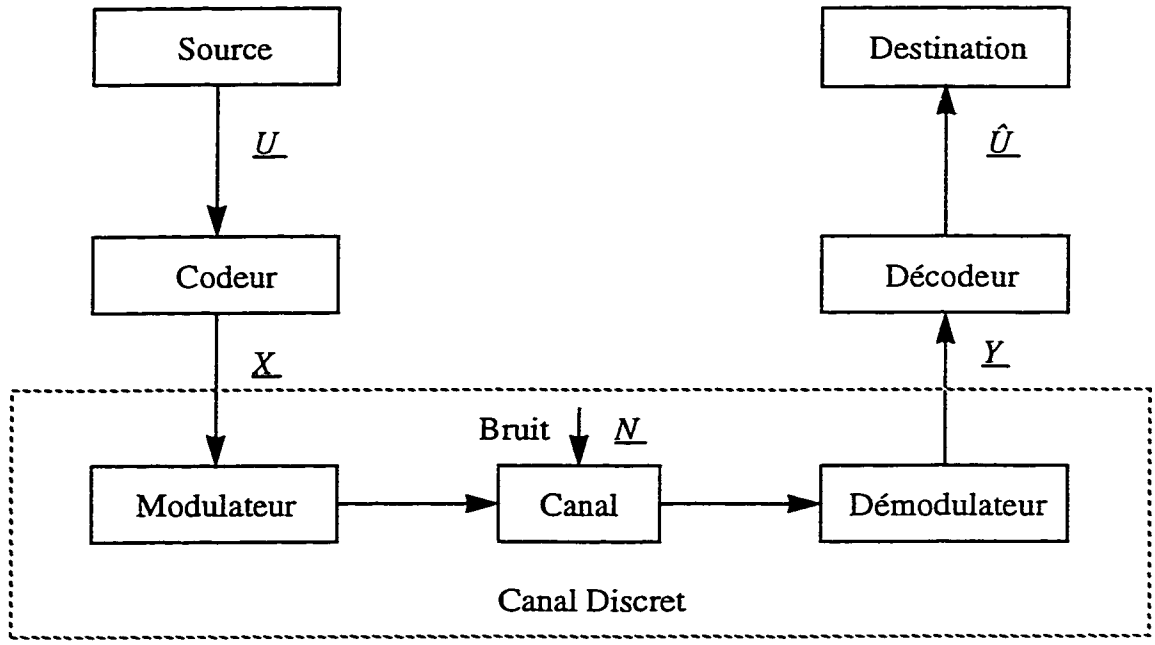


Figure 2.1: Système de communication numérique

2.2 CANAL DISCRET SANS MÉMOIRE

Dans sa forme le plus simple, un canal DMC est représenté par le canal binaire symétrique (en anglais Binary Symetric Channel ou BSC) illustré à la figure 2.2. Pour ce genre de canal, les symboles d'entrée et de sortie sont binaires, les symboles d'entrée sont souvent équiprobables.

On suppose habituellement que le bruit qui affecte le canal de transmission est un bruit blanc gaussien additif, de densité spectrale unilatérale N_0 W/Hz. Le rapport signal à bruit est E_b/N_0, où E_b représente l'énergie par bit d'information reçue au décodeur. Ce rapport est utilisé afin d'évaluer les performances des systèmes codés.

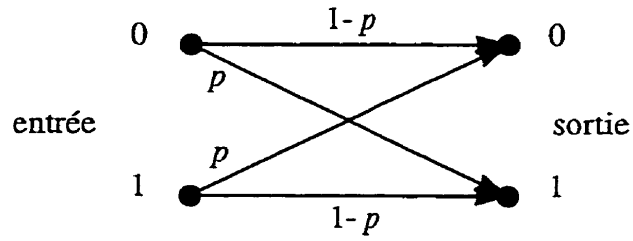


Figure 2.2: Canal discret sans mémoire, binaire et symétrique

La probabilité que le récepteur reçoive un symbole "1" ("0") sachant qu'un symbole "0" ("1") a été transmis est égale à la probabilité de transition dans le canal p, et elle dépend du bruit, du type de modulation utilisée dans le canal et du rapport signal à bruit. Par exemple, pour une modulation PSK, la probabilité de transition du canal de la figure 2.2 est donnée par:

$$p = Q\left(\sqrt{2\frac{E_s}{N_0}}\right) \tag{2-1}$$

où E_s est l'énergie par signal et $Q(x)$ est de la forme:

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

2.3 CODAGE CONVOLUTIONNEL

Un codeur convolutionnel de taux de codage $R = 1/\nu$ et de longueur de contrainte K peut être représenté par une machine linéaire à nombre d'états finis composée d'un

registre à décalage de K cellules, de v additionneurs modulo-2 connectés à certaines cellules du registre à décalage et d'un commutateur qui balaie les v additionneurs. L'ensemble des connections entre le registre à décalage et les additionneurs spécifie le code [4]. Ce codeur est représenté à la figure 2.3.

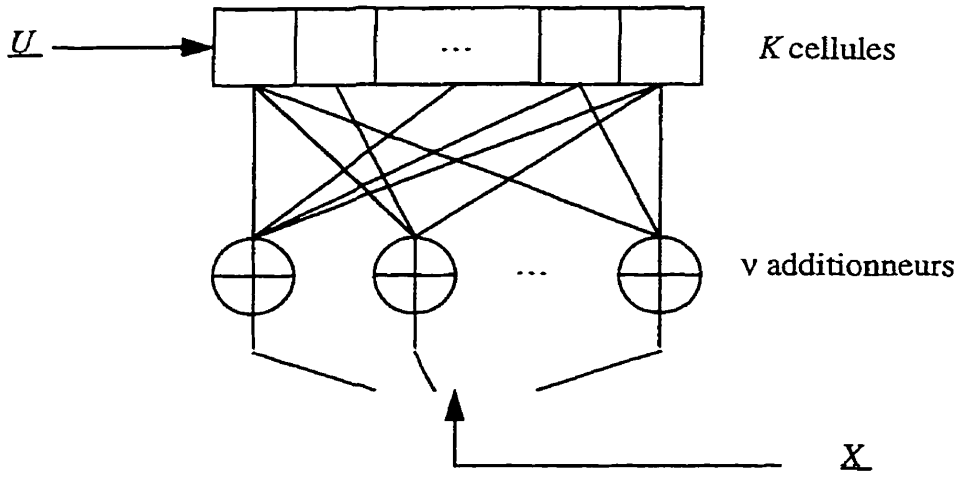


Figure 2.3: Codeur de taux de codage $R = 1/v$

La séquence d'information est

$$\underline{U} = (u_1, u_2, \ldots, u_L) \tag{2-3}$$

où $u_i \in \{0,1\}$.

Supposons que les K cellules du registre à décalage soient initialisées à zéro. Le premier bit d'information u_1 est alors introduit dans la première cellule du registre. Les v additionneurs modulo-2 sont ensuite balayés séquentiellement par le commutateur afin de fournir v symboles codés: $x_1^{(1)}, x_1^{(2)}, \ldots, x_1^{(v)}$. Le contenu des cellules est ensuite déplacé d'un bit vers la droite, afin que le second bit d'information u_2 puisse être introduit dans la

première cellule, fournissant ainsi v nouveaux symboles codés: $x_2^{(1)}$, $x_2^{(2)}$,..., $x_2^{(v)}$. Ce processus se poursuit jusqu'à ce que le dernier bit d'information u_L entre dans le registre à décalage. On fait suivre la séquence d'information de (K-1) zéros pour réinitialiser à zéro le registre à décalage. Cette séquence de zéros s'appelle la queue du message. Or à l'instant t, on a la séquence codée suivante:

$$x_t^{(i)} = \sum_{j=0}^{K-1} u_{t-j} \cdot g_{ij} \tag{2-4}$$

où $g_{ij} \in \{0,1\}$, $i = 1, 2, \ldots, v$, représentent les connections entre les v additionneurs modulo-2 et les cellules du registre à décalage.

Selon la définition de Forney [5], les polynômes générateurs binaires G_i, $i = 1, 2, \ldots, v$, sont donnés par:

$$G_i(D) = \sum_{j=0}^{K-1} g_{ij} \cdot D^j \tag{2-5}$$

En associant les séquences en D qui est une variable temporelle appelée unité de retard, la relation (2-4) devient:

$$X_i = G_i(D) \cdot U \tag{2-6}$$

où $i = 1, 2, \ldots, v$.

Le vecteur

$$G_i = (g_{i0}, g_{i1}, \cdots, g_{i(K-1)}), \; i = 1, 2, \ldots, \upsilon \tag{2-7}$$

spécifie les connexions (1: connexion, 0: pas de connexion) du $i^{ème}$ additionneur aux K cellules du registre à décalage.

De cette façon, on spécifie un code de taux 1/v par v vecteurs de connexions (un pour chaque additionneur) chacun comportant K éléments.

Par exemple, les vecteurs:

$$G_1 = 101$$
$$G_2 = 111 \tag{2-8}$$

spécifient le codeur de la figure 2.4: v = 2, K = 3, R = 1/2.

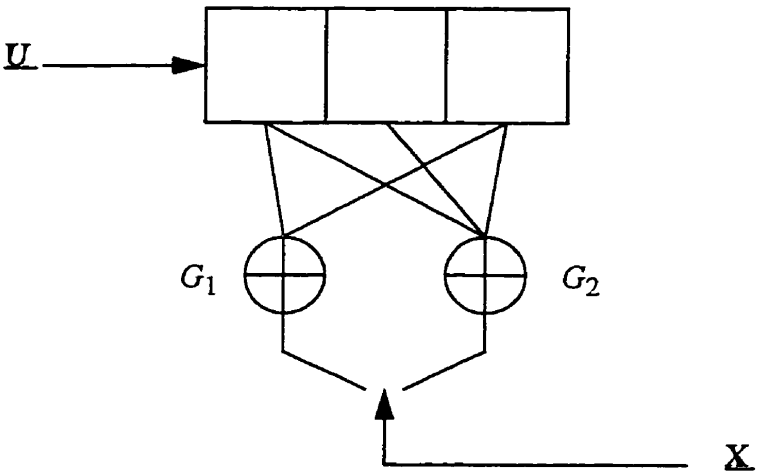


Figure 2.4: Codeur de taux de codage R = 1/2, K = 3, v = 2, G = (5, 7)

Dans cet exemple, on a:

$$G_1(D) = 1 \oplus D^2$$
$$G_2(D) = 1 \oplus D \oplus D^2 \tag{2-9}$$

2.3.1 Diagramme d'état

Le diagramme d'état est un graphe décrivant les transitions d'états du codeur. Il est constitué de 2^{K-1} sommets correspondants aux états du codeur et d'arcs orienté représentants les transitions entre les états.

Le diagramme d'état est une représentation efficace des transitions d'un code en particulier et il est très utilisé pour les codes dont le nombre d'état est assez petit. Le diagramme d'état du codeur de la figure 2.4 est représenté à la figure 2.5.

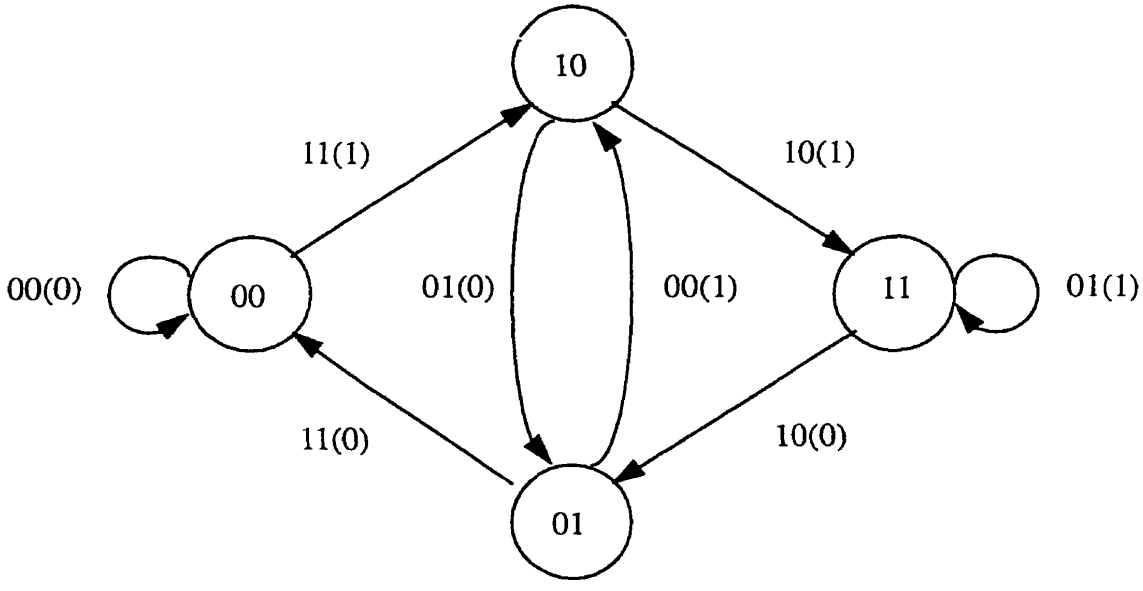


Figure 2.5: Diagramme d'état du codeur de la figure 2.4

Les états sont représentés par des cercles. Les transitions du codeur d'un état à un autre sont représentées par des branches reliant les différents états. Les symboles codés ainsi que le bit d'information (entre parenthèses) ayant causé la transition sont indiqués

sur chacune des branches.

Le diagramme d'état est un outil pratique pour la caractérisation des codes. Mais il ne permet cependant pas de suivre aisément les évolutions dynamiques du codeur dans le temps. Pour pallier à ces inconvénients, deux représentations schématiques plus complètes du codeur sont alors utilisées: la représentation en arbre et la représentation en treillis. Ces deux représentations fournissent une information temporelle qui n'est pas aussi évidente avec le diagramme d'état.

2.3.2 Représentation en arbre

Dans cette représentation, les transitions entre les états de code sont illustrées comme un cheminement à partir d'un état de départ (généralement l'état zéro) qui est l'origine de l'arbre, vers les états ultérieurs, qui sont reliés par les branches de l'arbre.

Chaque noeud de l'arbre illustre un état du codeur, la branche supérieure correspond à un bit d'information "0" et la branche inférieure à un bit d'information "1". Les symboles codés à la sortie de l'encodeur sont indiqués sur chacune des branches. Pour le codeur de la figure 2.4, l'arbre qui y correspond est illustré à la figure 2.6.

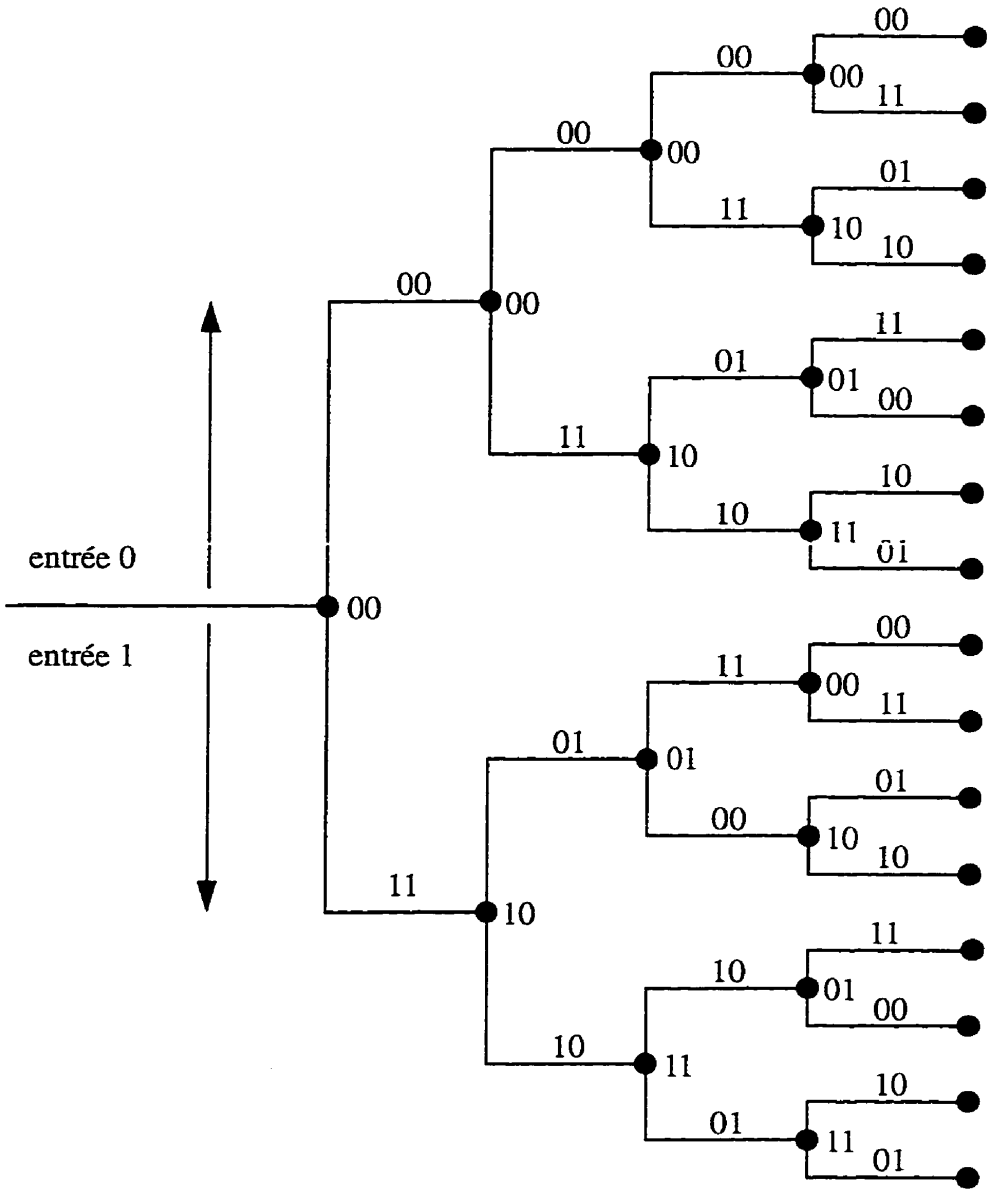


Figure 2.6: Représentation en arbre du codeur de la figure 2.4

2.3.3 Représentation en treillis

Dans un arbre, le nombre de noeuds croît exponentiellement avec le nombre de niveau représentés. Un moyen plus pratique qui permet d'illustrer l'action du codeur, est la représentation en treillis de la figure 2.7 qui correspond au codeur de la figure 2.4.

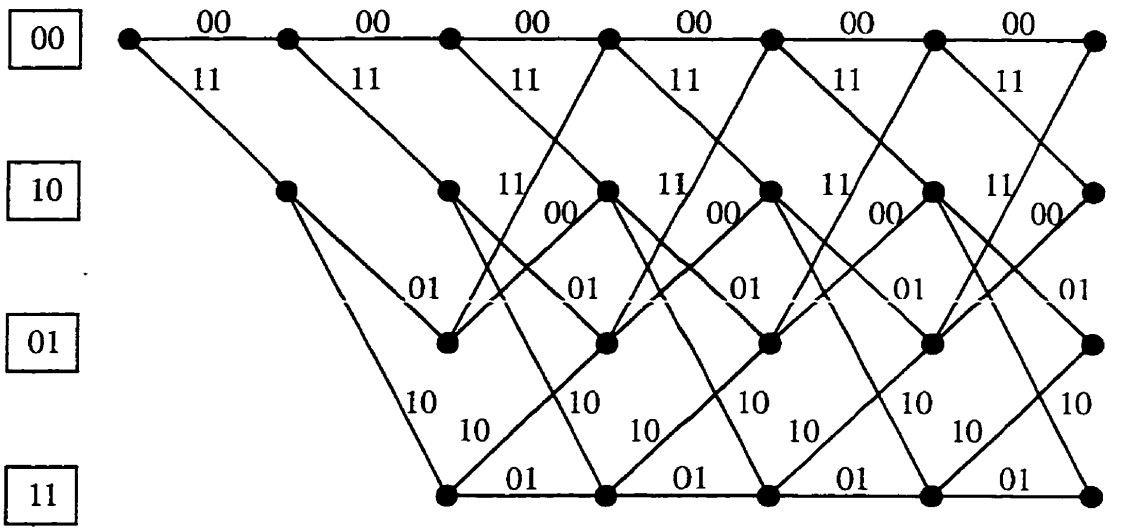


Figure 2.7: Représentation en treillis du codeur de la figure 2.4

Dans le treillis, le nombre de noeuds de chaque niveau est le même que celui de la représentation d'état. Cela permet d'éliminer la redondance de l'arbre. Les états sont représentés sur un axe vertical. Les symboles codés à la sortie de l'encodeur sont indiqués à côté de chacune des branches.

2.4 PROPRIÉTÉS DE DISTANCE

Les performances d'un code convolutionnel dépendent de ses propriétés de distance [6]. La puissance de correction de ces codes dépend de la longueur de contrainte K, du taux de codage R, ainsi que de certains paramètres de distance tel que la distance libre et le profil de distance.

Soient $\underline{X}$ et $\underline{Y}$ deux mots de code. La distance de Hamming d_H entre les deux mots de code est égale au nombre de positions qui différent entre ces deux mots de code. Elle est définie comme suit:

$$d_H(\underline{X}, \underline{Y}) = W_H(\underline{X} \oplus \underline{Y}) \tag{2-10}$$

où W_H est le poids de Hamming. La distance de Hamming est égale au nombre de "1" dans l'opération $\underline{X} \oplus \underline{Y}$ qui est la somme modulo-2 des séquences $\underline{X}$ et $\underline{Y}$.

La fonction de distance des colonnes d'ordre n, notée $d_c(n)$, est la distance de Hamming minimale entre toutes les paires de mots de code de longueur n branches qui diffèrent dans leur première branche.

$$d_c(n) = min\left(d_H(\underline{X}_n, \underline{Y}_n)\right) \tag{2-11}$$

Le profil de distance $\underline{d}$ est constitué de l'ensemble des $d_c(n)$ tel que $n = 1, 2, \ldots, K$. En général, il est préférable que le profil de distance d'un code croisse le plus rapidement possible.

$$\underline{d} = (d_c(1), d_c(2), \ldots, d_c(K)) \tag{2-12}$$

La distance libre dénoté d_{free} est la distance de Hamming minimale entre deux chemins quelconques ayant divergé sur une longueur arbitrairement grande avant de reconverger. La distance minimale dénoté d_{min} est la plus petite distance entre le mot de code $\underline{0}$ et tous autres mots de code non nuls sur une longueur de contrainte égale à K.

$$d_{free} = \lim_{n \to \infty} d_c(n) \tag{2-13}$$

Par définition,

$$d_{free} \geq d_{min} \tag{2-14}$$

La distance libre d_{free} est une notion importante pour le décodage de Viterbi et le décodage séquentiel. En général, les codes les plus puissants sont ceux dont la distance libre est maximale.

2.5 CODAGE CONVOLUTIONNEL RÉCURSIF SYSTÉMATIQUE

A partir d'un code convolutionnel non systématique, on peut construire un code convolutionnel récursif systématique avec une boucle de retour. Un codeur convolutionnel récursif systématique est illustré à la figure 2.8.

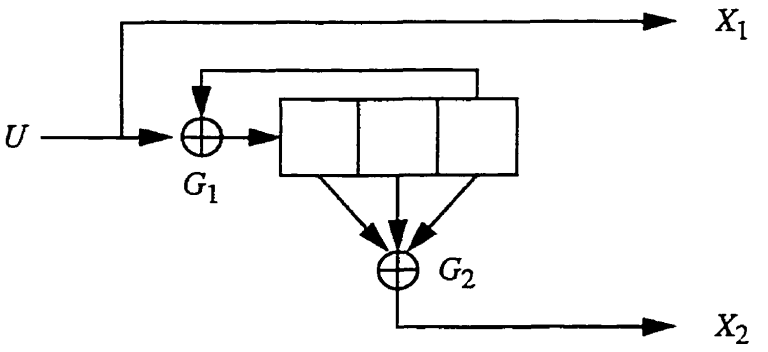


Figure 2.8: Codeur convolutionnel récursif systématique

$R = 1/2, K = 3, \nu = 2, G = [1, (1+D+D^2)/(1+D^2)]$

Le code convolutionnel récursif systématique possède la même distance libre et le même nombre de chemins que le code convolutionnel non systématique, mais leur nombre total de bits d'information sont différents.

Après avoir comparé les performances d'erreur de codes, on constate que pour un faible rapport signal à bruit, le code convolutionnel récursif systématique a un léger avantage par rapport au code convolutionnel non systématique à partir duquel il a été construit. On analysera plus profondément cet aspect au chapitre 4.

2.6 CODAGE CONVOLUTIONNEL RÉCURSIF SYSTÉMATIQUE PERFORÉ

Les codes convolutionnels perforés sont une classe de codes convolutionnels de taux de codage élevé qui sont obtenus à partir de codes convolutionnels de faible taux de codage par l'élimination périodique (perforation) de symboles codés de la sortie d'un

codeur de faible taux de codage. Un codeur convolutionnel récursif systématique perforé est présenté à la figure 2.9.

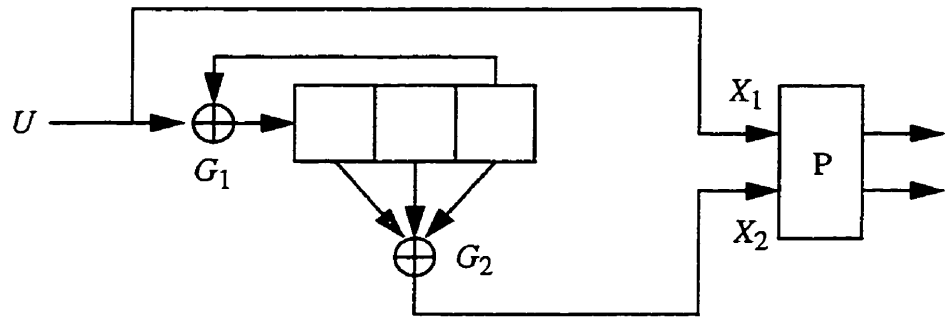


Figure 2.9: Codeur convolutionnel récursif systématique perforé

$R = 1/2, K = 3, \nu = 2, G = [1, (1+D+D^2)/(1+D^2)]$

Au chapitre 5, nous étudions la perforation des codes convolutionnels récursifs systématiques, issus des codes de taux de codage $R = 1/2$. Nous examinons alors tous les codes de longueurs de contrainte K allant de 3 à 9 avec des patrons de perforation différents qui gardent la caractéristique 'systématique'. Les taux de codage des codes perforés deviennent égales à 2/3, 3/4, 4/5, 5/6, 6/7 et 7/8.

Aussi, à partir d'un même code, nous comparons la performance des codes de différents patrons de perforation. La probabilité d'erreur d'un code perforé a une borne inférieure par égale à la probabilité d'erreur de son code origine. Puis le taux de codage est faible, plus les performances d'erreur sont meilleures. On analysera plus profondément cet aspect au chapitre 5.

2.7 CODAGE À CONCATÉNATION PARALLÈLE DE CODES CONVOLUTIONNELS

Le codage concaténé parallèle de codes convolutionnels (CPCC) consiste en une concaténation parallèle de deux codeurs récursifs systématiques et un entrelaceur tel qu'illustré à la figure 2.10. Le taux de codage résultant est de $1/(2v-1)$ pour des codeurs de taux $1/v$. La présence d'un entrelaceur de dimension N fait en sorte que les mêmes bits d'information sont appliqués aux deux codeurs C_1 et C_2, mais dans un ordre différent. L'entrelaceur joue effectivement un rôle de "décorrélation". Ce type de codage permet d'obtenir de très bonnes performances même pour des faibles valeurs du rapport signal à bruit.

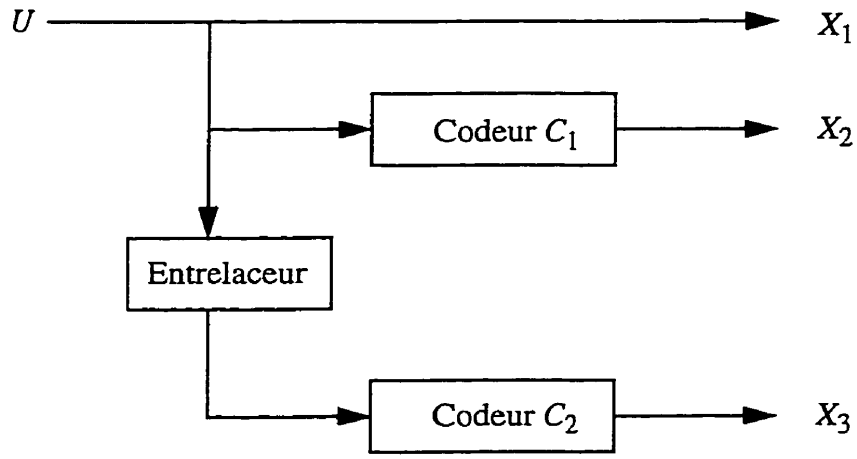


Figure 2.10: Codeur concaténé parallèle de codes convolutionnels

En se référant à l'article de Benedetto et Montorsi [7] et de Podemski, Holubowicz, Berrou, Glavieux [8], nous avons simulé par ordinateur le système afin de calculer les spectres et les performances. Deplus, nous traitons les codes CPCC perforés.

Au chapitre 6, nous définissons de façon plus précise les concepts de base liés aux codes CPCC, tel que spectre, hyper treillis, performance, ainsi que ceux des codes CPCC perforés.

2.8 DÉCODAGE À MAXIMUM DE VRAISEMBLANCE

La séquence de symboles reçue par le décodeur diffère généralement de la séquence transmise à cause du bruit qui perturbe le canal. Le décodage des codes convolutionnels consiste à retrouver la séquence d'information transmise à partir de la séquence reçue. Cela peut être vu comme l'opération inverse du processus de codage.

Le décodage probabiliste des codes convolutionnels est un moyen très efficace pour détecter et corriger certaines erreurs introduites dans le canal par le bruit.

Le décodage à maximum de vraisemblance est un décodage probabiliste. Il consiste à déterminer le message le plus vraisemblable à partir de la séquence reçue.

Soient m le message transmis, $\underline{X}^m = (\underline{X}_1, \underline{X}_2, \ldots, \underline{X}_L) = (x_1^{(1)}, x_1^{(2)}, \ldots, x_1^{(v)}, x_2^{(1)}, \ldots, x_L^{(v)})$ la séquence codée et $\underline{Y} = (\underline{Y}_1, \underline{Y}_2, \ldots, \underline{Y}_L) = (y_1^{(1)}, y_1^{(2)}, \ldots, y_1^{(v)}, y_2^{(1)}, \ldots, y_L^{(v)})$ la séquence reçue. Pour minimiser la probabilité d'erreur, il faut choisir le message $\hat{m}$ tel que [9]:

$$P(\underline{Y}|\hat{m}) \geq P(\underline{Y}|m) \qquad \forall (\hat{m} \neq m) \tag{2-15}$$

En utilisant la règle de Bayes sur les probabilités conditionnelles dans l'équation (2-15), on peut écrire:

$$P(\hat{m})\,P(\underline{Y}|\underline{X}^{\hat{m}}) \geq P(m)\,P(\underline{Y}|\underline{X}^{m}) \qquad \forall(\hat{m} \neq m) \tag{2-16}$$

où $P(\hat{m})$ représente la probabilité à priori du message $\hat{m}$. Si les messages sont équiprobables, l'équation (2-16) devient:

$$P(\underline{Y}|\underline{X}^{\hat{m}}) \geq P(\underline{Y}|\underline{X}^{m}) \qquad \forall(\hat{m} \neq m) \tag{2-17}$$

Minimiser la probabilité d'erreur revient donc à maximiser cette fonction de vraisemblance. Pour un canal discret sans mémoire (DMC) de probabilités de transitions $P(y_i^{(j)}|x_i^{(j)})$, on peut écrire pour la branche i:

$$P(\underline{Y}_i|\underline{X}_i^{\hat{m}}) = \prod_{j=1}^{\nu} P(y_i^{(j)}|x_i^{(j)}) \tag{2-18}$$

Afin d'utiliser une mesure additive et donc plus pratique, on utilise le logarithme de $P(\underline{Y}_i|\underline{X}_i^{\hat{m}})$. Le résultat représente alors la mesure de vraisemblance pour la branche i,

$$\gamma_i = \sum_{j=1}^{\nu} \log\left(P(y_i^{(j)}|x_i^{(j)})\right) \tag{2-19}$$

Pour une séquence de L bits d'information, la fonction de vraisemblance est:

$$\Gamma = \sum_{i=1}^{L} \gamma_i \qquad (2\text{-}20)$$

où Γ, γ_i et $\log\left(P\left(y_i^{(j)} \middle| x_i^{(j)}\right)\right)$ sont appelés respectivement: métrique cumulative totale, métrique de branche et métrique de symbole. Cette dernière est habituellement arrondie à un entier.

Le processus de décodage à maximum de vraisemblance consiste donc à choisir le mot de code dont la métrique cumulative est maximale. Pour le canal binaire symétrique, le mot de code ayant la métrique cumulative maximale est le chemin dont la distance de Hamming est minimale [3]. Mais la difficulté du décodage provient plutôt du fait que le nombre de chemins à considérer est très grand pour permettre une recherche exhaustive. Pour remédier à cela, il existe deux techniques de décodage qui sont très répandues et qui permettent d'accomplir cette tâche. Ces deux techniques sont le décodage de Viterbi et le décodage séquentiel. Peu importe la technique utilisée, le rôle du décodeur reste de déterminer le chemin qui, parmi tous les chemins de l'arbre ou du treillis possède la distance de Hamming la plus faible. de la séquence reçue

L'algorithme de Viterbi fut proposé en 1967 par Viterbi [3]. Cet algorithme est basé sur deux propriétés qui facilitent la recherche du chemin dont la distance de Hamming est minimale. La première propriété est une recherche exhaustive de chemins dans le treillis. La deuxième propriété permet d'éliminer tous les chemins reconvergeant sur un même état sauf un, sur la base d'une comparaison de distances de Hamming. Cet algorithme est une technique de décodage extrêmement puissante pour les codes convolutionnels. L'avantage de cette technique est que l'effort de calcul afin de décoder un certain nombre de branches reçues demeure constant, ce qui n'est pas le cas de toutes les

autres techniques de décodage. Cependant, la taille du treillis croît exponentiellement avec la longueur de contrainte du code. En pratique, cette augmentation exponentielle de la complexité limite son utilisation à des codes dont la longueur de contrainte est inférieure à 10 ($K < 10$).

L'algorithme de décodage de Viterbi est basé sur la structure en treillis du code tandis que celui du décodage séquentiel est basé sur la structure en arbre. Dans ce dernier la recherche ne se fait pas sur l'arbre du code en entier, mais seulement en niveau du chemin qui localement apparait comme le plus vraisemblable.

Pour le décodage séquentiel, il existe deux algorithmes: l'algorithme de Fano [10] et celui de Zigangirov-Jelinek [11] [12].

Le premier, l'algorithme de Fano, utilise un système de seuils. Tant que la métrique du chemin est supérieure au seuil courant, il va de l'avant, en explorant ses extensions. Si la métrique n'est pas suffisante, il retourne en arrière et cherche un meilleur chemin avec un réajustement du seuil.

L'algorithme de Zigangirov-Jelinek, contrairement à celui de Fano, utilise un moyen de stockage dans une pile qui lui permet de garder en mémoire la métrique. En d'autres termes on désire que le chemin le plus vraisemblable se retrouve au sommet de la pile. Ainsi chaque chemin qui se retrouve au sommet est prolongé en ses extensions. Après avoir retiré ce chemin du sommet de la pile, les métriques des nouveaux chemins ainsi obtenus sont comparé aux autres afin de réordonner la pile. Cette opération se poursuit jusqu'à ce qu'on ait atteint la profondeur finale de l'arbre.

Quelque soit l'algorithme utilisé, il existe une variabilité de l'effort de calcul due aux retours en arrière. Pour tenir compte de cette variabilité, on utilise des tampons d'entrée et de sortie, qui permettent de régler les débits.

Le décodeur séquentiel présente d'autres limitations qui sont plus difficiles à contourner. La principale limitation vient du fait que l'effort de décodage est variable. L'effort de calcul varie de façon aléatoire et il faut utiliser dans ce cas des tampons à l'entrée et à la sortie du décodeur afin d'isoler le décodeur et son fonctionnement asynchrone des éléments synchrones du système de communication. Pour améliorer cet algorithme, on cherche donc autant que possible, d'une part, à régulariser l'effort de calcul pour limiter les débordements du tampon d'entrée, et d'autre part, à limiter la croissance de la taille de la pile pour éviter les débordement de pile.

L'algorithme de Viterbi a été et reste toujours utilisé pour le décodage des codes convolutionnels. Mais il ne garantit pas la minimisation de la probabilité d'erreur par bit (ou symbole). Or, pour le décodage des codes CPCC, l'algorithme de Viterbi n'est pas idéal. Actuellement, une autre technique est proposée pour le décodage des codes CPCC, soit l'algorithme MAP (maximum à posteriori) dû à Bahl *et al.*

En 1966, Chang et Hancock développèrent un algorithme qui minimise la probabilité d'erreur par symbole, ou encore maximise-la probabilité à posteriori. En 1972, Bahl *et al.* [13] et McAdam *et al.* [14] ont simultanément adapté l'algorithme MAP pour les codes correcteurs d'erreurs. Le principe de l'algorithme MAP est basé sur deux boucles récursives, dont l'une est croissante et l'autre est décroissante. À cause de la boucle décroissante, le décodage d'un bloc ne peut commencer que lorsque la réception d'un bloc est complétée. Les performances de l'algorithme MAP sont supérieures à celles

de l'algorithme de Viterbi de quelque 0.3dB. Cependant, l'algorithme MAP est beaucoup plus complexe, et c'est pour cette raison qu'il a été longtemps ignoré. En effet la différence de performance entre deux algorithmes n'est plus un facteur déterminant si l'on ne tient pas compte de la complexité. Malgré les performances présentées dans [15], un inconvénient majeur subsiste: la complexité du décodage.

Une simplification de l'algorithme MAP fut proposée par Robertson [16], et reprise par Berrou *et al.* dans [17] afin de l'adapter à un processus de décodage itératif. La recherche d'un processus de décodage pour les codes CPCC réalisée par Naoufel Bouzouita [18] a été récemment introduite sous la direction de D. Haccoun à la section de Télécommunication (Génie électrique) de l'École Polytechnique. La méthode que Bouzouita a utilisé n'apporte que de légères modifications à l'algorithme proposé dans [17].

CHAPITRE 3

SPECTRE DES CODES CONVOLUTIONNELS

La performance des codes convolutionnels est évaluée à partir de leur spectre. Dans ce chapitre, la notion de spectre du code est définie. Ensuite, nous passons à l'évaluation des performances à partir des relations entre le spectre et la probabilité d'erreur. Finalement, deux approches essentielles de détermination du spectre sont analysées.

3.1 DÉFINITION DU SPECTRE D'UN CODE

Le spectre est une conception générale qui s'applique à toutes les classes de codes linéaires. Le spectre d'un code convolutionnel représente l'ensemble des mots de code non nuls dénombrés en fonction de leur distance de Hamming par rapport au mot de code nul. Il est habituellement présenté sous la forme de tableau de trois colonnes:

d: le poids des mots de code,

A_d: le nombre de mots de code de poids d,

C_d: le nombre total de bits d'information "1" correspondant à ces mots de code de poids d.

Chaque triplet $\{d, A_d, C_d\}$ compose une raie du spectre. Seules les raies de poids

supérieur ou égal à d_{free} influencent le calcul des probabilité d'erreur du code. Cela fait appel à la notion de raie solide du spectre. La première raie solide est la première raie non-nulle du spectre. Le tableau 3.1 illustre le spectre du code convolutionnel de taux de codage $R = 1/2$ du chapitre 2 (Figure 2.4).

Tableau 3.1: Spectre du code de la figure 2.4

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

Pour la deuxième ligne du tableau 3.1, il est indiqué que le nombre de mot de code ayant le poids 6 est 2 et le nombre total de bits d'information "1" est 4.

Les mots de code convolutionnel sont habituellement très longs, d'une longueur que l'on considère infinie. Le spectre contient alors un nombre illimité de raies. Or pour des raisons pratiques, on se doit de limiter le nombre de raies. Le spectre de L raies se définit comme le dénombrement de tous les chemins de treillis ou de l'arbre de poids inférieur ou égal à L qui débutent par un bit d'information "1" et qui se terminent sur un noeud d'état 0. Il ne doit y avoir aucun autre noeud d'état 0 sur ces chemins. Les figures

3.1 et 3.2 décrivent respectivement sur l'arbre et le treillis les séquences qui composent les deux raies solides de plus faible poids du code du tableau 3.1.

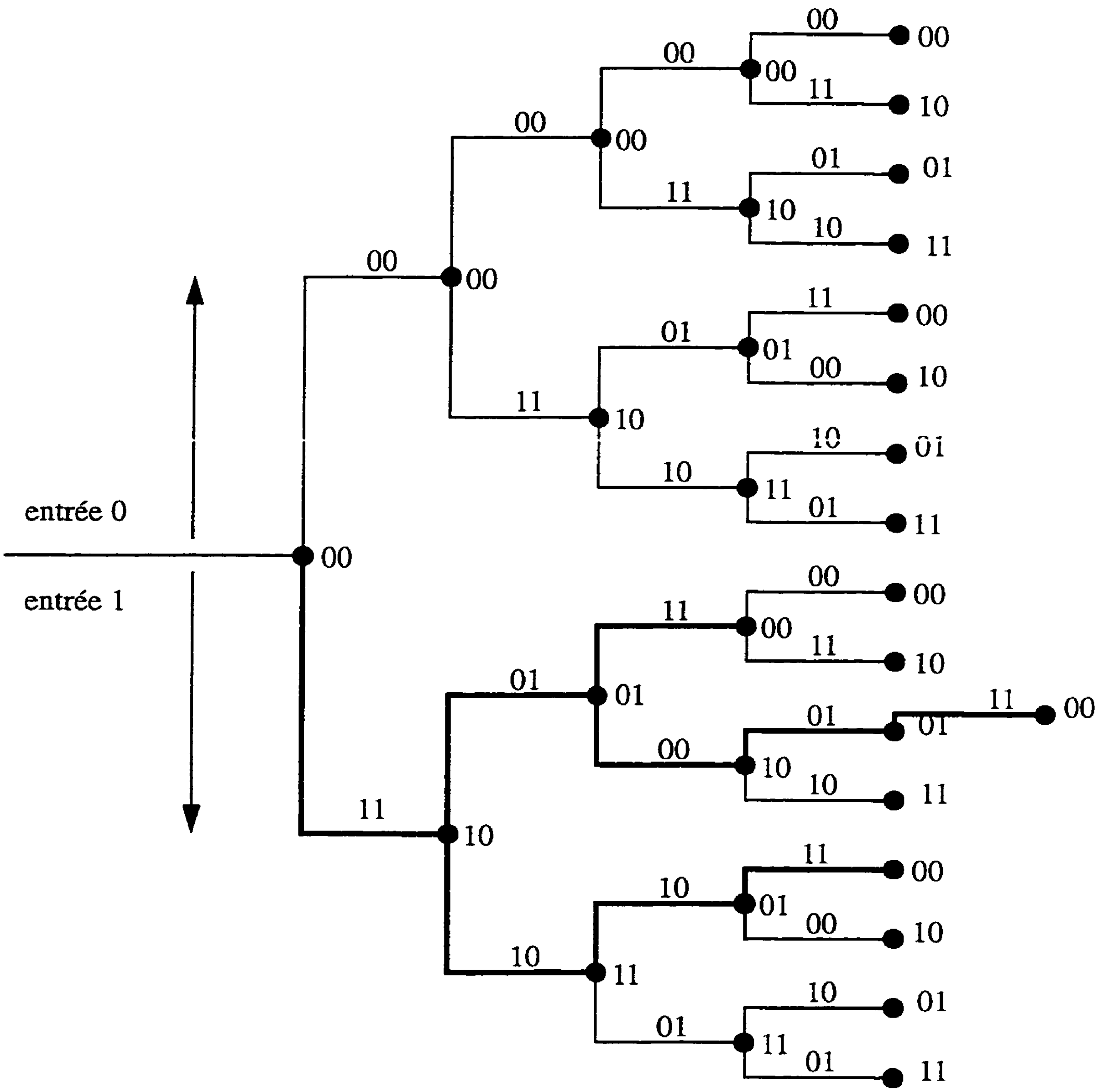


Figure 3.1: Représentation en arbre

Chemins associés aux deux premières raies du spectre du code de la figure 2.4

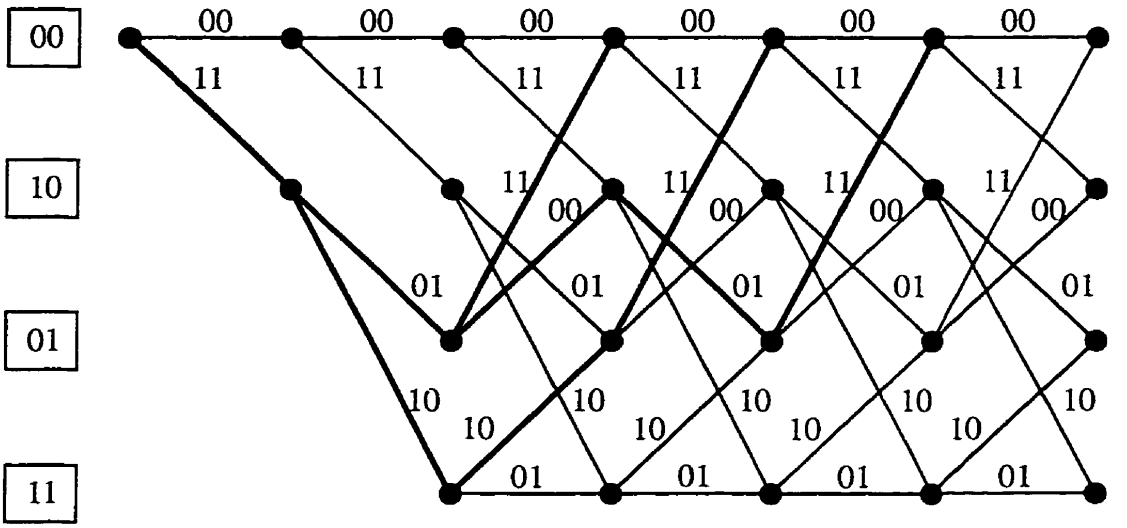


Figure 3.2: Représentation en treillis

Chemins associés aux deux premières raies du spectre du code de la figure 2.4

Le tableau 3.2 donne quelques raies solides du spectre de différents codes convolutionnels.

A ce stade, il est utile de faire deux observations:

(1) le poids de la première raie solide est égale à d_{free};

(2) la valeur des raies (A_d et C_d) augmente avec d. Cette augmentation est de nature exponentielle de sorte que le nombre de raies atteint assez vite des valeurs astronomiques.

Tableau 3.2: Spectre de quelques codes convolutionnels de taux $R = 1/v$

d	A_d	C_d
6	1	2
7	3	7
8	5	18
9	11	49
10	25	130
11	55	333

(a) $R = 1/2, K = 4, G_1 = 15, G_2 = 17$

d	A_d	C_d
12	2	14
13	8	26
14	15	74
15	35	257
16	68	496
17	170	1378

(b) $R = 1/2, K = 10, G_1 = 1167, G_2 = 1545$

d	A_d	C_d
15	3	11
16	5	16
17	5	19
18	6	28
19	11	55
20	15	96

(c) $R = 1/5, K = 7, G_1 = 175, G_2 = 131, G_3 = 135, G_4 = 135, G_5 = 147$

3.2 ÉVALUATION DE LA PERFORMANCE D'UN CODE CONVOLUTIONNEL

La performance d'un code convolutionnel est déterminée en fonction d'une probabilité d'erreur. Cette probabilité d'erreur est évaluée à partir du spectre du code. Cette section montre la relation qui existe entre le spectre et la probabilité d'erreur. Trois mesures différentes de la probabilité d'erreur sont présentées, soient la probabilité d'erreur entre deux mots de code, la probabilité d'erreur d'une séquence de symboles codés et enfin la probabilité d'erreur par bits d'information. Pour tous ces cas, on suppose l'utilisation d'un décodeur à maximum de vraisemblance.

3.2.1 Probabilité d'erreur entre deux mots de code

Soient $\underline{X}_i$ et $\underline{X}_j$ deux mots de code ayant une distance de Hamming $d_H(\underline{X}_i,\underline{X}_j) = d$ (c'est-à-dire qui diffèrent entre eux de d positions), et soit P_d la probabilité d'erreur entre ces deux mots de code. Posons $\underline{X}_i$ comme étant la séquence de symboles codés effectivement transmise et $\underline{Y}$ la séquence reçue. Le décodeur optimal commettra une erreur en estimant $\underline{X}_j$ par $\underline{X}_i$ si plus de la moitié des d symboles qui distinguent $\underline{X}_i$ et $\underline{X}_j$ sont modifiés lors de la transmission. Lorsque la moitié des d symboles ont été altérés, le décodeur choisit au hasard entre $\underline{X}_i$ et $\underline{X}_j$.

Pour un canal à bruit blanc gaussien additif dont la densité de bruit bilatérale est $No/2$, et une modulation cohérente PSK, la probabilité d'erreur P_d entre deux mots de code de distance entre eux égale à d est donnée par [6]:

$$P_d = Q\left(\sqrt{2 \cdot d \cdot R \cdot \frac{E_b}{N_0}}\right) \tag{3-1}$$

où E_b est l'énergie transmise par bit et $Q(x)$ est donnée par:

$$Q(x) = \int_x^{\infty} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt \tag{3-2}$$

Pour un canal binaire symétrique, la probabilité d'erreur P_d entre deux mots de code de distance d se calcule de la façon suivante [6]:

- si d est impair:

$$P_d = \sum_{i=\frac{d+1}{2}}^{d} \binom{d}{i} p^i (1-p)^{d-i} \tag{3-3}$$

- si d est pair:

$$P_d = \sum_{i=\frac{d}{2}+1}^{d} \binom{d}{i} p^i (1-p)^{d-i} + \frac{1}{2}\binom{d}{d/2} p^{d/2} (1-p)^{d/2} \tag{3-4}$$

où p est la probabilité de transition dans le canal.

Même si un code compte généralement plus de deux mots de code, il est raisonnable de penser que la probabilité d'erreur globale puisse se calculer à partir de la distribution de distance de Hamming entre toutes les paires de mots de code. Ceci nous conduit à avoir deux mesures de probabilité d'erreur:

- la probabilité d'erreur de séquence P_E: probabilité qu'une séquence soit mal décodée après son passage dans le canal.

$$P_E = P[\hat{\underline{X}} \neq \underline{X}] = P[\hat{\underline{U}} \neq \underline{U}]$$

- la probabilité d'erreur par bit d'information P_b: probabilité qu'un bit soit mal décodé. Du point de vue de l'utilisateur, P_b a beaucoup plus de signification que P_E.

Soit $\underline{U} = (u_1, u_2, \ldots)$, $\hat{\underline{U}} = (\hat{u}_1, \hat{u}_2, \ldots)$,
alors $P_b = P[\hat{u}_i \neq u_i]$.

En pratique, P_b se calcule comme ceci:

$$P_b = \frac{\text{nombre de bits en erreur}}{\text{nombre de bits transmis}}$$

3.2.2 Probabilité d'erreur de séquence

La probabilité d'erreur de séquence, notée P_E, est la probabilité d'erreur que la séquence de symboles codés transmise $\underline{X}$ soit décodée par une séquence erronée quelconque $\hat{\underline{X}}$.

$$P_E = P[\hat{\underline{X}} \neq \underline{X}] \tag{3-5}$$

Pour une séquence d'information particulière $\underline{U}_i$, $P[\varepsilon|\underline{U}_i]$ est la probabilité d'erreur de décodage entre $\underline{X}_i$ et un autre mot de code $\underline{X}_j$. En utilisant la borne union [1], on obtient:

$$P[\varepsilon|\underline{U}_i] \leq \sum_{j \neq i} P_2[\underline{X}_i, \underline{X}_j] \tag{3-6}$$

où $P_2[\underline{X}_i,\underline{X}_j]$ est la probabilité d'erreur entre la paire de mots de code $\underline{X}_i$ et $\underline{X}_j$. Or, $P_2[\underline{X}_i,\underline{X}_j] = P_d$, où $d = d_H(\underline{X}_i,\underline{X}_j)$. Donc,

$$P[\varepsilon|\underline{U}_i] \leq \sum_{d=d_{free}}^{\infty} A_d^i P_d \tag{3-7}$$

où A_d^i est le nombre de mots de code tels que $d = d_H(\underline{X}_i,\underline{X}_j)$, $j \neq i$.

Pour étendre ce résultat à l'ensemble des séquences d'information, il suffit d'exploiter la propriété de linéarité des codes convolutionnels qui entraîne que $A_d^i = A_d^j = A_d$ pour tous d, i, j. Par conséquent, l'équation (3-7) se généralise ainsi:

$$P[\varepsilon|\underline{U}_i] = P[\varepsilon|\underline{U}_j] \leq \sum_{d=d_{free}}^{\infty} A_d P_d \tag{3-8}$$

Donc,

$$P_E = \sum_i P[\varepsilon|\underline{U}_i] P[\underline{U}_i] = \sum_i P[\varepsilon|\underline{U}_j] P[\underline{U}_i] = P[\varepsilon|\underline{U}_j] \sum_i P[\underline{U}_i]$$

$$P_E = P[\varepsilon|\underline{U}_j] = P[\varepsilon|\underline{U}_i] \tag{3-9}$$

En y substituant la relation (3-8), on déduit la borne sur la probabilité d'erreur. Donc, la probabilité d'erreur de séquence P_E est:

$$P_E \leq \sum_{d=d_{free}}^{\infty} A_d P_d \tag{3-10}$$

3.2.3 Probabilité d'erreur par bit d'information

La probabilité d'erreur par bit d'information, P_b, s'obtient simplement en pondérant la probabilité d'erreur de séquence par le nombre de bits en erreur porté par chaque séquence en erreur. On en déduit immédiatement que:

$$P_b \leq \sum_{d=d_{free}}^{\infty} C_d P_d \tag{3-11}$$

où C_d est le nombre total de bit d'information "1" associé à ces mots de code.

Ces résultats dévoilent clairement la relation intime entre la performance d'un code et son spectre. Le problème du calcul de la borne sur la probabilité d'erreur se réduit à celui de la détermination du spectre.

L'application des équation (3-10) et (3-11) exige la connaissance d'un nombre infini de raies. Toutefois, en pratique, seules les quelques premières raies exercent un effet important sur P_E et P_b. Pour s'en convaincre, il suffit d'examiner le tableau 3.3 qui présente l'évolution de P_d selon d pour quelques valeurs de probabilité de transition dans le canal. Lorsque $p < 10^{-2}$, la décroissance de P_d est extrêmement rapide. Ainsi en est-il des produits $A_d P_d$ et $C_d P_d$, même si les valeurs de A_d et de C_d augmentent avec d.

Tableau 3.3: Évaluation de P_d avec d pour quelques valeurs de probabilité de transition p

d	P_d		
	$p = 10^{-2}$	$p = 10^{-4}$	$p = 10^{-6}$
1	1.000×10^{-2}	1.000×10^{-4}	1.000×10^{-6}
2	1.000×10^{-2}	1.000×10^{-4}	1.000×10^{-6}
3	2.980×10^{-4}	3.000×10^{-8}	3.000×10^{-12}
4	2.980×10^{-4}	3.000×10^{-8}	3.000×10^{-12}
5	9.851×10^{-6}	9.999×10^{-12}	1.000×10^{-17}
6	9.851×10^{-6}	9.999×10^{-12}	1.000×10^{-17}
7	3.417×10^{-7}	3.499×10^{-15}	3.500×10^{-23}
8	3.417×10^{-7}	3.499×10^{-15}	3.500×10^{-23}
9	1.219×10^{-8}	1.260×10^{-18}	1.260×10^{-28}
10	1.219×10^{-8}	1.260×10^{-18}	1.260×10^{-28}
11	4.425×10^{-10}	4.618×10^{-22}	4.620×10^{-34}
12	4.425×10^{-10}	4.618×10^{-22}	4.620×10^{-34}
13	1.628×10^{-11}	1.715×10^{-25}	1.716×10^{-39}
14	1.628×10^{-11}	1.715×10^{-25}	1.716×10^{-39}

3.3 DÉTERMINATION DU SPECTRE

Dans cette section, deux méthodes permettant de déterminer le spectre des codes convolutionnels sont présentées. La première est basée sur l'évaluation de la fonction de transfert du code à partir de son diagramme d'état. La seconde consiste en une recherche

exhausive des mots de code dans l'arbre du code.

3.3.1 Fonction de transfert

La fonction de transfert d'un code convolutionnel représente le rapport qui existe entre la sortie et l'entrée du codeur [6]. Elle est évaluée à partir du diagramme d'état du code. Ce dernier est modifié de façon à définir une entrée et une sortie correspondant respectivement à l'état initial E_{0i} et à l'état final E_{0f}. Le noeud de l'état initial est donc scindé en deux noeuds distincts. Tous les autres noeuds du diagramme sont considérés comme des noeuds associés à des états intermédiaires.

La figure 3.3 illustre le diagramme d'état modifié du code de la figure 2.4. Elle est différente de la figure 2.5. Sur les branches du diagramme d'état sont inscrites les informations nécessaires à la compilation du spectre sous la forme D^iB^j, où i représente le poids de Hamming associé à chacune des branches du treillis ou de l'arbre et où j représente le nombre de bits d'information "1" correspondant. La fonction de transfert, notée $T(D, B)$, est égale au rapport entre E_{0f} et E_{0i}.

$$T(D,B) = \frac{E_{0f}}{E_{0i}} \tag{3-12}$$

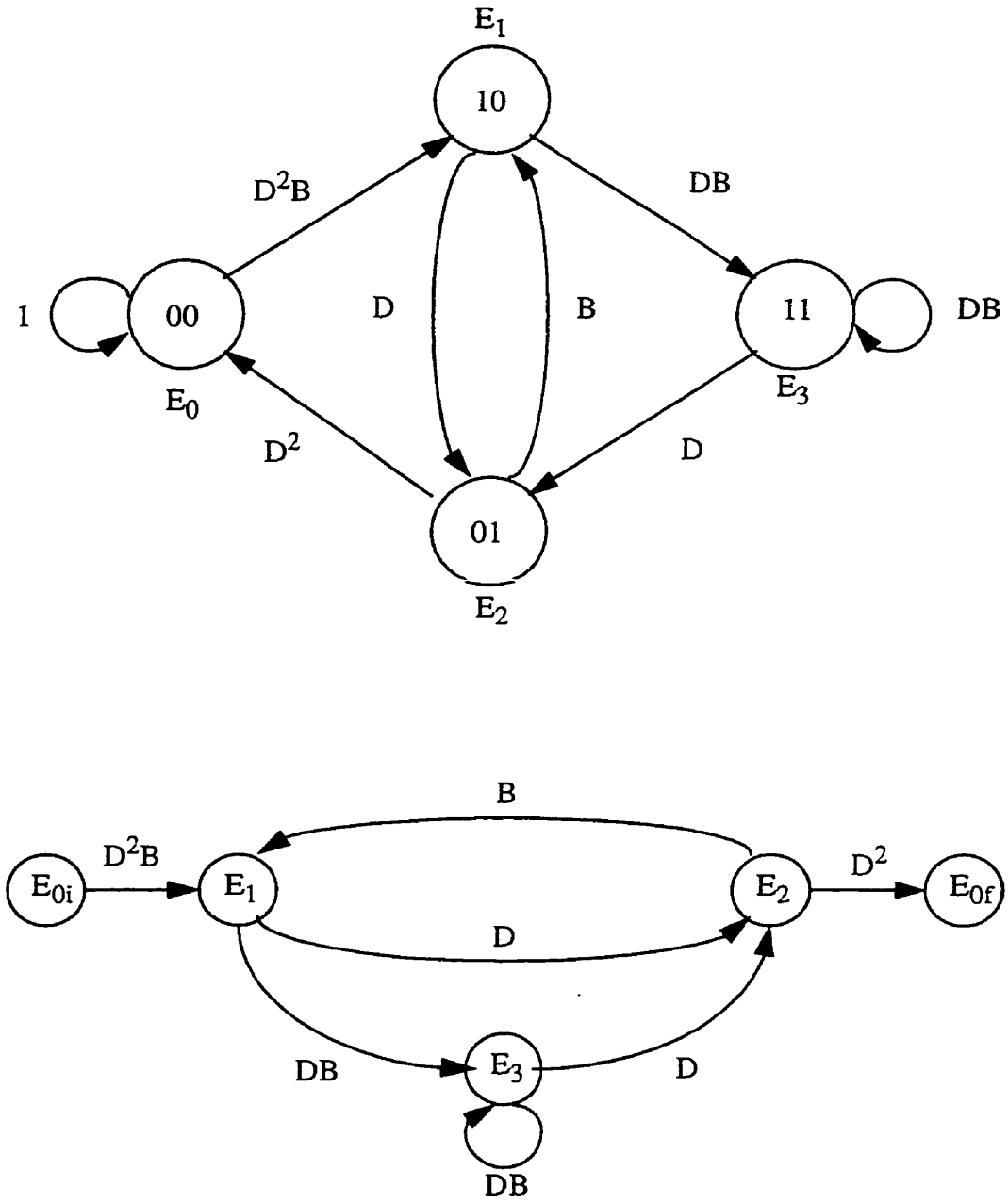


Figure 3.3: Diagramme d'état du code de la figure 2.4

(a) forme initiale, (b) forme modifiée

Sous forme matricielle, le diagramme d'état modifié s'écrit [19]:

$$\begin{bmatrix} E_1 \\ E_2 \\ E_3 \end{bmatrix} = \begin{bmatrix} 0 & B & 0 \\ D & 0 & D \\ DB & 0 & DB \end{bmatrix} \begin{bmatrix} E_1 \\ E_2 \\ E_3 \end{bmatrix} + \begin{bmatrix} D^2 B \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} E_{0i} \end{bmatrix} \tag{3-13}$$

et

$$\begin{bmatrix} E_{0f} \end{bmatrix} = \begin{bmatrix} 0 & D^2 & 0 \end{bmatrix} \begin{bmatrix} E_1 \\ E_2 \\ E_3 \end{bmatrix} \tag{3-14}$$

De façon générale, les équations (3-13) et (3-14) deviennent

$$\boldsymbol{E} = \boldsymbol{A}\boldsymbol{E} + \boldsymbol{F}E_{0i}$$

$$\boldsymbol{E} = [\boldsymbol{I} - \boldsymbol{A}]^{-1} \boldsymbol{F} E_{0i} \tag{3-15}$$

$$E_{0f} = \boldsymbol{G}^T \boldsymbol{E} \tag{3-16}$$

où $\boldsymbol{E}$ est le vecteur colonne contenant tous les états intermédiaires;

$\boldsymbol{A}$, la matrice des transitions entre les états intermédiaires;

$\boldsymbol{F}$, le vecteur colonne des transitions entre l'état E_{0i} et tous les états intermédiaires;

$\boldsymbol{G}$, le vecteur colonne des transitions entre tous les états intermédiaires et l'état E_{of}.

En substituant l'équation (3-15) par l'expression (3-16), on obtient ceci:

$$E_{0f} = \boldsymbol{G}^T [\boldsymbol{I} - \boldsymbol{A}]^{-1} \boldsymbol{F} E_{0i} \tag{3-17}$$

Finalement, la fonction de transfert s'écrit:

$$T(D, B) = \frac{E_{0f}}{E_{0i}} = \boldsymbol{G}^T [\boldsymbol{I} - \boldsymbol{A}]^{-1} \boldsymbol{F} \tag{3-18}$$

Pour le diagramme d'état de la figure 3.3(b), la fonction de transfert et sa dérivée deviennent:

$$T(D, B) = \frac{D^5 B}{1 - 2DB} \tag{3-19-a}$$

$$\frac{dT(D, B)}{dB} = \frac{D^5}{(1 - 2DB)^2} \tag{3-19-b}$$

En posant $B = 1$, on obtient pour (3-19-a) et (3-19-b):

$$T(D, B)\big|_{B=1} = \frac{D^5}{1 - 2D} \tag{3-20-a}$$

$$\left.\frac{dT(D, B)}{dB}\right|_{B=1} = \frac{D^5}{(1 - 2D)^2} \tag{3-20-b}$$

En pratique, pour calculer la probabilité d'erreur, il suffit de prendre les premières raies solides, car ce résultat approxime bien la valeur exacte. Donc, la dérivation sur $T(D, B)$ devient nécessaire.

Par le développant en série, l'expression (3-20-a) devient:

$$T(D,B)\big|_{B=1} = D^5 + 2D^6 + 4D^7 + 8D^8 + \ldots$$

$$= \sum_{d=d_{free}}^{\infty} A_d D^d \qquad (3\text{-}21\text{-}a)$$

A partir de l'équation (3-21-a), on en déduit les informations suivantes:

1 chemin de poids 5 qui contient 1 bit d'information "1",

2 chemins de poids 6 contenant 2 bits d'information "1",

4 chemins de poids 7 contenant 3 bits d'information "1",

8 chemins de poids 8 contenant 4 bits d'information "1", ...

Le développement en série de l'équation (3-20-b) s'obtient de la forme suivante:

$$\left.\frac{dT(D,B)}{dB}\right|_{B=1} = D^5 + 4D^6 + 12D^7 + 32D^8 + \ldots$$

$$= \sum_{d=d_{free}}^{\infty} C_d D^d \qquad (3\text{-}21\text{-}b)$$

Ce qui permet de tirer comme information que:

1 chemin de poids 5 qui contient 1 bit d'information "1" au total,

2 chemins de poids 6 contenant 4 bits d'information "1"au total,

4 chemins de poids 7 contenant 12 bits d'information "1"au total,

8 chemins de poids 8 contenant 32 bits d'information "1"au total, ...

Le spectre de code sera donc illustré par le tableau suivant:

Tableau 3.4: Spectre du code convolutionnel de la figure 2.4

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
15	1024	11264
16	2048	24576
17	4096	53248
18	8192	114688
19	16384	245760
20	32768	524288

Cette technique a un très net avantage qui est celui de l'obtention simultanément

de toutes les raies du spectre. Cependant, elle se heurte à une très grande difficulté d'ordre pratique: celle de la manipulation de l'inversion d'une matrice de dimension très élevée $((2^{K-1}-1) \times (2^{K-1}-1))$, où K est la longueur de contraint du code et 2^{K-1} est le nombre d'états du code. Il est possible de faire l'inversion $[\boldsymbol{I}-\boldsymbol{A}]^{-1}$ en exprimant ce terme sous forme de série,

$$[I-A]^{-1} = I+A+A^2+A^3+\ldots \tag{3-22}$$

Dans ces cas en substituant (3-21) dans (3-18), on obtient:

$$T(D,B) = G^T F + G^T A F + G^T A^2 F + G^T A^3 F + \ldots \tag{3-23}$$

Néanmoins, la complexité du calcul reste très élevée et devient dépendante du nombre de raies.

La probabilité d'erreur du code est alors évaluée directement à partir de la fonction de transfert.

$$P_E \leq T(D,B)\Big|_{B=1,\, D^d = P_d} \tag{3-24}$$

$$P_b \leq \frac{dT(D,B)}{dB}\Bigg|_{B=1,\, D^d = P_d} \tag{3-25}$$

Pour le code de la figure 2.4, P_E et P_b sont exprimées comme suit:

$$P_E \leq \frac{D^5 B}{1-2DB}\bigg|_{B=1,\, D^d = P_d} = P_5 + 2P_6 + 4P_7 + 8P_8 + \ldots$$

$$P_b \leq \frac{d}{dB}\left(\frac{D^5 B}{1-2DB}\right)\bigg|_{B=1,\, D^d = P_d} = P_5 + 4P_6 + 12P_7 + 32P_8 + \ldots$$

3.3.2 Recherche dans l'arbre du code

La deuxième méthode qui permet de déterminer le spectre d'un code convolutionnel consiste à rechercher le spectre en explorant l'arbre du code (ou le treillis ou le diagramme d'état, ce qui revient au même).

Puisque les chemins compilés dans le spectre commencent par un bit d'information "1" et se terminent sur un noeud d'état 0, il s'agit de recueillir le poids et le nombre de bits d'information "1" cumulés sur le chemin reliant le noeud d'état 0 situé dans le sous-arbre du code débutant par un bit d'information "1". La figure 3.4 représente le demi-arbre issu de l'état 0 et débutant par un bit d'information "1" du code de la figure 2.4. Seuls les noeuds terminaux d'état 0 doivent être considérés, mais à l'exception du noeud origine.

L'algorithme d'exploration de l'arbre peut être aisément adapté à la recherche du spectre. La procédure élémentaire est la suivante: au problème de détermination des L premières raies du spectre, c'est-à-dire correspondant à un poids maximal à égale à L,

l'algorithme prolonge chacun des chemins de l'arbre jusqu'à ce que l'une des deux conditions suivantes soit rencontrée:

(1) le poids du chemin est supérieur à L;

(2) l'état du noeud visité est l'état 0.

Lorsque le noeud d'état 0 est atteint, le poids et le nombre de bits d'information "1" cumulé à ce noeud sont compilés dans le spectre.

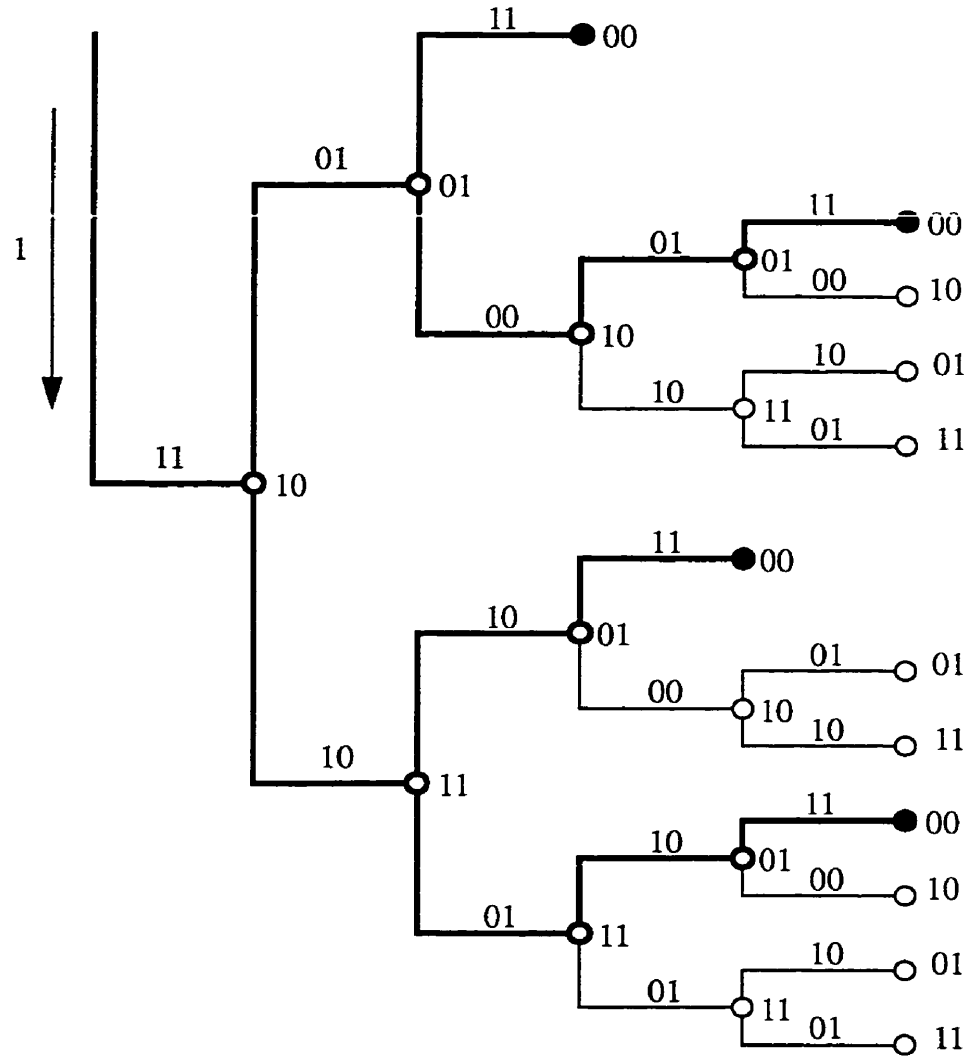


Figure 3.4: Recherche du spectre dans l'arbre

La recherche des spectres dans l'arbre du code occupe beaucoup de mémoire et de temps pour mémoriser les noeuds qui sont visités et sont utilisés après afin de trouver d'autres chemins. Mais cette méthode est pratique pour nos études.

Pierre Montreuil [20] a mis au point les logiciels qui permettent d'obtenir le spectre des codes convolutionnels classiques. Depuis, Chantal Paquin [21] a modifié ces logiciels pour les codes convolutionnels perforés. Dans ce mémoire, on a dû modifier l'algorithme "bidirectionnel" décrit dans [20] et [21] afin de déterminer le spectre des codes convolutionnels récursifs systématiques et celui des codes convolutionnels récursifs systématiques perforés. Le nouvel algorithme permet désormais de calculer en même temps le spectre de distance des codes convolutionnels non récursifs non systématiques non perforés ou perforés et celui des codes convolutionnels récursifs systématiques perforés ou non.

CHAPITRE 4

CODES CONVOLUTIONNELS RÉCURSIFS SYSTÉMATIQUES

Dans les chapitres 2 et 3, nous avons étudié les codes convolutionnels non systématiques en évaluant leurs spectres et leur probabilité d'erreur. Dans ce chapitre, nous nous intéressons aux codes convolutionnels récursifs systématiques qui possèdent la même distance libre et les mêmes nombres de chemins, mais avec un nombre total de bits d'information différents que le code convolutionnel non systématique. De plus, à de faibles rapports signal à bruit, la probabilité d'erreur d'un code convolutionnel récursif systématique est plus faible que celle d'un code convolutionnel non systématique.

Ce chapitre décrit la notion du code convolutionnel récursif systématique, la structure du codeur, la relation entre le code convolutionnel récursif systématique et le code convolutionnel non systématique, et enfin le spectre et la probabilité d'erreur.

4.1 PRINCIPES DES CODES CONVOLUTIONNELS RÉCURSIFS SYSTÉMATIQUES

Dans le chapitre 2, nous avons étudié les codes convolutionnels non systématiques. A partir d'un tel code, on peut construire un nouveau code avec une boucle

de retour interne. Ce code s'appelle le code convolutionnel récursif systématique.

Considérons un codeur convolutionnel non systématique de taux de codage R=1/2 et de longueur de contrainte K et de générateurs G_1 et G_1. Soient $x_t^{(1)}$ et $x_t^{(2)}$ deux symboles codés à l'instant t:

$$x_t^{(1)} = \sum_{j=0}^{K-1} g_{1j} \cdot u_{t-j} \qquad (4\text{-}1\text{-a})$$

$$x_t^{(2)} = \sum_{j=0}^{K-1} g_{2j} \cdot u_{t-j} \qquad (4\text{-}1\text{-b})$$

où u_t est le symbole présent à l'entrée du codeur à l'instant t et g_{ij} les coefficients binaires de générateurs G_i, $g_{ij} \in \{0, 1\}$, $i = 1, 2, j = 1, 2, \ldots, K$.

Suivant la définition de Forney [5], les expressions (4-1-a) et (4-1-b) peuvent se mettre sous la forme:

$$X_1 = G_1(D) \cdot U \qquad (4\text{-}2\text{-a})$$

$$X_2 = G_2(D) \cdot U \qquad (4\text{-}2\text{-b})$$

Selon Forney, il existe un code convolutionnel récursif systématique, avec un boucle de retour interne, ayant la même distance libre que le code convolutionnel non systématique [5] [22].

En divisant les équations (4-2-a) et (4-2-b) par $G_1(D)$, on obtient:

$$\tilde{X}_1 = \frac{X_1}{G_1(D)} = U \tag{4-3-a}$$

$$\tilde{X}_2 = \frac{X_2}{G_1(D)} = \frac{G_2(D)\,U}{G_1(D)} \tag{4-3-b}$$

En introduisant la séquence A définie par:

$$A = \frac{U}{G_1(D)} \tag{4-4}$$

les séquences sortant du codeur peuvent finalement s'écrire sous la forme:

$$\tilde{X}_1 = U \tag{4-5-a}$$

$$\tilde{X}_2 = G_2(D)\,A \tag{4-5-b}$$

La caractère récursif du code ainsi construit est illustré par la relation (4-4). En effet, dans un espace temporel, nous pouvons écrire la relation (4-4) comme tel:

$$u_t = \sum_{j=0}^{K-1} g_{1j} \cdot a_{t-j} \tag{4-6}$$

En faisant l'hypothèse que $g_{10} = 1$, le symbole a_t peut s'exprimer récursivement en fonction des symboles a_{t-j}, $j = 1, 3, \ldots, K\text{-}1$ et du symbole u_t:

$$a_t = u_t + \sum_{j=1}^{K-1} g_{1j} \cdot a_{t-j} \tag{4-7}$$

Pour un code convolutionnel récursif systématique, les symboles a_t sont désormais contenus dans le registre à décalage du codeur.

D'après les équations (4-5-a), (4-5-b) et (4-6), les sorties du codeur à l'instant t peuvent s'exprimer sous la forme:

$$x_t^{(1)} = u_t = \sum_{j=0}^{K-1} g_{1j} \cdot a_{t-j} \tag{4-8-a}$$

$$x_t^{(2)} = \sum_{j=0}^{K-1} g_{2j} \cdot a_{t-j} \tag{4-8-b}$$

Un code convolutionnel récursif systématique est construit à partir d'un code convolutionnel non systématique en conservant les mêmes générateurs g_{ij} ($i = 1, 2$), $i = 1, 2, j = 1, 2, \ldots, K$, mais en substituant les symboles a_t aux symboles d'information u_t.

La figure 4.1 illustre la construction du code convolutionnel récursif systématique à partir du code convolutionnel non systématique de la figure 2.4.

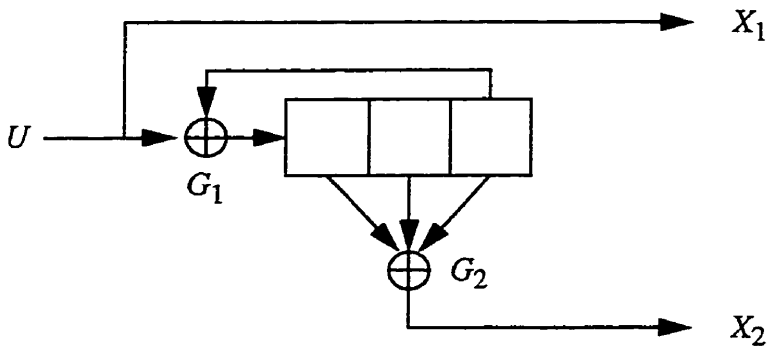


Figure 4.1: Codeur convolutionnel récursif systématique

$R = 1/2, K = 3, \nu = 2, G = [1, (1+D+D^2)/(1+D^2)]$

D'ailleurs, nous pouvons utiliser aussi la deuxième générateur pour la boucle de retour. La construction d'un code convolutionnel récursif systématique à partir d'un code convolutionnel non systématique est comme nous avons démonstré avant. La figure 4.2 présente cette construction. Dans ce travail, nous utilisons la première construction.

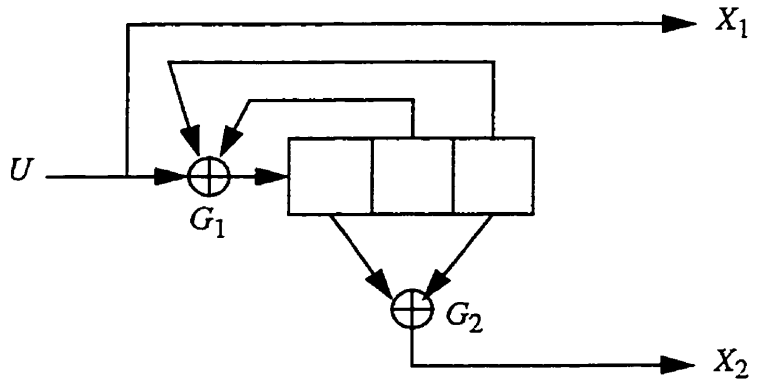


Figure 4.2: Codeur convolutionnel récursif systématique

$R = 1/2, K = 3, \nu = 2, G = [1, (1+D^2)/(1+D+D^2)]$

En général, à partir d'un code convolutionnel non systématique de taux de codage $R = 1/v$, de longueur de contrainte K, le code convolutionnel récursif systématique peut être construit de la même façon.

Soit un code convolutionnel de taux de codage $R = 1/n$, de longueur de contrainte K et soient $x_t^{(1)}, x_t^{(2)}, \ldots, x_t^{(n)}$ ses n symboles codés à l'instant t. Nous pouvons établir la relation suivante:

$$x_t^{(i)} = \sum_{j=0}^{K-1} g_{ij} \cdot u_{t-j} \tag{4-9}$$

où g_{ij} sont les coefficients binaires de générateurs G_i, $g_{ij} \in \{0, 1\}$, $i = 1, 2, \ldots, n$.

L'équation (4-9) en fonction de D s'écrit:

$$X_i = G_i(D) \cdot U \tag{4-10}$$

En divisant l'équation (4-10) par $G_1(D)$, on obtient:

$$\tilde{X}_1 = \frac{X_1}{G_1(D)} = U \tag{4-11-a}$$

$$\tilde{X}_i = \frac{X_i}{G_1(D)} = \frac{G_i(D)\,U}{G_1(D)} \tag{4-11-b}$$

où $i = 2, 3, \ldots, n$.

En substituant l'équation (4-4) dans (4-11-b), on obtient:

$$\tilde{X}_i = G_i(D)\,A \tag{4-12}$$

Ainsi, les sorties du codeur à l'instant t peuvent s'exprimer sous la forme:

$$x_t^{(1)} = u_t = \sum_{j=0}^{K-1} g_{1j} \cdot a_{t-j} \tag{4-13-a}$$

$$x_t^{(i)} = \sum_{j=0}^{K-1} g_{ij} \cdot a_{t-j} \tag{4-13-b}$$

où $i = 2, 3, \ldots, n$.

4.2 RELATION ENTRE UN CODE CONVOLUTIONNEL RÉCURSIF SYSTÉMATIQUE ET UN CODE CONVOLUTIONNEL NON SYSTÉMATIQUE

A la figure 4.3, nous avons illustré le diagramme d'état du code de la figure 4.1. En comparant avec le diagramme d'état de la figure 2.5, on peut constater que les symboles codés sur chacune des branches sont identiques à ceux du code convolutionnel non systématique dont il est issu, mais que les bits d'information sont différents. Les représentations en treillis et en arbre d'encodage du code sont présentées par les figures 4.4 et 4.5.

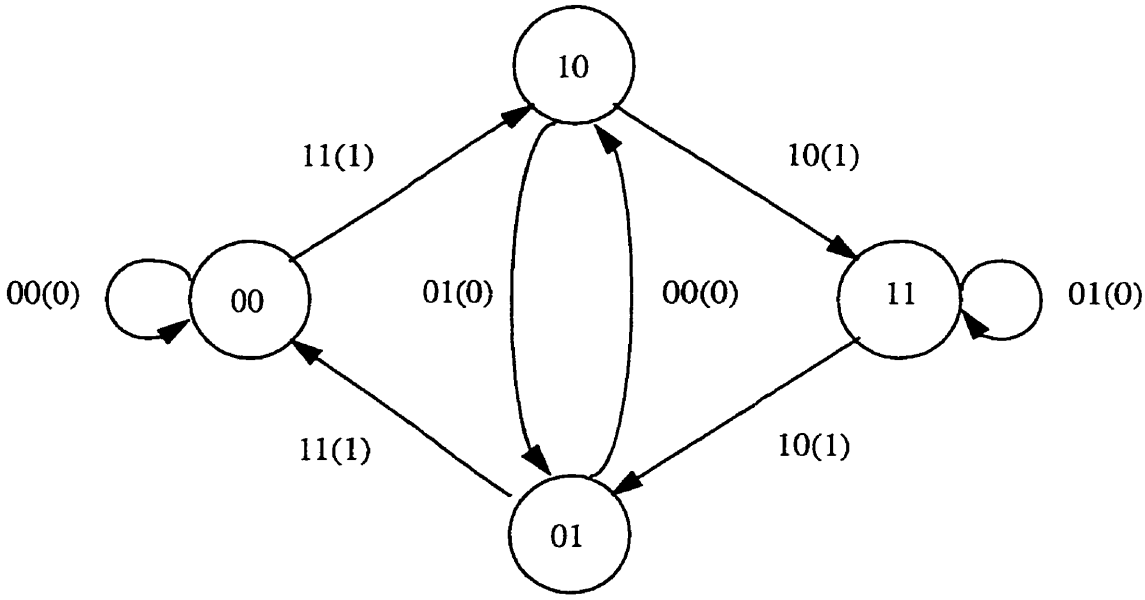


Figure 4.3: Diagramme d'état du codeur de la figure 4.1

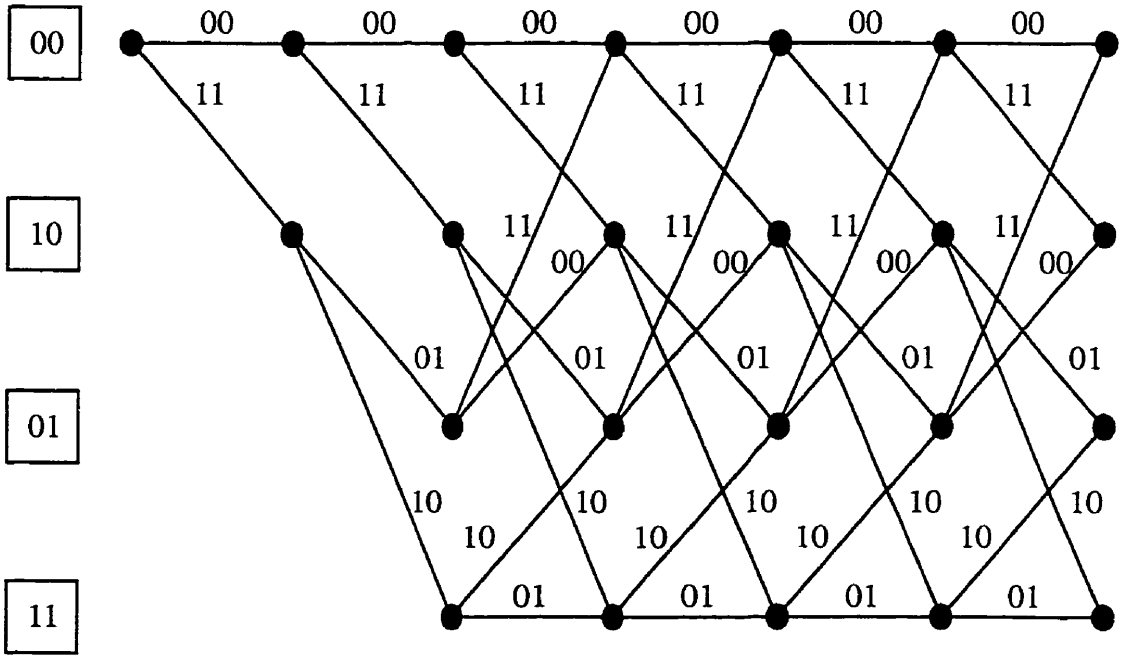


Figure 4.4: Représentation en treillis du codeur de la figure 4.1

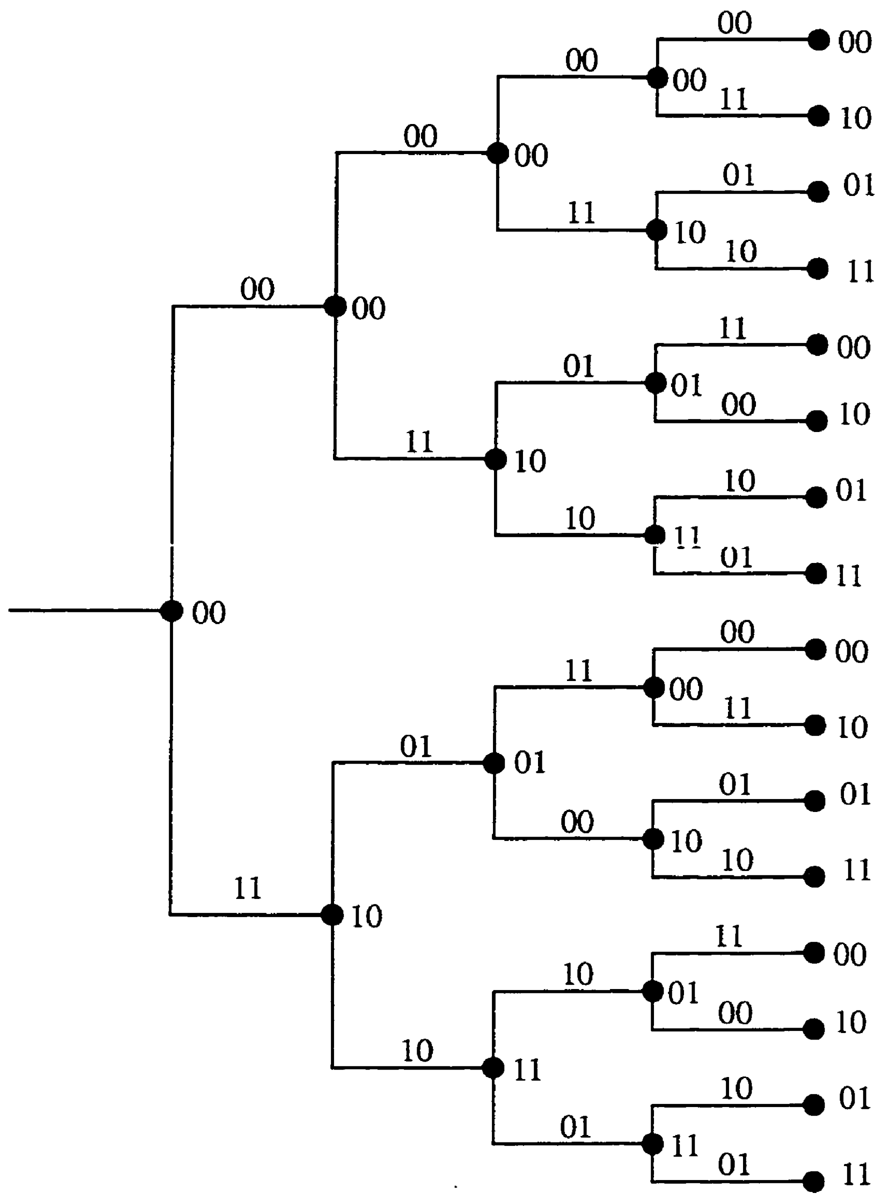


Figure 4.5: Représentation en arbre du codeur de la figure 4.1

D'après les figures 4.4 et 4.5, nous remarquons que le code convolutionnel récursif systématique possède les mêmes structures de treillis et d'arbre que le code convolutionnel non systématique à partir duquel il a été construit. En d'autres mots, pour une transition entre les deux mêmes états, les mots de code sont identiques.

En utilisant la même méthode du chapitre 3, le diagramme d'état modifié de la figure 4.3 peut être représenté comme ci-dessous:

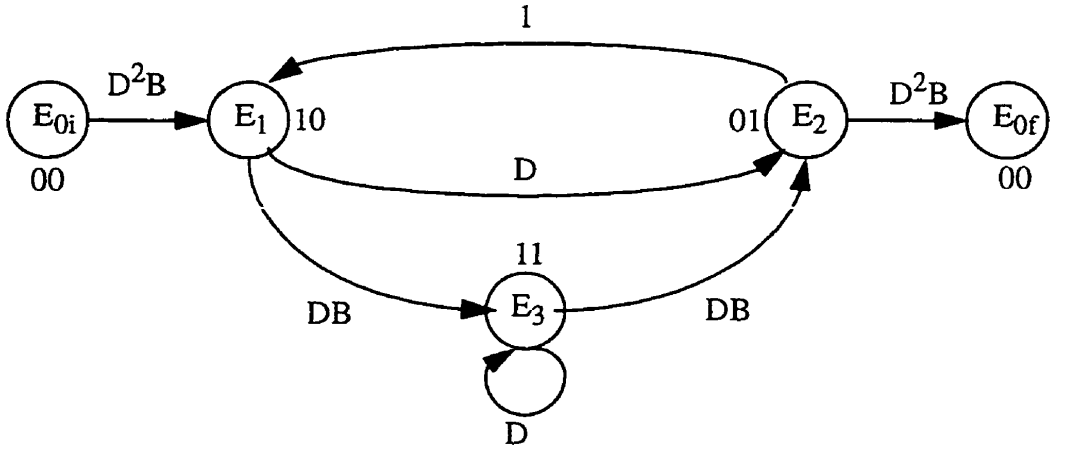


Figure 4.6: Diagramme d'état modifié du codeur de la figure 4.1

La fonction de transfert est alors égale à:

$$T(D,B) = \frac{D^5B^2(DB^2 - D + 1)}{1 - D^2B^2 + D^2 - 2D} \qquad (4\text{-}14)$$

Le développement de cette fonction de transfert sera présenté à Annexe I.

En développant en série la fonction de transfert ainsi que sa dérivée, on obtient:

$$T(D,B)\big|_{B=1} = \frac{D^5}{1-2D} = \sum_{d=d_{free}}^{\infty} A_d D^d \qquad (4\text{-}15\text{-a})$$

Cette expression est identique que l'expression (3-20-a).

$$\left.\frac{dT(D,B)}{dB}\right|_{B=1} = \frac{2D^5(1-D-D^2)}{(1-2D)^2} = \sum_{d=d_{free}}^{\infty} C_d D^d \qquad (4\text{-}15\text{-b})$$

Cette expression est différente que l'expression (3-20-b).

Le spectre de ce code est donné dans le tableau 4.1:

Tableau 4.1: Spectre du code convolutionnel récursif systématique de la figure 4.1

d	A_d	C_d
5	1	2
6	2	6
7	4	14
8	8	32
9	16	72
10	32	160
11	64	352
12	128	768
13	256	1664
14	512	3584
15	1024	7680
16	2048	16384
17	4096	34816
18	8192	73728
19	16384	155648
20	32768	327680

En examinant les fonctions de transfert ((3-20) et (4-15)) et les spectres du code convolutionnel non systématique et ceux du code convolutionnel récursif systématique (tableau 3.4, tableau 4.1), nous pouvons constater que les deux codes possèdent la même distance libre ($d_{free} = 5$), les mêmes nombres de chemins (A_d), mais un nombre total de bits d'information "1" différents (C_d).

Pour déterminer le spectre des codes convolutionnels récursifs systématiques, on a modifié l'algorithme "bidirectionnel" décrit dans [20]. Le nouvel algorithme permet désormais de calculer en même temps le spectre de distance des codes convolutionnels non récursifs non systématiques et des codes convolutionnels récursifs systématiques. Cet algorithme est appelé l'algorithme "bidirectionnel modifié".

Le tableau 4.2 décrit les spectres de codes convolutionnels non systématiques et ceux de codes convolutionnels récursifs systématiques ayant un taux de codage $R = 1/2$ et des longueurs de contrainte K allant de 3 à 16. Les générateurs G_1 et G_2 utilisés sont ceux des codes convolutionnels non systématiques ayant une distance libre maximale [23] [24] [25].

Tableau 4.2: Spectres de codes convolutionnels non systématiques et des codes convolutionnels récursifs systématiques, $R = 1/2$, $3 \leq K \leq 9$

K	G_1 G_2	d_{free}	(A_n) $[C_n]$ $\{Cr_n\}$
3	5 7	5	(1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384) [1, 4, 12, 32, 80, 192, 448, 1024, 2304, 5120, 11264, 24576, 53248, 114688, 245760] {2, 6, 14, 32, 72, 160, 352, 768, 1664, 3584, 7680, 16384, 34816, 73728, 155648}
4	15 17	6	(1, 3, 5, 11, 25, 55, 121, 267, 589, 1299, 2865, 6319, 13937, 30739, 67797) [2, 7, 18, 49, 130, 333, 836, 2069, 5060, 12255, 29444, 70267, 166726, 393635, 925334] {4, 9, 20, 51, 124, 303, 728, 1739, 4134, 9771, 22990, 53885, 125858, 293049, 680440}
5	23 35	7	(2, 3, 4, 16, 37, 68, 176, 432, 925, 2156, 5153, 11696, 26868, 62885, 145085) [4, 12, 20, 72, 225, 500, 1324, 3680, 8967, 22270, 57403, 142234, 348830, 867106, 2134239] {8, 12, 16, 84, 213, 406, 1156, 3104, 7021, 17372, 44427, 106518, 257200, 634556, 1537281}
6	53 75	8	(1, 8, 7, 12, 48, 95, 281, 605, 1272, 3334, 7615, 18131, 43197, 99210, 237248) [2, 36, 32, 62, 332, 701, 2342, 5503, 12506, 36234, 88576, 225685, 574994, 1400192, 3554210] {4, 34, 32, 62, 288, 604, 1884, 4430, 9926, 27794, 67380, 168606, 424768, 1025664, 2570672}
7	133 171	10	(11, 0, 38, 0, 193, 0, 1331, 0, 7275, 0, 40406, 0, 234969, 0, 1337714) [36, 0, 211, 0, 1404, 0, 11633, 0, 77433, 0, 502690, 0, 3322763, 0, 21292910] {60, 0, 223, 0, 1368, 0, 10963, 0, 66171, 0, 408918, 0, 2619965, 0, 16222096}
8	247 371	10	(1, 6, 12, 26, 52, 132, 317, 730, 1823, 4446, 10739, 25358, 60773, 146396, 350399) [2, 22, 60, 148, 340, 1008, 2642, 6748, 18312, 48478, 126364, 320062, 821350, 2102864, 5335734] {6, 28, 70, 182, 360, 984, 2530, 6156, 16308, 41924, 107014, 265396, 666098, 1677978, 4189876}
9	561 753	12	(11, 0, 50, 0, 286, 0, 1630, 0, 9639, 0, 55152, 0, 320782) [33, 0, 281, 0, 2179, 0, 15035, 0, 105166, 0, 692330, 0, 4580007] {67, 0, 349, 0, 2295, 0, 14575, 0, 96680, 0, 606538, 0, 3848633}

Tableau 4.2 (suite): Spectre des codes convolutionnels non systématiques et des codes convolutionnels récursifs systématiques, $R = 1/2$, $10 \leq K \leq 16$

K	G_1 G_2	d_{free}	(A_n) $[C_n]$ $\{Cr_n\}$
10	1167 1545	12	(2, 8, 15, 35, 68, 170, 458, 1084, 2574, 6177, 14939, 36200, 86856) [14, 26, 74, 257, 496, 1378, 4122, 10832, 27988, 72209, 186920, 483102, 1234736] {14, 50, 96, 271, 552, 1428, 4106, 10192, 25644, 64627, 163408, 414532, 1038094}
11	2335 3661	14	(21, 0, 74, 0, 454, 0, 2687, 0, 15629, 0, 90518, 0, 526556) [94, 0, 463, 0, 3783, 0, 26711, 0, 181571, 0, 1207474, 0, 7919894] {148, 0, 585, 0, 4085, 0, 26881, 0, 172285, 0, 1086924, 0, 6849856}
12	4335 5723	15	(16, 31, 44, 129, 309, 697, 1713, 4175, 10158, 24508, 58600, 141960, 343347) [76, 180, 374, 1142, 2783, 6836, 18709, 49242, 128178, 329408, 836478, 2151230, 5497355] {122, 234, 360, 1134, 2875, 6854, 17637, 45154, 114532, 289116, 721920, 1818368, 4564999}
13	10533 17661	16	(33, 0, 111, 0, 779, 0, 4128, 0, 24173, 0, 142500, 0, 828402) [152, 0, 971, 0, 6933, 0, 45436, 0, 303435, 0, 2036131, 0, 13256560] {268, 0, 1049, 0, 7973, 0, 46662, 0, 296423, 0, 1892487, 0, 11821688}
14	21675 27123	16	(4, 17, 35, 76, 193, 454, 1047, 2624, 6138, 14944, 36179, 86640, 210568) [22, 99, 218, 608, 1724, 4404, 11108, 30438, 75942, 196714, 507232, 1289364, 3311290] {34, 143, 310, 722, 1890, 4708, 11420, 30044, 73432, 185730, 467800, 1163266, 2933186}
15	44735 63057	18	(26, 0, 165, 0, 845, 0, 4844, 0, 28513, 0, 166583, 0, 968884) [133, 0, 1321, 0, 7901, 0, 54864, 0, 370057, 0, 2450650, 0, 15893608] {229, 0, 1627, 0, 9191, 0, 57526, 0, 367507, 0, 2309798, 0, 14416882}
16	126723 152711	19	(30, 67, 54, 167, 632, 1402, 2812, 7041, 18178, 43631, 104526, 251134, 605947) [174, 420, 534, 1712, 5838, 14210, 32898, 87786, 237228, 609868, 1556396, 3945414, 10039681] {292, 666, 552, 1880, 7304, 16870, 35336, 91922, 245526, 612890, 1520392, 3779288, 9423360}

4.3 PROBABILITÉ D'ERREUR DES CODES CONVOLUTIONNELS RÉCURSIFS SYSTÉMATIQUES

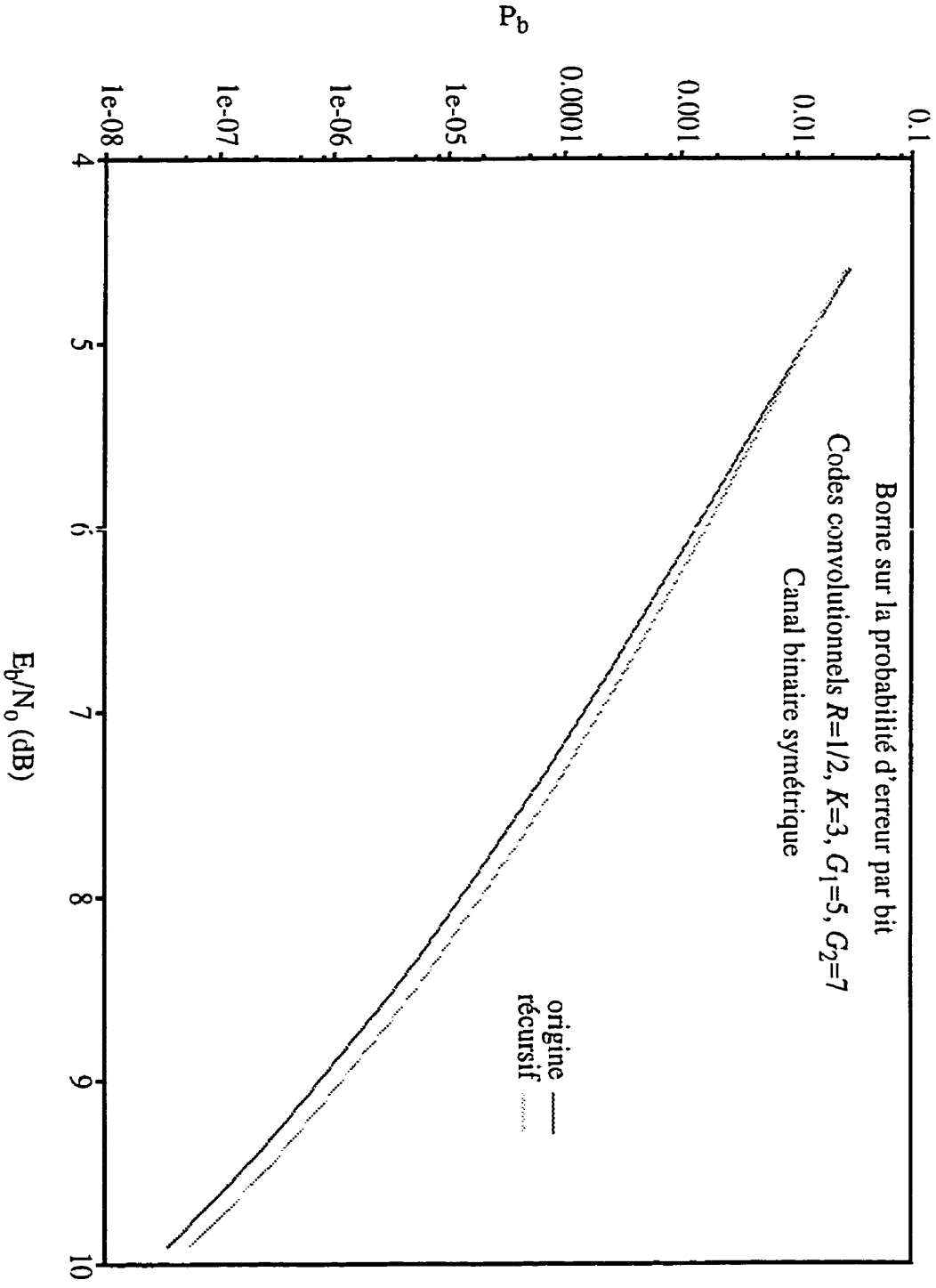
La probabilité d'erreur après décodage peut être déterminée à l'aide des mêmes bornes union qui sont présentées au chapitre 3. En pratique, pour la grande majorité des codes convolutionnels, il est impossible de déterminer la fonction de transfert complète. On se contente habituellement des premiers termes de la série A_d et C_d, dont on ne connait souvent d'ailleurs que les premiers termes. La borne supérieure permet uniquement d'approximer la probabilité d'erreur, mais l'approximation est assez bonne lorsque le rapport signal à bruit est grand. Pour de faibles rapport signal à bruit, l'approximation n'est plus valide, et on fait appel dans ce cas à des simulations pour évaluer les performances d'erreur du code. Le programme de simulation (algorithme de Viterbi) permet d'estimer la probabilité d'erreur en fonction du rapport signal à bruit E_b/N_0 pour n'importe quels types de codes (convolutionnels, non systématiques ou récursifs systématiques), et pour différents types de canal: canal binaire symétrique et le canal à bruit blanc gaussien.

La borne supérieure sur la probabilité d'erreur par bit est comme suit:

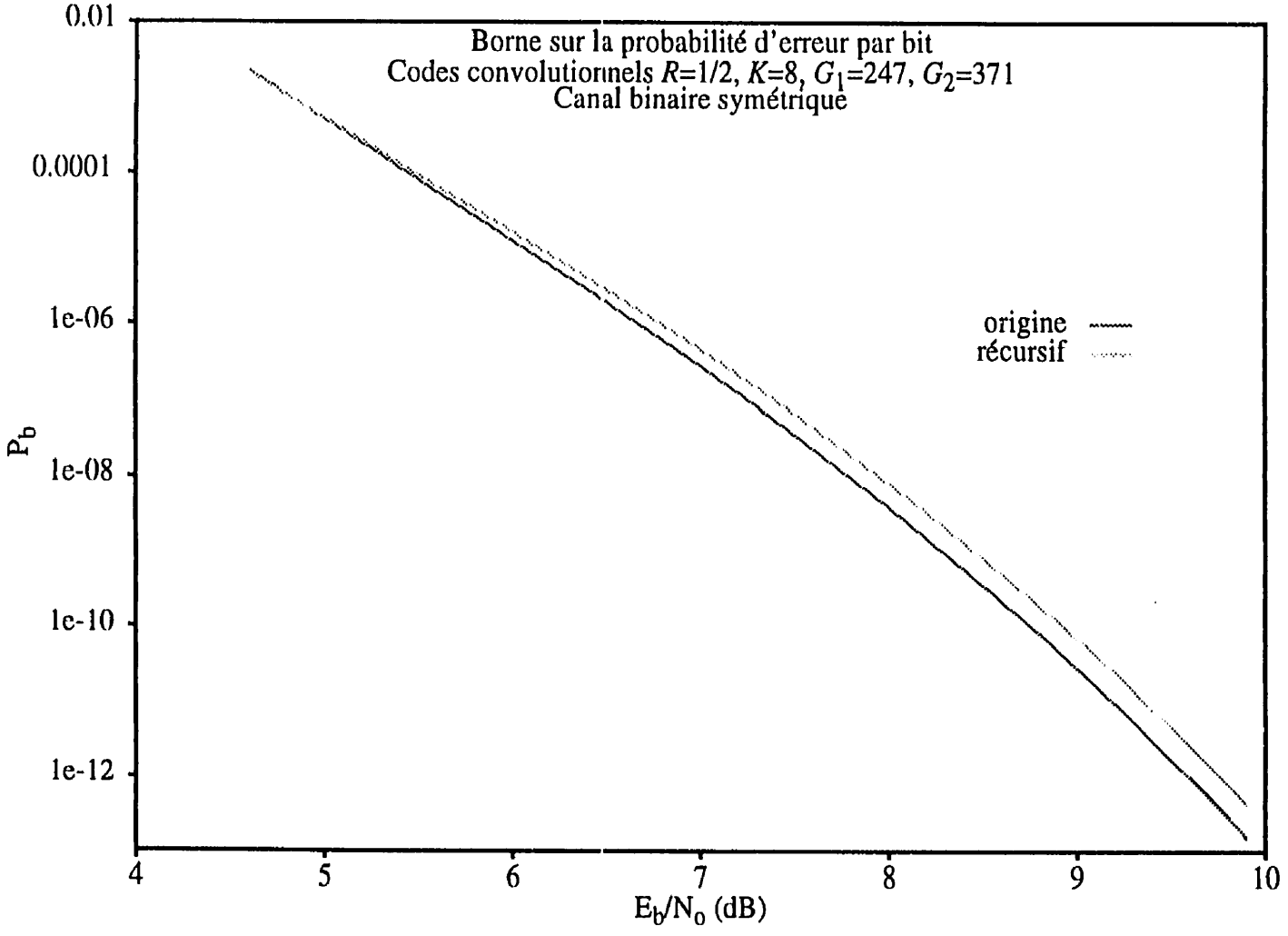
$$P_b = \left.\frac{dT(D,B)}{dB}\right|_{B=1,\, D^d = P_d} \approx \sum_{d=d_{free}}^{\infty} C_d P_d \qquad (4\text{-}16)$$

Les résultats de la simulation permettent de déterminer les performances d'erreur de certains codes récursifs décrits au tableau 4.2. Sur les figures 4.7(a) et 4.7(b), on présente les performances en fonction du rapport signal à bruit E_b/N_0 de ces codes travers le canal binaire symétrique. Sur les figures 4.8(a) et 4.8(b), ces performances d'erreur sont

calculées pour un canal à bruit blanc gaussien.

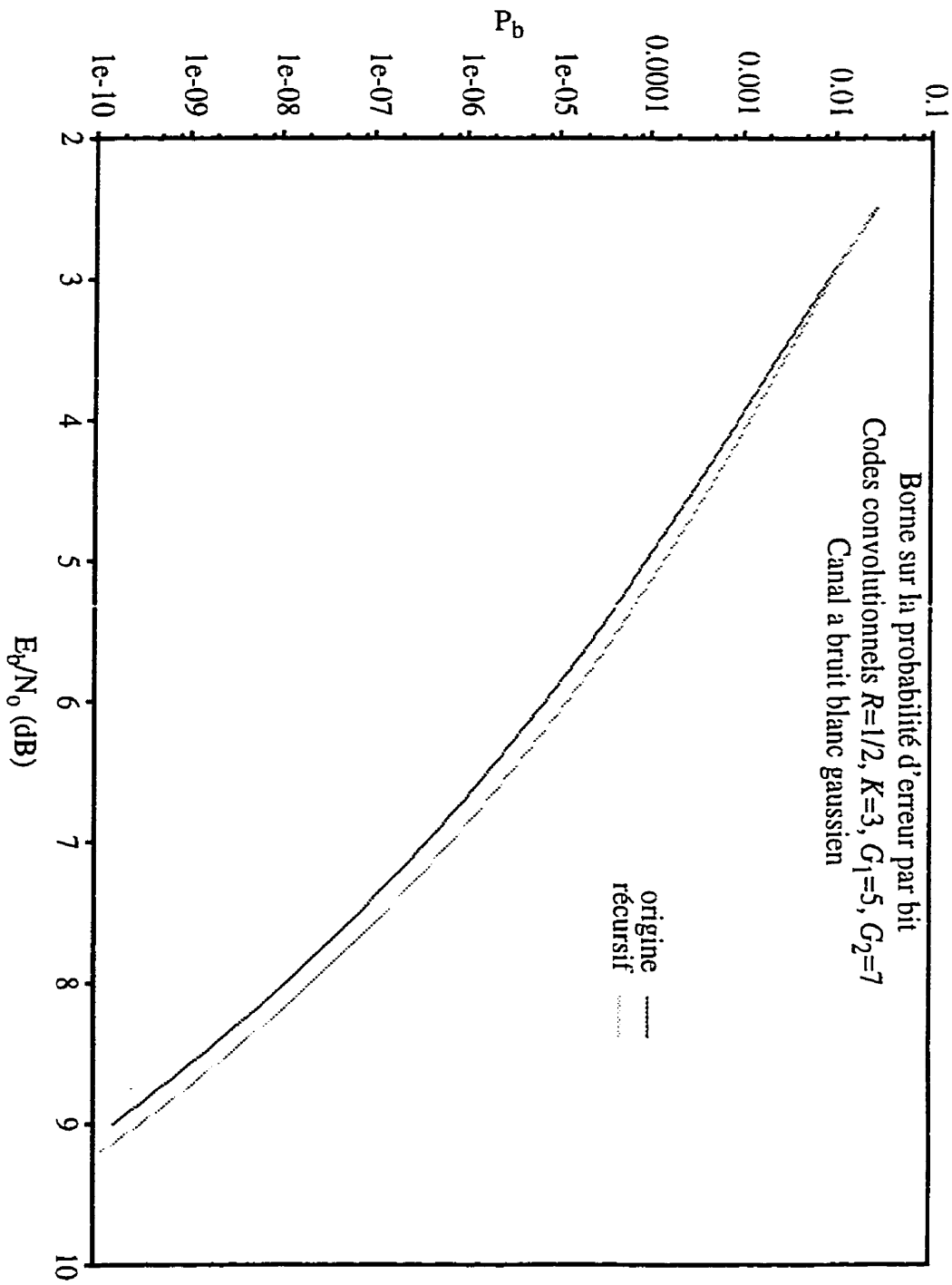


(a) $R = 1/2$, $K = 3$, $G_1 = 5$, $G_2 = 7$

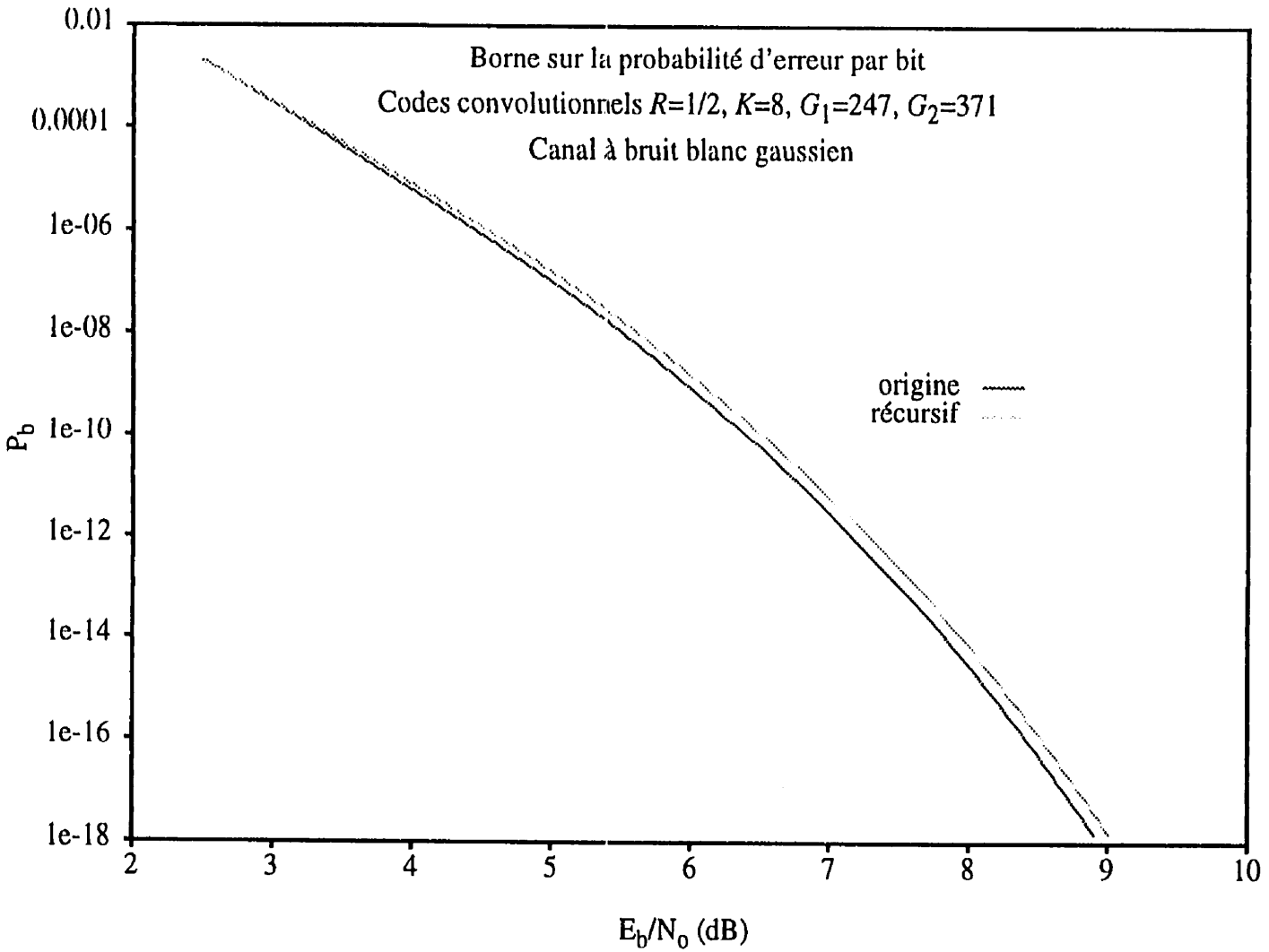


(b) $R = 1/2$, $K = 8$, $G_1 = 247$, $G_2 = 371$

Figure 4.7: Les performances des codes dans un canal binaire symétrique



(a) $R = 1/2$, $K = 3$, $G_1 = 5$, $G_2 = 7$



(b) $R = 1/2$, $K = 8$, $G_1 = 247$, $G_2 = 371$

Figure 4.8: Les performances des codes dans un canal à bruit blanc gaussien

Lorsqu'on observe ces figures, on constate que les codes convolutionnels récursifs systématiques fournissent une légère amélioration de la probabilité d'erreur par bit par rapport aux codes convolutionnels non systématiques lorsque le rapport signal à bruit et faible. En pratique, les deux codes peuvent être toutefois considérés comme équivalents car ils ont la même distance libre. Les séquences d'erreur qui sont produites après décodage d'un code convolutionnel récursif systématique sont presque toujours de poids plus faible que celles d'un code convolutionnel non systématique. C'est ce que conduit à une probabilité d'erreur par bit inférieure. Cette propriété est intéressante lorsque l'on envisage de faire la concaténation des codes.

CHAPITRE 5

PERFORATION DES CODES CONVOLUTIONNELS RÉCURSIFS SYSTÉMATIQUES

Un code convolutionnel de taux de codage élevé diminue l'expansion de la largeur de bande du canal, mais introduit moins de redondance ce qui réduit le pouvoir de correction d'erreur du code tout en augmentant la complexité de décodage. Ces difficultés peuvent être éliminées par l'utilisation de codes perforés.

Les codes convolutionnels perforés sont une classe de codes convolutionnels de taux de codage élevé $R = b/v$ qui sont obtenus à partir de codes convolutionnels de faible taux de codage $R_0 = 1/v_0$, par élimination périodique (perforation) de certains symboles codés à la sortie de ce codeur de faible taux de codage [26]. Cette élimination est effectuée par une matrice binaire appelée patron de perforation. Le code perforé dépend du code origine, et du patron de perforation [27].

Dans ce chapitre, on présente le principe de la perforation, le spectre et la performance d'erreur des codes perforés récursifs.

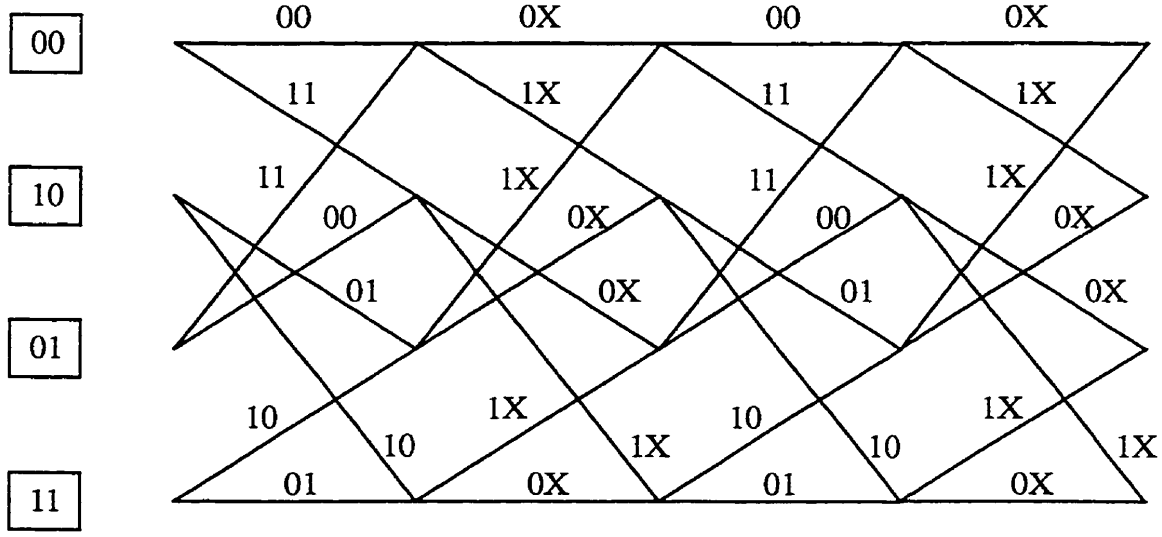
5.1 PRINCIPE DE LA PERFORATION

Pour comprendre l'effet de la perforation, prenons un exemple. Considérons le codeur convolutionnel illustré à la figure 2.4. Ce codeur produit un code de taux de codage $R_0 = 1/2$ dont le treillis a été représenté à la figure 2.7.

Supposons qu'on choisisse la matrice

$$P_1 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

comme patron de perforation. Les éléments de la première colonne de P_1 sont "1", cela veut dire que l'on ne perfore pas les deux bits de la première branche de treillis. Pour la deuxième colonne, on ne perfore pas le premier bit de la deuxième branche de treillis, mais on perfore le deuxième bit, comme illustré à la figure 5.1 par le symbole "X". Le taux de codage résultant est $R = 2/3$.



X: symbole perforé

Figure 5.1: Treillis du code perforé avec P_1

Considérons maintenant les transitions de ce code après l'entrée de deux bits d'information. On peut représenter ces transitions sur un treillis qui possède encore quatre états mais dont les états sont reliés par quatre transitions possibles, correspondant aux quatre combinaisons des deux bits entrés successivement dans le codeur. Portons sur ces transitions les trois symboles codés qui subsistent après l'entrée de deux bits et la perforation. Le nouveau treillis est représenté à la figure 5.2.

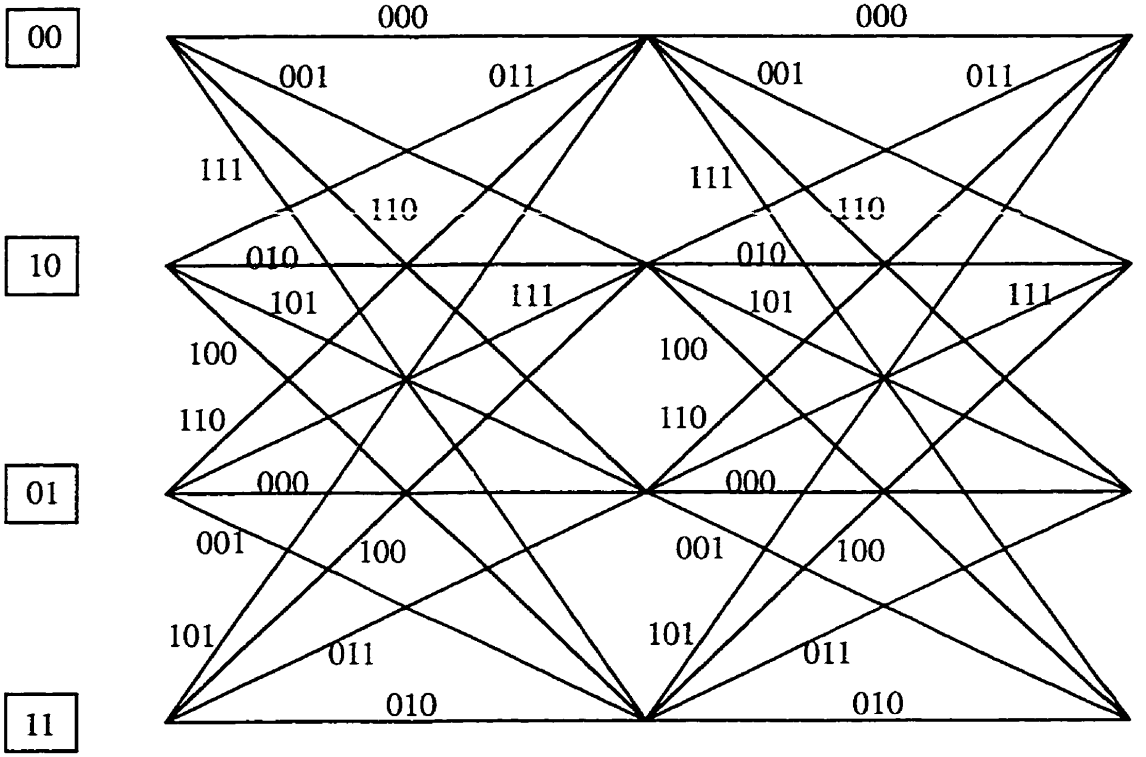


Figure 5.2: Treillis du code $R = 2/3$ après la perforation du code de la figure 2.4 avec P_1

Ce treillis correspond tout à fait à celui d'un code convolutionnel à deux entrées et à trois sorties, c'est-à-dire de taux de codage $R = 2/3$. Ce code peut être considéré comme un véritable code de taux de codage élevé.

Le principe des codes perforés est qu'à partir d'un code convolutionnel de faible taux de codage $R_0 = 1/v_0$ que l'on appelle code origine, on perfore périodiquement un nombre $S = bv_0 - v$ de symboles dans toutes les b branches du code origine. Ces groupes de b branches deviennent les branches du nouveau code perforé de taux de codage élevé. Le taux de codage de code perforé est $R = b/v$.

Les codes perforés peuvent être considérés de deux manières différentes selon les hypothèses suivantes [21]: la première hypothèse veut qu'un code perforé soit considéré comme un code de faible taux dont la redondance a été réduite. Son treillis est illustré à la figure 5.1. Quant à la deuxième hypothèse, un code perforé est un vrai code de taux élevé (R = b/v), dont le treillis est représenté à la figure 5.2. On considérera toujours cette dernière hypothèse dans le cadre de ce travail.

5.2 PATRON DE PERFORATION

Le code de taux de codage élevé qui est effectivement obtenu dépend du code origine utilisé au départ, du nombre de branches regroupées ensemble et de la position des symboles perforés au sein de ces branches. Les deux dernières informations composent une matrice binaire de b colonnes par v_0 lignes. Cette matrice est appelée le patron de perforation.

Les éléments d'un patron de perforation P sont définis ainsi:

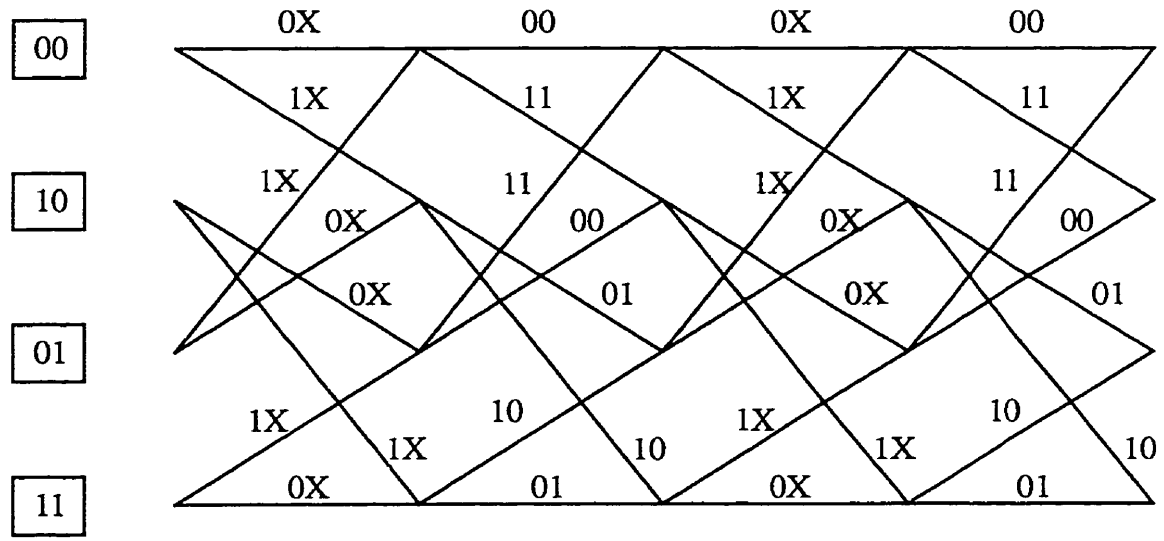
$$p_{ij} = \begin{cases} 0, \text{ si le } j^{\text{ème}} \text{ symbole de la } i^{\text{ème}} \text{ branche est perforé} \\ 1, \text{ si le } j^{\text{ème}} \text{ symbole de la } i^{\text{ème}} \text{ branche n'est pas perforé} \end{cases}$$

où $i = 1, 2, \ldots, b$; $j = 1, 2, \ldots, v_0$. On a autant de colonnes qu'il y a de branches regroupées, c'est-à-dire que b est le nombre de colonnes dans la matrice. Le nombre total de "1" dans la matrice est égal à v.

Si on choisit le patron de perforation P_2 suivant:

$$P_2 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

avec le même code origine, on obtient un code perforé qui est différent du code de la figure 5.2, mais dont le taux de codage est le même. Le treillis pour ce deuxième code est représenté aux figures 5.3 et 5.4.



X: symbole perforé

Figure 5.3: Treillis du code perforé avec P_2

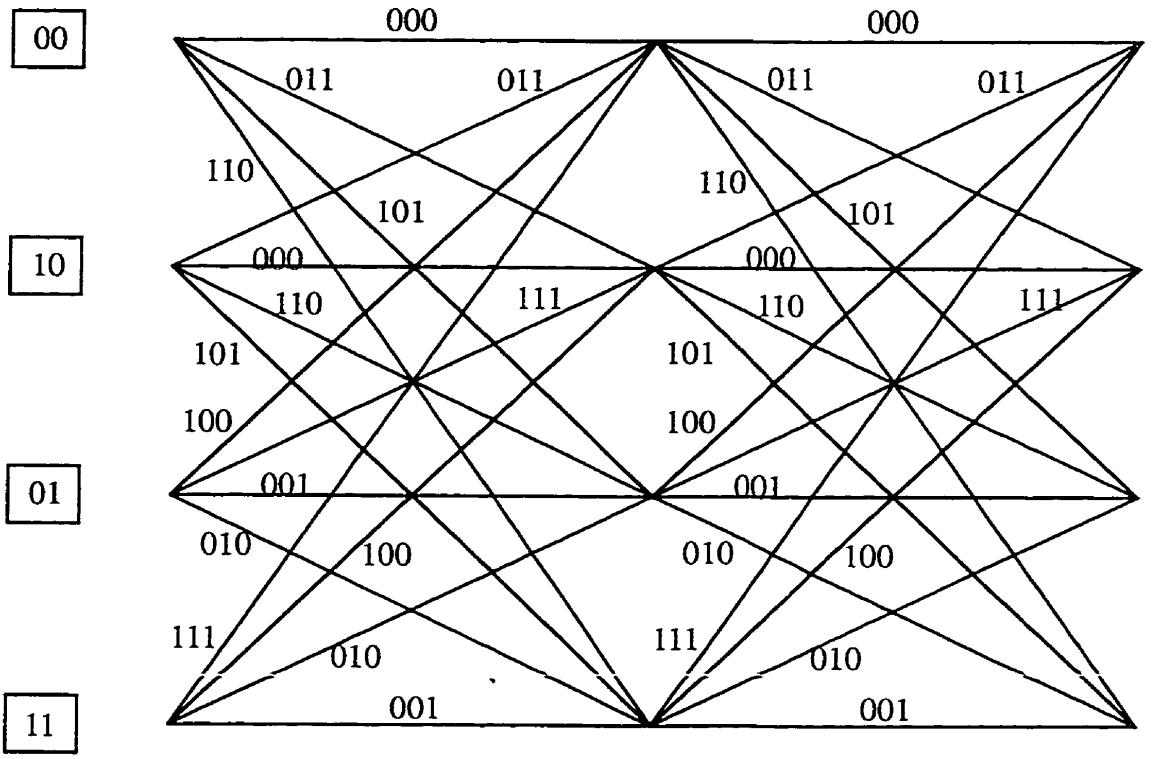


Figure 5.4: Treillis du code $R = 2/3$ après la perforation du code de la figure 2.4 avec P_2

Si on utilise la patron de perforation suivant

$$P_3 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

avec le même code origine, on obtient un codes perforé de taux de codage $R = 3/4$.

On constate ainsi qu'il est très simple de changer le taux de codage du code perforé en changeant simplement le patron de perforation. Cette flexibilité est une caractéristique importante des codes perforés.

5.3 CODAGE CONVOLUTIONNEL RÉCURSIF SYSTÉMATIQUE PERFORÉ

Un codeur convolutionnel récursif systématique perforé est illustré à la figure 5.5. La fonction du perforateur est d'éliminer les symboles codés du code origine correspondants aux symboles "0" du patron de perforation.

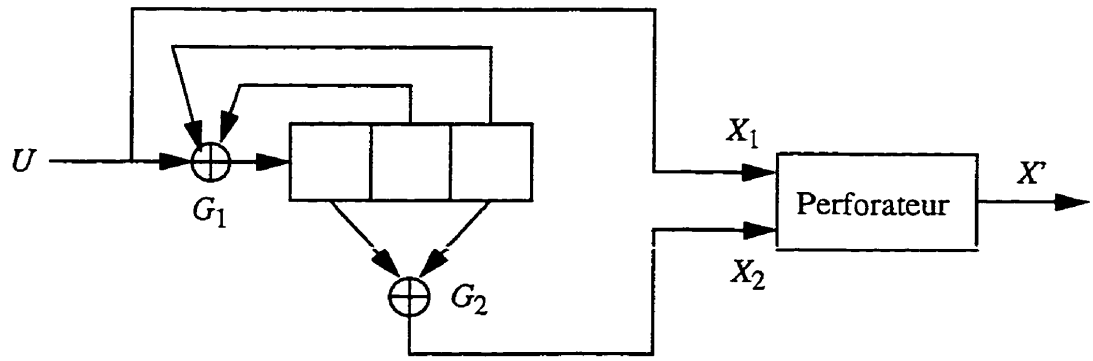


Figure 5.5: Codeur convolutionnel récursif systématique perforé

$R_0 = 1/2$, $K = 3$, $\nu_0 = 2$, $G = [1, (1+D^2)/(1+D+D^2)]$

Grâce la deuxième hypothèse décrite à la section dernière, le diagramme d'état de ce code perforé est équivalent à celui d'un code de taux élevé b/ν. Pour tracer le diagramme d'état de ce code perforé, il suffit de considérer les transitions par groupes de b branches. Ces groupes de b branches que l'on appelle super-branches portent les ν symboles codés. Le patron de perforation a donc b colonnes et ν_0 rangées.

Supposons que le patron de perforation soit P_1, le diagramme d'état du code de la figure 5.5 est alors représenté à la figure 5.6.

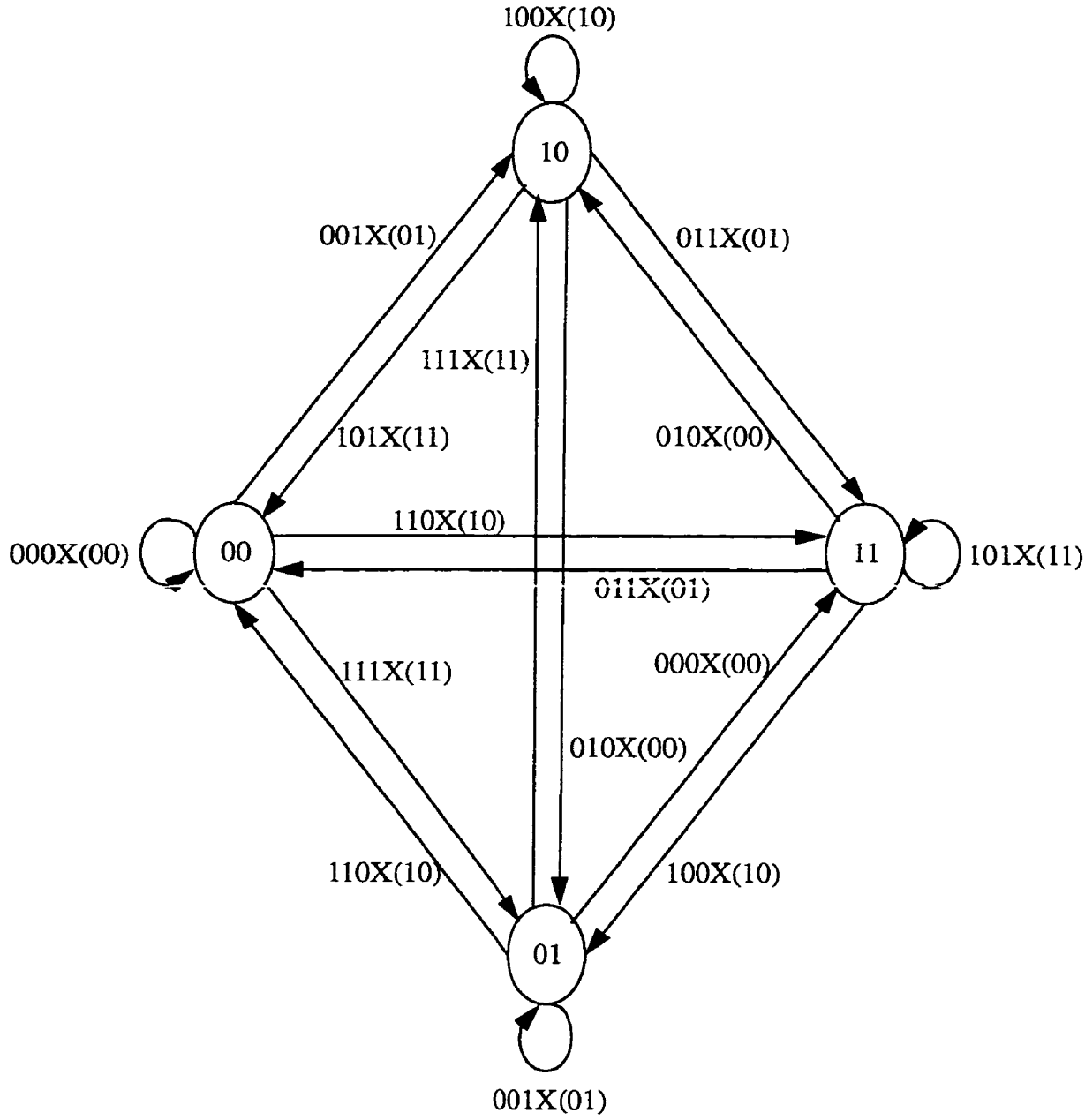


Figure 5.6: Diagramme d'état d'un code perforé $R = 2/3$

En interprétant les codes perforés comme de vrais codes de taux de codage élevé, la représentation des mots de code dans l'arbre ou dans le treillis du code origine est

légèrement modifiée. Les branches sont formées de super-branches qui regroupent les b branches du code origine. Les représentations de l'arbre et du treillis de ce code perforé sont illustrées aux figures 5.7 et 5.8 respectivement.

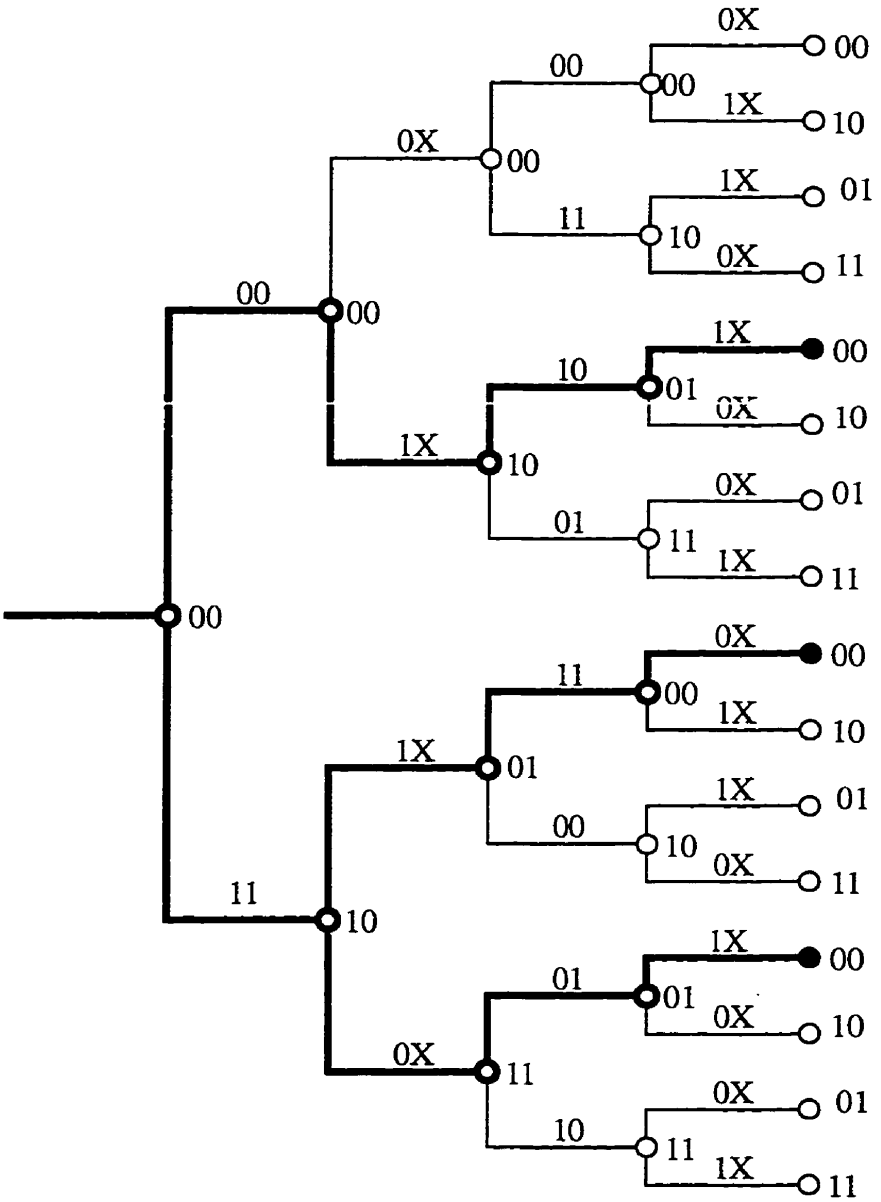


Figure 5.7: Représentation de l'arbre d'un code perforé $R = 2/3$

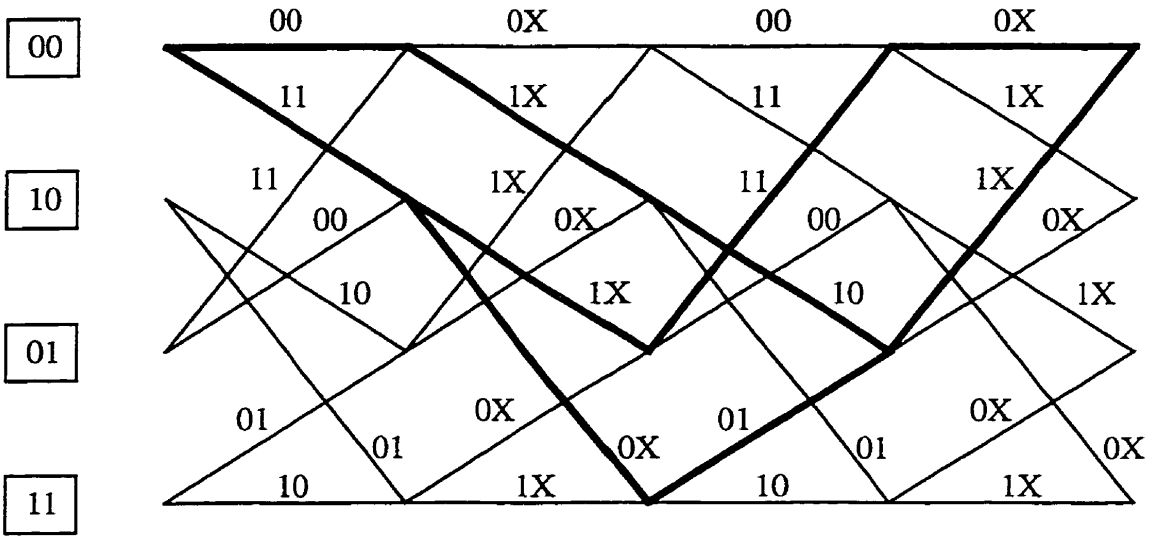


Figure 5.8: Représentation du treillis d'un code perforé $R = 2/3$

Pour bien garder la caractéristique "systématique" des codes perforés, les éléments de la première ligne du patron de perforation que l'on choisit doivent être uniquement constitués de "1". Par exemple, pour un code origine de taux de codage $R = 1/2$, on voudrait obtenir les code perforés de taux de codage $R = \{2/3, 3/4, 4/5, 5/6, 6/7, 7/8\}$. Dans ce cas, on choisit les patron de perforation de la sorte:

$$P = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$P = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Une propriété des codes perforés veut que, pour un même taux de codage, lorsqu'on change la position du "1" de la deuxième ligne du patron de perforation, le

spectre reste inchangé. Par exemple, pour $P = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ ou $P = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ les spectres du code perforé du taux $R = 2/3$ sont identiques. Le tableau 5.1 illustre les spectres de ces deux codes perforés.

Tableau 5.1: Comparaison de codes perforés avec les patrons de perforation différents

$P=\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$	d	A_d	C_d	$P=\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$	d	A_d	C_d
	3	1	3		3	1	3
	4	4	10		4	4	10
	5	14	44		5	14	44
	6	40	154		6	40	154
	7	115	521		7	115	521
	8	331	1724		8	331	1724
	9	953	5609		9	953	5609
	10	2744	18008		10	2744	18008
	11	7901	57201		11	7901	57201
	12	22750	180106		12	22750	180106
	13	65506	562944		13	65506	562944
	14	188617	1748632		14	188617	1748632
	15	543101	5402681		15	543101	5402681
	16	1563797	16615138		16	1563797	16615138
	17	4502774	50889898		17	4502774	50889898
	18	12965221	155309470		18	12965221	155309470
	19	37331866	472470604		19	37331866	472470604

5.4 PERFORMANCE DES CODES CONVOLUTIONNELS RÉCURSIFS SYSTÉMATIQUES PERFORÉS

Comme dans le cas des codes non perforés, la performance des codes convolutionnels récursifs systématiques perforés est évaluée à partir de leurs spectres. Au chapitre 3, nous avons déjà présenté la définition du spectre, les relations entre le spectre et le calcul de la probabilité d'erreur, et les méthodes permettant de déterminer le spectre. Dans cette section, nous calculons les performances des codes convolutionnels récursifs systématiques perforés.

Il est aussi possible d'évaluer les performances d'un code convolutionnel récursif systématique perforé à partir de son diagramme d'état. Reprenons le diagramme d'état du code perforé que l'on utilise comme exemple (figure 5.6) en remplaçant les transitions par B et D afin d'obtenir le diagramme d'état de la figure 5.9.

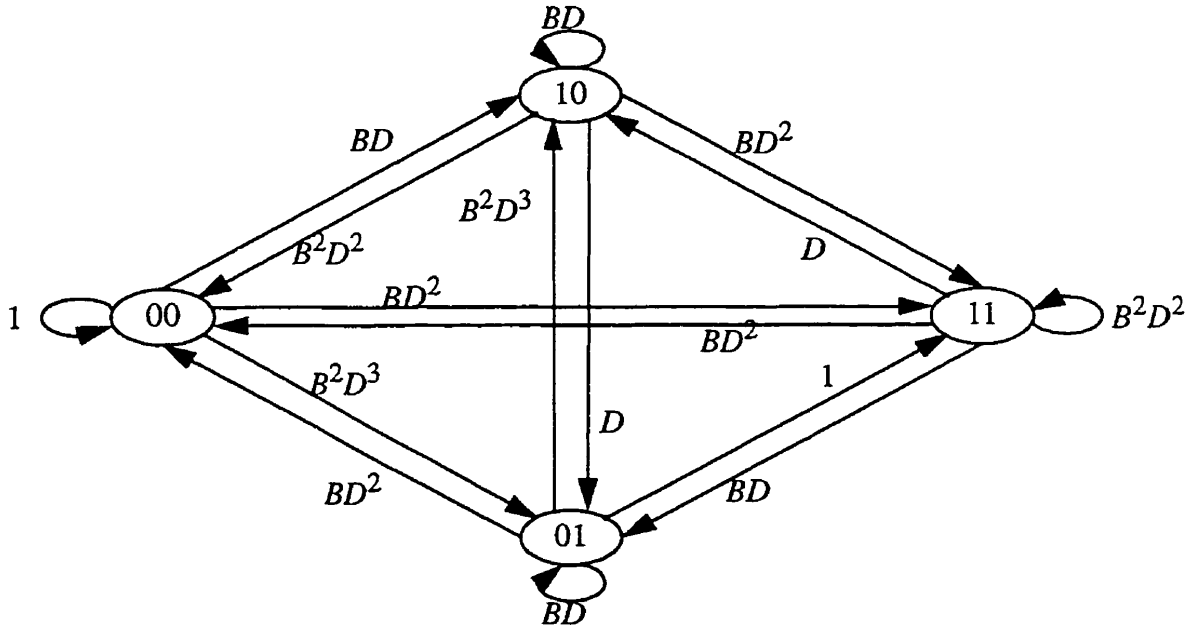


Figure 5.9: Diagramme d'état d'un code perforé $R = 2/3$ en B et D

En utilisant la même méthode du chapitre 3, la fonction de transfert en série de puissance D est obtenue de la façon suivante:

$$T(D,B) = D^3B^3 + D^4(3B^2 + B^4) + D^5(13B^3 + B^5) + \ldots$$

$$\frac{\partial T(D,B)}{\partial B} = 3D^3B^2 + D^4(6B + 4B^3) + D^5(39B^2 + 5B^4) + \ldots$$

Le développement de cette fonction de transfert sera démonstrée à Annexe I.

On peut donc obtenir les termes A_j et C_j du spectre des poids à partir de $T(D, B)$

$$T(D,B)\big|_{B=1} = \sum_{j=d_{free}}^{\infty} A_j D_j = D^3 + 4D^4 + 14D^5 + \ldots$$

$$\left.\frac{\partial T(D,B)}{\partial B}\right|_{B=1} = \sum_{j=d_{free}}^{\infty} C_j D_j = 3D^3 + 10D^4 + 44D^5 + \ldots$$

Il est possible d'évaluer le spectre de codes perforés en déterminant la fonction de transfert à partir du diagramme d'état. Mais le spectre peut aussi être évalué par ordinateur en utilisant la technique d'exploration de l'arbres. Cette méthode est beaucoup plus rapide et efficace. On a modifié l'algorithme "bidirectionnel" décrit dans [21] pour les codes perforés. Ce logiciel modifié permet en même temps de trouver les spectres des codes convolutionnels non récursifs non systématiques perforés et ceux des codes convolutionnels récursifs systématiques perforés.

Les spectres des codes perforés de $R = 2/3$ avec le patron de perforation P_1 qui est définit dans la section 5.1 sont présentés dans le tableau 5.2.

Tableau 5.2: Spectre des codes perforés: R= 2/3, $P = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$, $3 \leq K \leq 9$

codes origines		codes perforés R= 2/3	
K	G	d_f	(a_d) $[c_d]$
3	1, 5/7	3	(1, 4, 14, 40, 115, 331, 953, 2744, 7901, 22750, 65506, 188617, 543101, 1563797, 4502774) [3, 10, 44, 154, 521, 1724, 5609, 18008, 57201, 180106, 562944, 1748632, 5402681, 16615138, 50889898]
4	1, 17/15	4	(3, 11, 35, 114, 378, 1253, 4147, 13725, 45428, 150362, 497681, 1647267, 5452266, 18046379, 59731456) [10, 33, 146, 538, 2046, 7595, 27914, 101509, 366222, 1312170, 4674258, 16567175, 58462900, 205511847, 719960394]
5	1, 35/23	4	(1, 0, 27, 0, 345, 0, 4515, 0, 59058, 0, 772627, 0, 10107642, 0, 132230266) [3, 0, 106, 0, 1841, 0, 30027, 0, 471718, 0, 7201171, 0, 107686039, 0, 1585096699]
6	1, 53/75	6	(19, 0, 220, 0, 3089, 0, 42725, 0, 586592, 0, 8085210, 0, 111315783, 0, 1532925641) [82, 0, 1260, 0, 21530, 0, 354931, 0, 5643947, 0, 88444551, 0, 1364368734, 0, 20808088020]
7	1, 171/133	6	(1, 16, 48, 158, 642, 2435, 9174, 34701, 131533, 499312, 1891754, 7165914, 27160547, 102939934, 390103650) [3, 76, 269, 960, 4290, 18034, 74197, 303431, 1237276, 5030802, 20324463, 81763094, 328002734, 1311800821, 5231393084]
8	1, 371/247	6	(3, 0, 71, 0, 1032, 0, 14469, 0, 210819, 0, 3041149, 0, 43985963, 0, 635562375) [13, 0, 392, 0, 7081, 0, 118852, 0, 2008342, 0, 33041542, 0, 536313579, 0, 8595455776]
9	1, 753/561	7	(3, 9, 50, 190, 641, 2507, 9745, 37120, 142220, 544948, 2086538, 7988101, 30588852, 117140948, 448544740) [15, 50, 292, 1246, 4703, 20167, 84481, 346376, 1422291, 5809959, 23643588, 95854870, 387422765, 1561740153, 6279014364]

En changeant le patron de perforation P, on a les différents codes perforés de taux de codage allant de 2/3 à 7/8. Le tableau 5.2 montre uniquement des codes perforés de taux de codage R=2/3. Les autres codes sont présentés à Annexe II.

Pour déterminer la borne supérieure sur la probabilité d'erreur par bit, il suffit d'appliquer l'équation (3-11) en utilisant les termes C_j trouvés au tableau 5.1. La figure 5.10 fournit les performances des codes perforés du tableau 5.1. Comme dans le cas des codes non perforés, lorsque la longueur de contrainte K est plus grande, ces performances sont meilleures. La figure 5.11 compare plusieurs codes perforés qui sont engendrés par le même code origine. On constate que lorsque le taux de codage R diminue, la probabilité d'erreur diminue aussi.

Les codes perforés augmentent le taux de codage de code origine, mais ils possèdent la structure sous-jacente des codes de faible taux dont ils sont dérivés. Ils peuvent être utilisés partout où un taux de codage élevé est requis, sans augmenter la complexité du processus de décodage des codes convolutionnels de faible taux de codage.

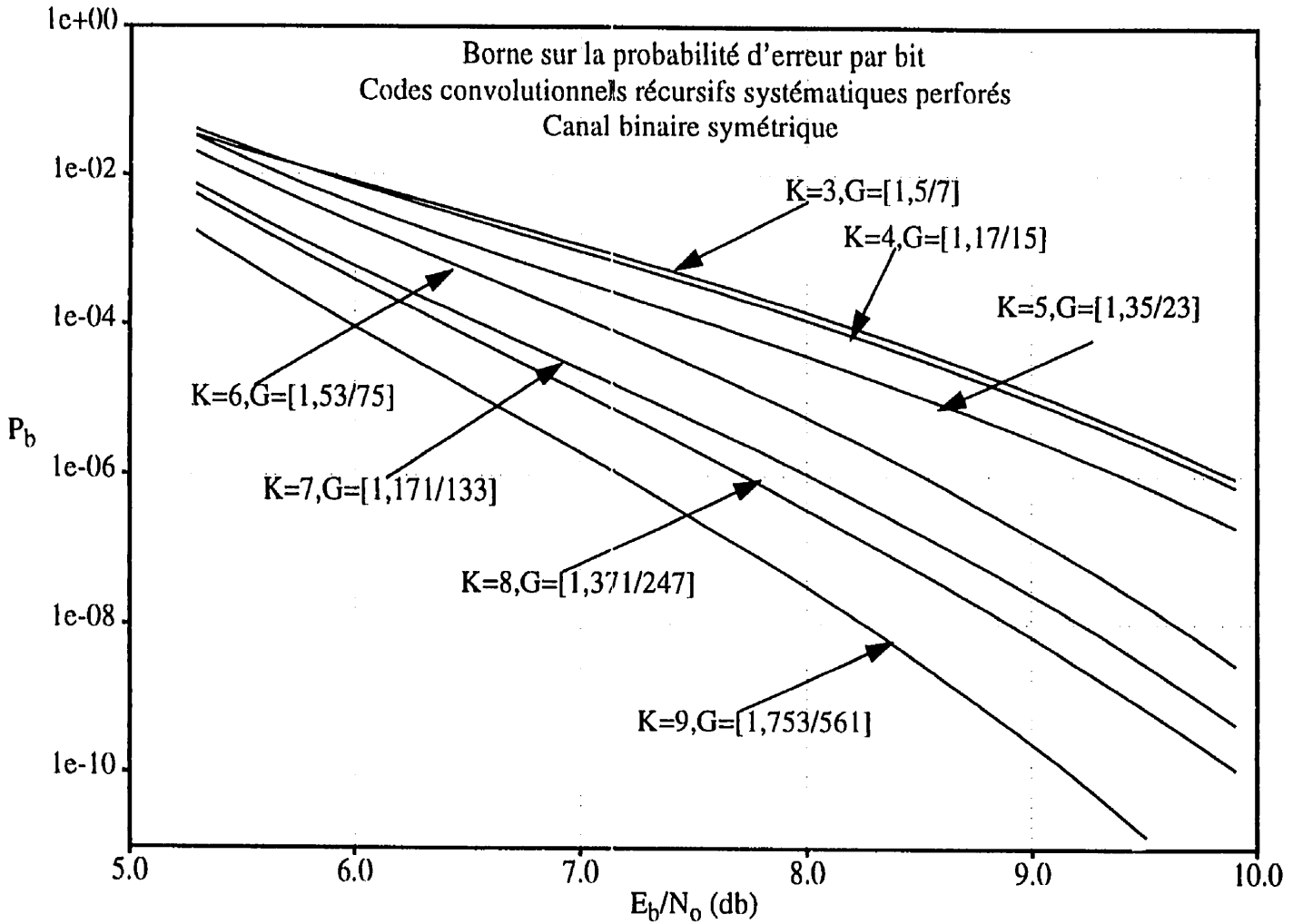


Figure 5.10: Codes perforés avec le patron de perforation $P = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$

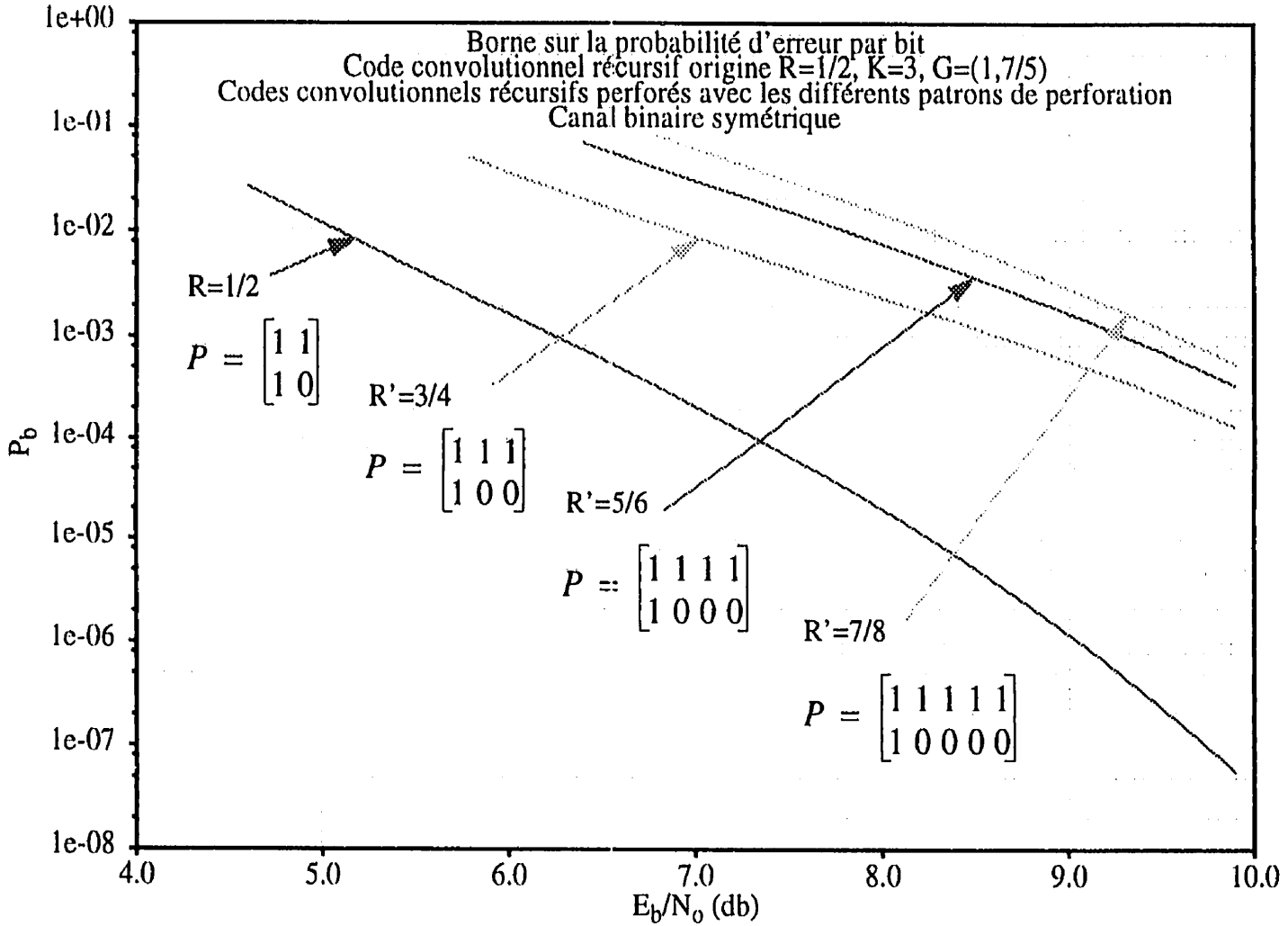


Figure 5.11: Codes perforés avec les différents patron de perforation

CHAPITRE 6

CODAGE À CONCATÉNATION PARALLÈLE DE CODES CONVOLUTIONNELS

Une nouvelle stratégie de correction d'erreur a été récemment proposée par Berrou *et al.* [15]. Depuis, plusieurs recherches ont été menées dans ce domaine, en particulier sur l'étude des distributions des poids pour des codes utilisant une concaténation parallèle de codes convolutionnels [8] [28].

Ce système de codage correcteur d'erreur consiste à utiliser une concaténation parallèle de deux ou plusieurs codeurs convolutionnels récursifs systématiques. Les performances d'erreur d'un tel système exprimées en termes de probabilité d'erreur par bit sont parmi les plus élevées, se rapprochant le plus de la limite de Shannon [15]. L'évaluation des performances d'erreur a déjà été obtenue par Benedetto et Montorsi [7] pour une concaténation parallèle de codes en blocs et pour celle de codes convolutionnels récursifs systématiques. En utilisant la méthode proposée par Podemski, Holubowicz, Berrou, et Glavieux [8] pour trouver le spectre d'un code, on obtient une borne supérieure sur la probabilité d'erreur par bit pour un canal à bruit blanc gaussien additif. Pour le calcul de cette borne, on suppose que le décodage utilise un décodeur à maximum de vraisemblance.

Dans ce chapitre, on présente la structure du codage CPCC (concaténation parallèle de codes convolutionnels). On y présente aussi le spectre, la représentation en

hyper-treillis et les performances d'erreur des codes CPCC. Par la suite, on aborde la perforation des codes CPCC.

6.1 PRINCIPE DU CODAGE CPCC

Le schéma de principe de codage CPCC est illustré à la figure 6.1. Ce codeur consiste à utiliser deux codeurs convolutionnels récursifs systématiques séparés par un entrelaceur.

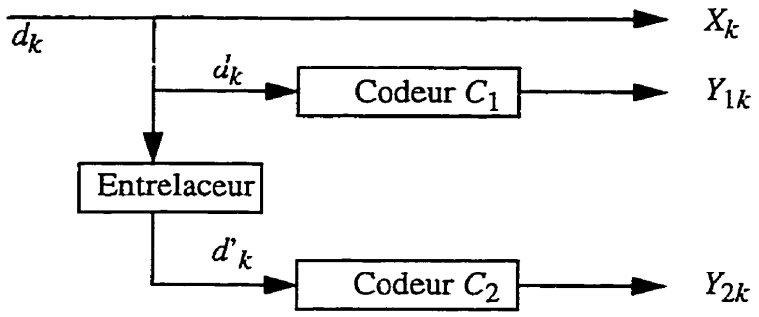


Figure 6.1: Codeur CPCC

De façon générale, les deux codeurs constituants C_1 et C_2 sont identiques. La présence d'un entrelaceur de dimension N fait en sorte que les mêmes bits d'information sont appliqués aux deux codeurs C_1 et C_2, mais dans un ordre différent. Ainsi, à un instant k donné, le premier codeur C_1 reçoit le bit d_k pour générer la paire (X_k, Y_{1k}), tandis que le deuxième codeur C_2 reçoit le bit d'_k pour produire la paire (X_k, Y_{2k}). Les bits d'information sont regroupés par blocs de longueur N, égale à celle de l'entrelaceur.

Notons que le nombre de codeurs constituants formant le codeur CPCC peut être

supérieur à deux. De plus, les codes utilisés peuvent être différents. Cependant, tout au long de ce mémoire, on considère que les deux codeurs utilisés sont identiques. Puisque la probabilité d'erreur d'un code convolutionnel récursif systématique est plus faible que celle d'un code convolutionnel non systématique à faible rapport signal à bruit. Et les codes convolutionnels non systématiques ne sont pas attrayants, car ils ont la même distance libre que les codes convolutionnels récursifs systématiques. Donc, on choisit les codes convolutionnels récursifs systématiques pour construire les codes CPCC de sorte que globalement les codes CPCC sont systématiques.

Soient les deux codeurs constituants de taux de codage $R_1 = b/v_1$, $R_2 = b/v_2$. Le taux de codage global du codeur CPCC est alors

$$R = \frac{b}{v_1 + v_2 - b} = \frac{1}{1/R_1 + 1/R_2 - 1} \text{ bits/symboles.} \qquad (6\text{-}1)$$

En réarrangeant les termes de (6-1), on obtient l'équation suivante:

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2} - 1 \qquad (6\text{-}2)$$

En général, pour un nombre L de codeurs, chacun de taux de codage R_i, $i = 1, 2, \ldots, L$, on obtient:

$$\frac{1}{R} = \sum_{i=1}^{L} \frac{1}{R_i} - 1 \qquad (6\text{-}3)$$

Donc, pour deux codeurs constituants identiques de taux de codage $R_1 = R_2 = 1/v$, le taux de codage total devient $R = 1/(2v-1)$. Par exemple, dans le cas du codeur de la figure 6.1, les taux de codage R_1 et R_2 sont égaux à 1/2, de sorte que le taux de codage total est $R = 1/3$.

Le calcul d'une borne supérieure sur la probabilité d'erreur par bit, permet d'évaluer la performance d'un code CPCC. Pour ce faire, il faut tout d'abord déterminer l'hyper-treillis correspondant au code CPCC.

6.1.1 Représentation en treillis

Considérons un codeur CPCC formé par une entrelaceur bloc de longueur N et de deux codeurs convolutionnels récursifs systématiques C_1 et C_2 identiques de taux de codage $R_1 = R_2 = 1/2$, de longueur de contrainte $K = 3$ et dont la matrice de générateurs $G = [1, 7/5]$. Ce codeur CPCC est illustré à la figure 6.2.

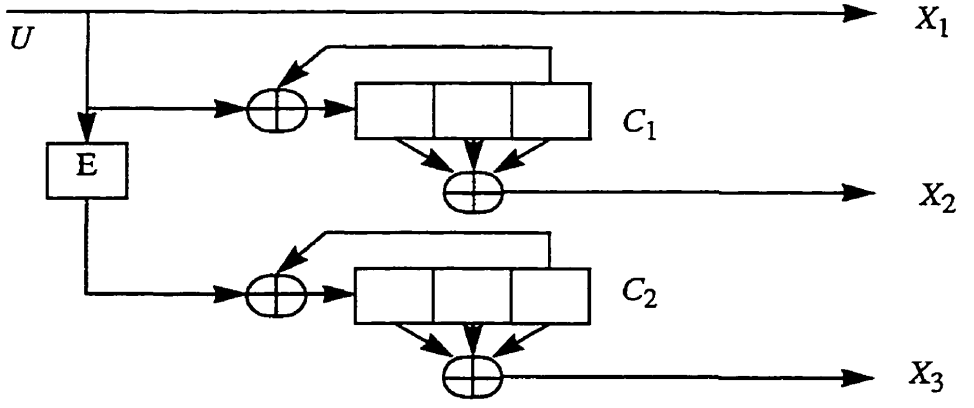


Figure 6.2: Codage CPCC
utilisant deux codeurs convolutionnels identiques R_1=R_2=1/2, K=3, G=[1, 7/5]

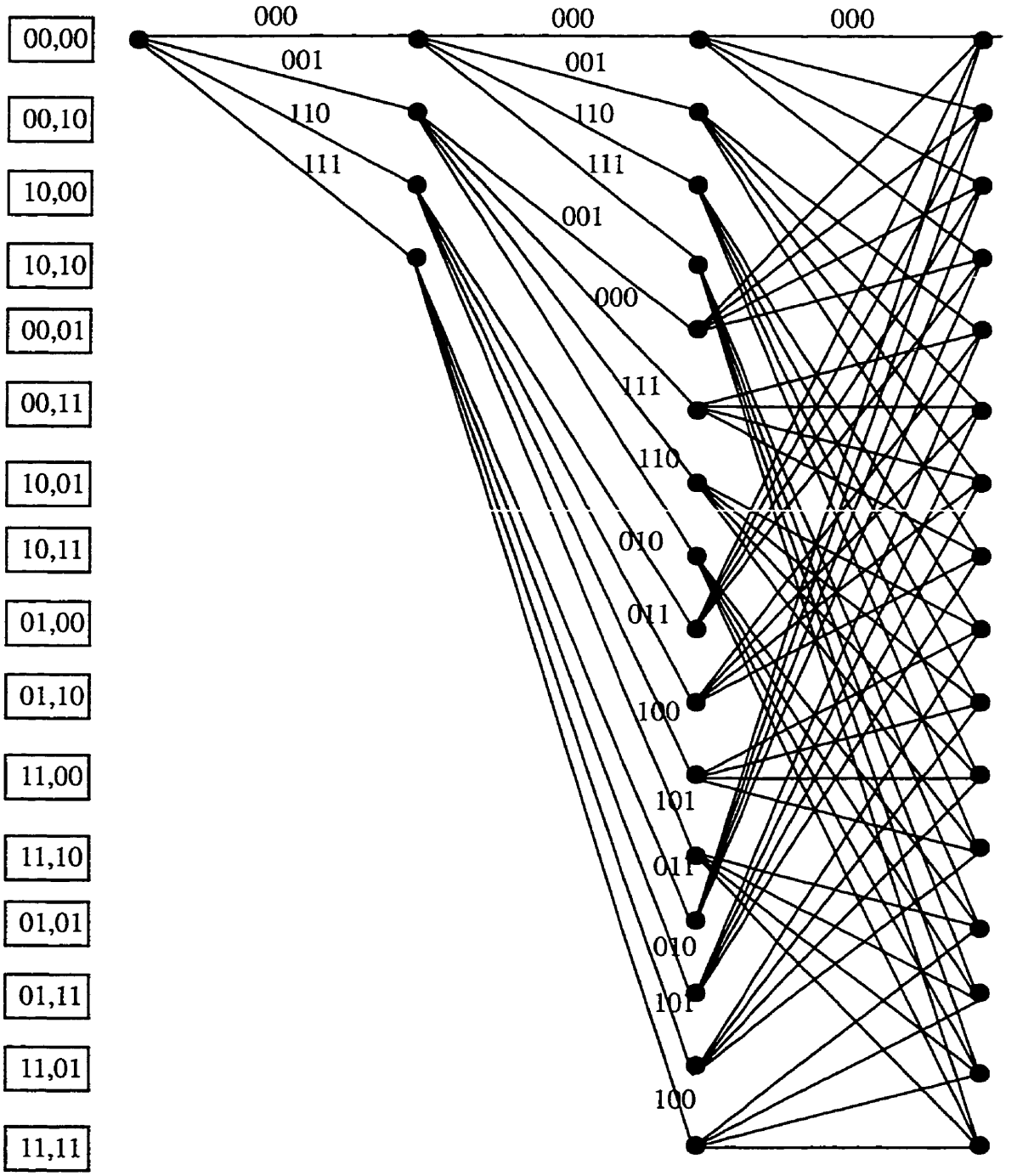


Figure 6.3: hyper-treillis d'un codeur CPCC de la figure 6.2

Pour examiner complètement le comportement de ce codeur CPCC, nous considérons un hyper-treillis de 4*4=16 états, tel que représenté à la figure 6.3. Dans ce treillis, le nombre de noeuds à chaque niveau est égal au nombre d'états. Les états sont représentés sur un axe vertical. Chaque état est une combinaison de deux états des codeurs constituants, séparés par une virgule. Les symboles codés à la sortie de l'encodeur sont indiqués à côté de chacune des branches. La longueur du bloc de données, donc de l'entrelaceur étant N, le treillis est tronqué à la $N^{\text{ième}}$ branche.

6.1.2 Représentation en arbre

Dans cette représentation, on illustre les transitions entre les états du code par un chemin partant de l'état zéro se trouvant à l'origine de l'arbre, vers les états ultérieurs. Les paires d'états sont reliées par une branche de l'arbre. Chaque noeud de l'arbre illustre alors un état du codeur CPCC. Les symboles de chaque état sont indiqués à la gauche du noeud de cet état. L'état de premier codeur constituant est écrit au-dessus de la branche, et l'état de deuxième codeur est indiqué au-dessous de la branche. Les trois symboles codés obtenus à la sortie de l'encodeur sont indiqués sur chacune des branches. La figure 6.4 montre une partie de l'arbre du code CPCC de la figure 6.2.

La représentation en arbre est très redondante par rapport à celle en treillis. Mais la plupart des algorithmes permettant de déterminer le spectre d'un code utilise l'exploration dans un arbre [20] et [21]. Ainsi, pour un code CPCC, on opte aussi pour la représentation en arbre afin de réaliser notre recherche.

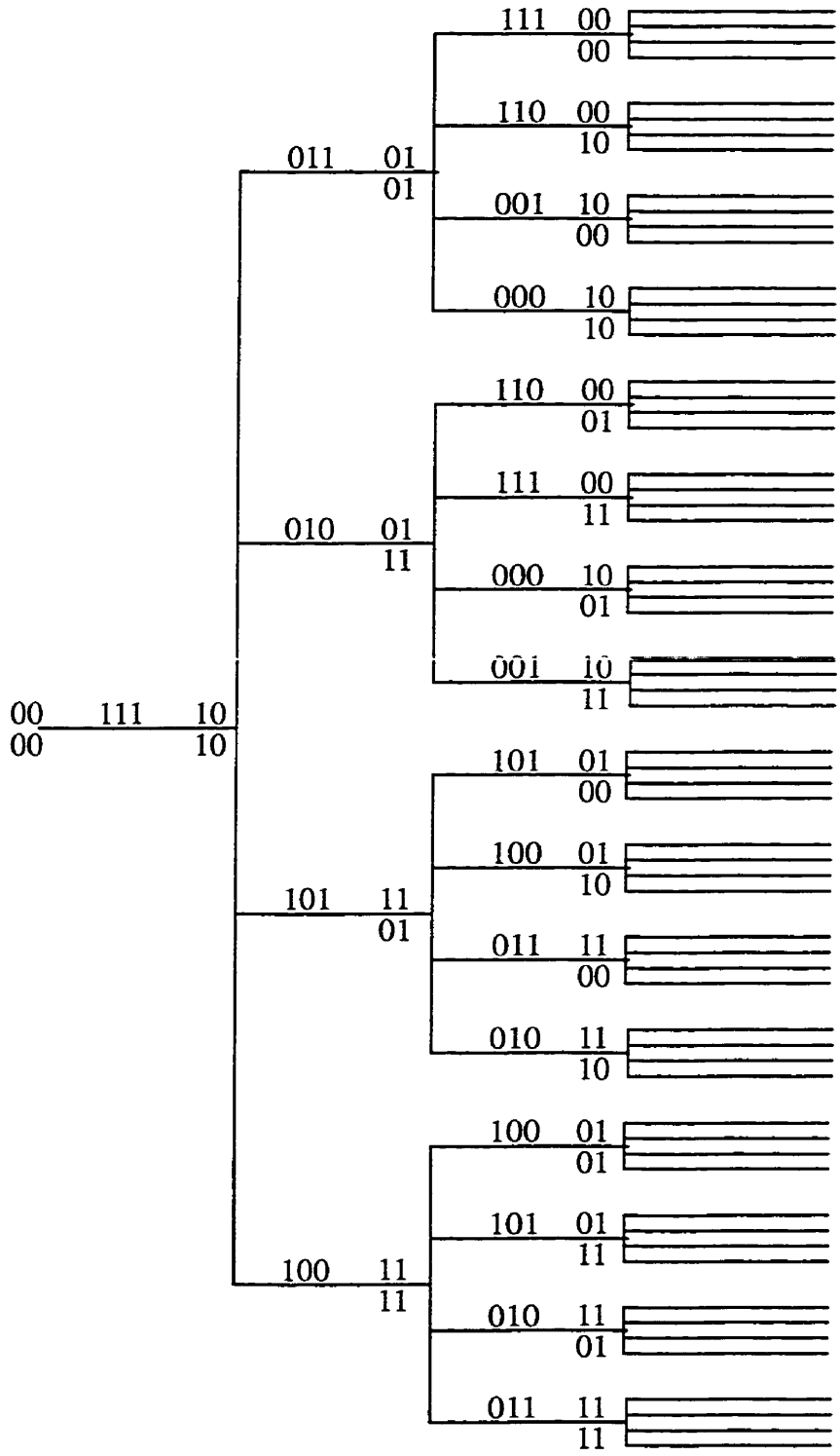


Figure 6.4: Représentation en arbre d'un codeur CPCC de la figure 6.2

6.1.3 Entrelaceur

La présence de l'entrelacement dans les codes CPCC permet de construire des codes en apparence longs sans toutefois utiliser des codeurs ayant une grande longueur de contrainte. A la réception, des entrelaceurs sont utilisés au décodage dans la but de fournir à chaque itération des séquences d'information en apparence différentes. En d'autres termes, l'entrelacement joue un rôle de décorrélateur entre les symboles utilisés dans la procédure de décodage. Cette décorrélation ou indépendance entre les symboles est nécessaire pour améliorer la convergence de la procédure de décodage.

L'entrelaceur permet de modifier l'ordre des bits d'information d'une séquence. Ainsi, pour un code CPCC, le codeur constituant C_1 encode directement la séquence d'information alors que le codeur C_2 encode un version entrelacée de la même séquence d'information. La méthode la plus simple d'entrelacer est d'utiliser un entrelaceur bloc. Il s'agit d'écrire les bits d'information ligne par ligne dans une matrice de dimension IxJ (idéalement carrée) et de les lire colonne par colonne. Dans ce cas, on dénote par $N = I$xJ, la longueur de l'entrelaceur. La figure 6.5 illustre un exemple d'entrelaceur en bloc ayant une longueur $N = 16$.

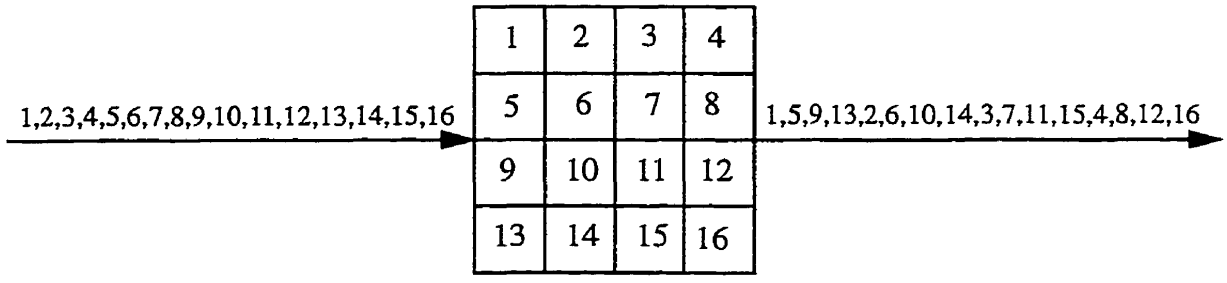


Figure 6.5: Exemple de fonctionnement d'un entrelaceur en bloc, N = 16

Un autre type d'entrelaceur utilise un entrelacement aléatoire. Un entrelaceur aléatoire de longueur N fonctionne de la manière suivante: pour chaque séquence (ou bloc) de N bits d'information, une série de nombres aléatoires compris entre 1 et N est générée. Cette série de nombres correspond à la nouvelle position des bits dans le bloc entrelacé [29]. Par exemple, soit une séquence de 16 bits d'information en l'ordre de 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16. Après entrelacement aléatoire, la séquence d'information suit l'ordre 1, 4, 6, 2, 9, 16, 13, 11, 15, 7, 3, 8, 5, 12, 10, 14, ou en l'ordre 1, 8, 5, 10, 15, 13, 7, 2, 14, 6, 16, 4, 11, 3, 12, 9.

6.2 ANALYSE DES PERFORMANCES DES CODES CPCC

L'analyse des performances d'erreur a déjà été obtenue par Benedetto et Montorsi [7] pour une concaténation parallèle de codes blocs ainsi que pour celle de codes convolutionnels. En utilisant la méthode proposée par Podemski, Holubowicz, Berrou, Glavieux [8], pour trouver le spectre d'un code, une borne supérieure sur la probabilité d'erreur par bit est obtenue en considérant un canal à bruit additif blanc et gaussien et un décodeur à maximum de vraisemblance.

L'analyse théorique des performances d'erreur des codes CPCC peut se faire en définissant un entrelaceur comme étant un composant probabiliste capable de produire, pour une séquence d'entrée de longueur N (égale à celle de l'entrelaceur) et de poids w, l'ensemble de toutes les permutations λ des bits de valeur "1" appartenant à cette séquence, ce qui s'écrit:

$$\lambda = \binom{N}{w}$$

En supposant que la variable λ a une densité de probabilité uniforme, la probabilité d'apparition de l'une de ces permutations est données par:

$$P_\lambda = 1 / \binom{N}{w}$$

Si on admet l'hypothèse que le bruit sur le canal est blanc, gaussien et additif et que le décodage est basé sur le maximum de vraisemblance, la borne supérieure sur la probabilité d'erreur moyenne par bit pour un code CPCC est donnée par [7]:

$$P_b \leq \sum_{d=d_{free}}^{\infty} \sum_{w=1}^{N} (n_w \cdot w) / \binom{N}{w} \cdot P_d \tag{6-4}$$

où n_w est le nombre de mots de code de poids w et où P_d est la probabilité d'erreur d'une paire de mots de code de distance égale à d:

$$P_d = Q\left(\sqrt{2 \cdot d \cdot R \cdot \frac{E_b}{N_0}}\right) \tag{6-5}$$

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

L'équation (6-4) montre la relation qui existe entre les performances d'erreur et le spectre d'un code. L'analyse des performances se réduit alors à la détermination de ce spectre. Une méthode permettant de déterminer le spectre d'un code consiste à effectuer une exploration dans un arbre représentant le code (voir Figure 6.6).

Dans cet arbre, chaque chemin débute par un bit d'information "1" et termine à un noeud d'état zéro pouvant se terminer au maximum à la $N^{\text{ième}}$ profondeur du treillis. L'état zéro du codeur CPCC est différent de celui du codeur convolutionnel classique, car le codeur CPCC consiste en deux codeurs convolutionnels. Selon [8], au lieu d'appliquer une séquence binaire arbitraire pour le code CPCC, on prend une séquence sélectionnée qui force à remettre le codeur convolutionnel C_1 à l'état zéro. Cette séquence s'appelle séquence de retour-à-zéro. Les auteurs de [8] ont prouvé que pour n'importe quel code convolutionnel récursif systématique, il existe une seule séquence de retour-à-zéro. Par exemple, pour le code convolutionnel récursif systématique (1, 7/5), la séquence de retour-à-zéro est $\underline{x}$ = [101], pour le code convolutionnel récursif systématique (1, 5/7), c'est $\underline{x}$ = [111]. Donc, on considère que l'état zéro d'un codeur CPCC correspond simplement à l'état zéro du premier codeur convolutionnel C_1.

Pour chercher le spectre dans un arbre, il s'agit de calculer le poids et le nombre de bits d'information "1" cumulés sur le chemin reliant le noeud d'état zéro situé dans le sous-section de l'arbre du code débutant par un bit d'information "1" et le noeud inclus l'état zéro du codeur convolutionnel C_1. Seuls les noeuds d'état zéro qui sont avant ou à la $N^{\text{ième}}$ étape doivent être considérés, à l'exception bien sûr du noeud origine.

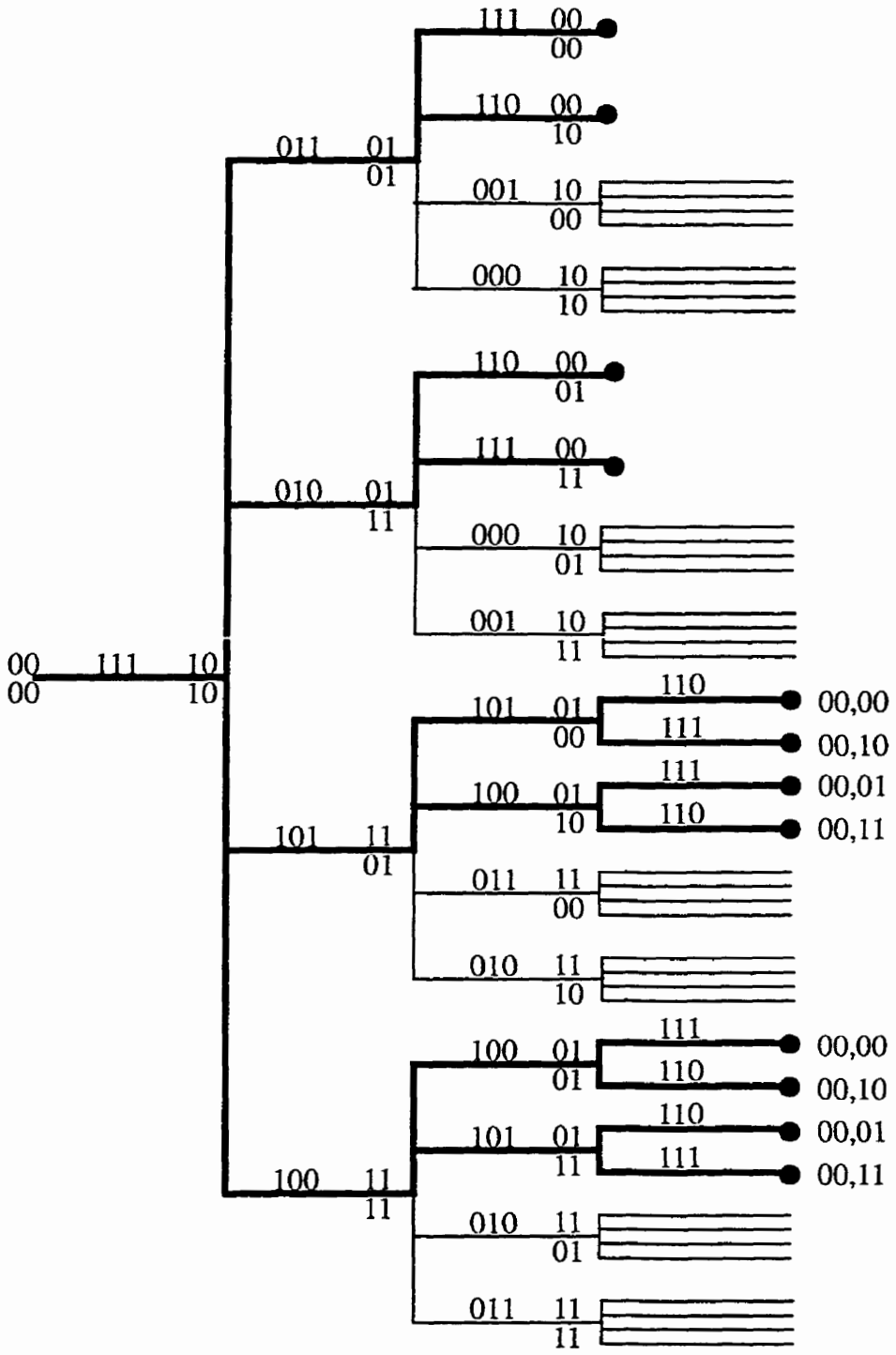


Figure 6.6: Représentation en arbre du codeur CPCC de la figure 6.2, chemins en trait gras associés à la longueur de l'entrelaceur $N = 4$

Dans le cas d'un code CPCC, la représentation du spectre est modifiée afin de faciliter l'analyse des performances d'erreur du code. Le tableau 6.1 donne le spectre du code de la figure 6.6.

Tableau 6.1: Spectre du code CPCC de $R = 1/3$, $N = 4$
avec les codes origines de $R_1 = R_2 = 1/2$, $K = 3$, $G = (1, 7/5)$

d	n	w
6	1	2
7	2	2
7	1	4
8	1	2
8	3	4
9	3	4
10	1	4

Le spectre d'un code convolutionnel représente l'ensemble des mots de code non-nuls dénombrés en fonction de leurs distances de Hamming par rapport à un mot de code nul. Le spectre du code CPCC est défini en utilisant un tableau de trois colonnes:

d: le poids des mots de code,

n: le nombre de mots de code de poids d,

w: le nombre de bits d'information "1" correspondant au mot de code de poids d.

Chaque triplet $\{d, n, w\}$ compose une raie du spectre. Seules les raies de poids égal ou supérieur à d_{free} du code origine influencent le calcul de la probabilité d'erreur du code.

Parmi les raies du spectre d'un code CPCC, quelques unes sont différentes par rapport à la longueur de l'entrelaceur N. Pour le code CPCC ayant un taux de codage total $R = 1/3$ utilisant des codes constituants avec $R_1 = R_2 = 1/2$, $K = 3$, $G = (1, 7/5)$, quelques résultats obtenus sont fournis aux tableaux 6.2 et 6.3.

Tableau 6.2: Spectre du code CPCC de $R = 1/3$, $N = 16$
avec les codes origines de $R_1 = R_2 = 1/2$, $K = 3$, $G = (1, 7/5)$

d	n	w	d	n	w
6	1	2	12	129	2
7	3	2	12	617	4
7	1	4	12	319	6
8	6	2	12	16	8
8	6	4	* 13	276	2
9	13	2	13	1642	4
9	23	4	13	1271	6
9	1	6	13	138	8
10	28	2	13	1	10
10	75	4	* 14	578	2
10	11	6	* 14	4217	4
11	60	2	* 14	4520	6
11	222	4	14	863	8
11	68	6	14	21	10
11	1	8			

Tableau 6.3: Spectre du code CPCC de $R = 1/3$, $N = 32$
avec les codes origines de $R_1 = R_2 = 1/2$, $K = 3$, $G = (1, 7/5)$

d	n	w	d	n	w
6	1	2	12	129	2
7	3	2	12	617	4
7	1	4	12	319	6
8	6	2	12	16	8
8	6	4	* 13	277	2
9	13	2	13	1642	4
9	23	4	13	1271	6
9	1	6	13	138	8
10	28	2	13	1	10
10	75	4	* 14	595	2
10	11	6	* 14	4232	4
11	60	2	* 14	4541	6
11	222	4	14	863	8
11	68	6	14	21	10
11	1	8			

On remarque, à partir des tableaux 6.2 et 6.3, que les raies du spectre d'un code CPCC varient peu en fonction de la taille de l'entrelaceur.

En effet dans les tableaux 6.2 et 6.3, nous constatons que les raies indiquées par le symbole * ne sont pas égales. Les valeurs de $N = 32$ sont plus grandes que celles de $N = 16$. Cela signifie que lorsque N augmente, le nombre de chemins qui débutent par un bit

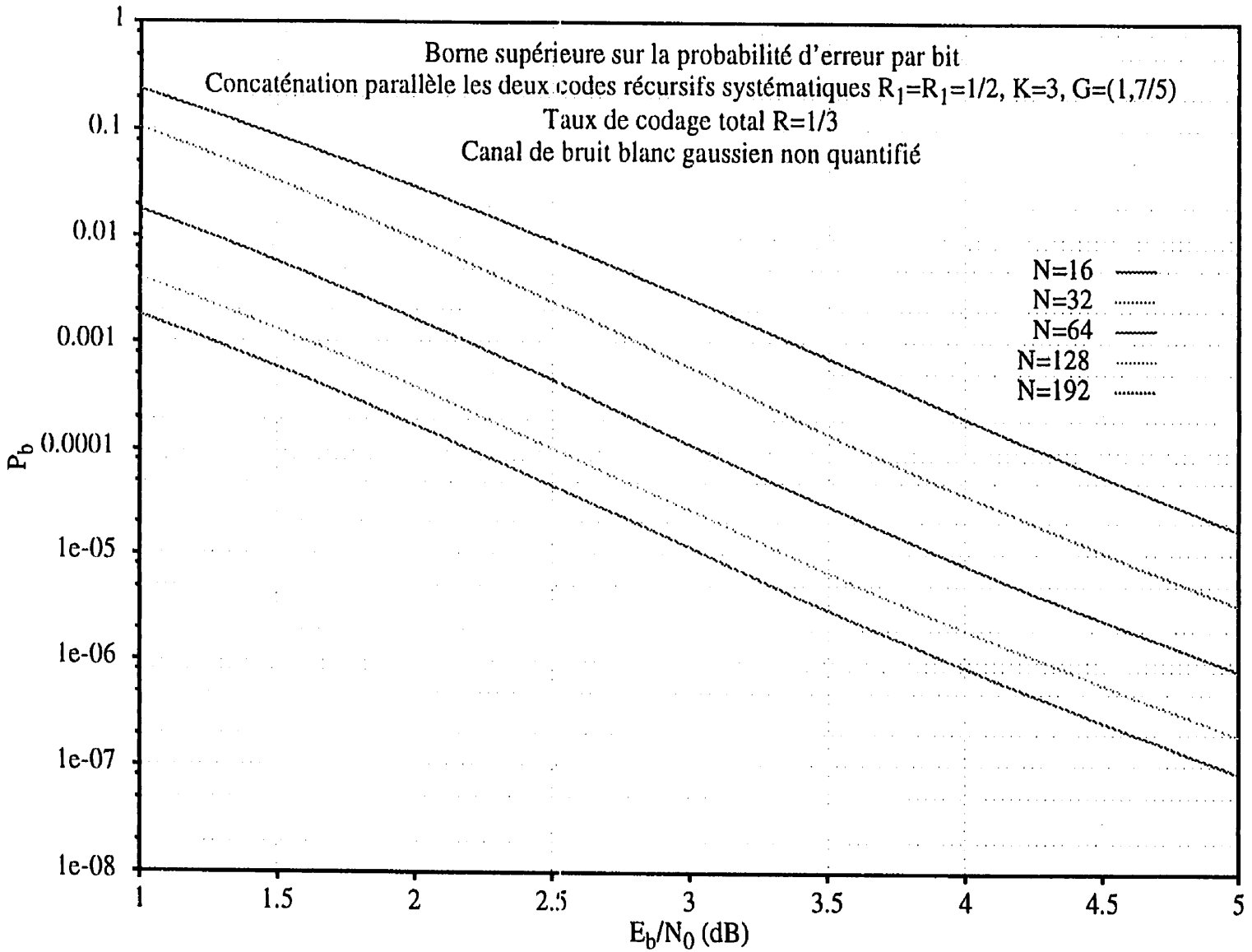


Figure 6.7: Probabilités d'erreur du code CPCC de la figure 6.2, différentes valeurs de N

d'information "1" et qui terminent à un noeud d'état zéro augmente aussi. En considérant l'équation (6-3), on constate que si longueur de l'entrelaceur N augmente, la probabilité d'erreur diminue. La figure 6.7 illustre la comparaison des probabilités d'erreur du code de la figure 6.2 pour différentes valeurs de N = 16, 32, 64, 128, 192.

Jusqu'à présent nous avons étudié les performances d'erreur du code CPCC. On peut maintenant se demander, si les performances d'erreur des codes CPCC sont meilleures que celle du code constituant. Donc, voici quelques comparaisons intéressantes.

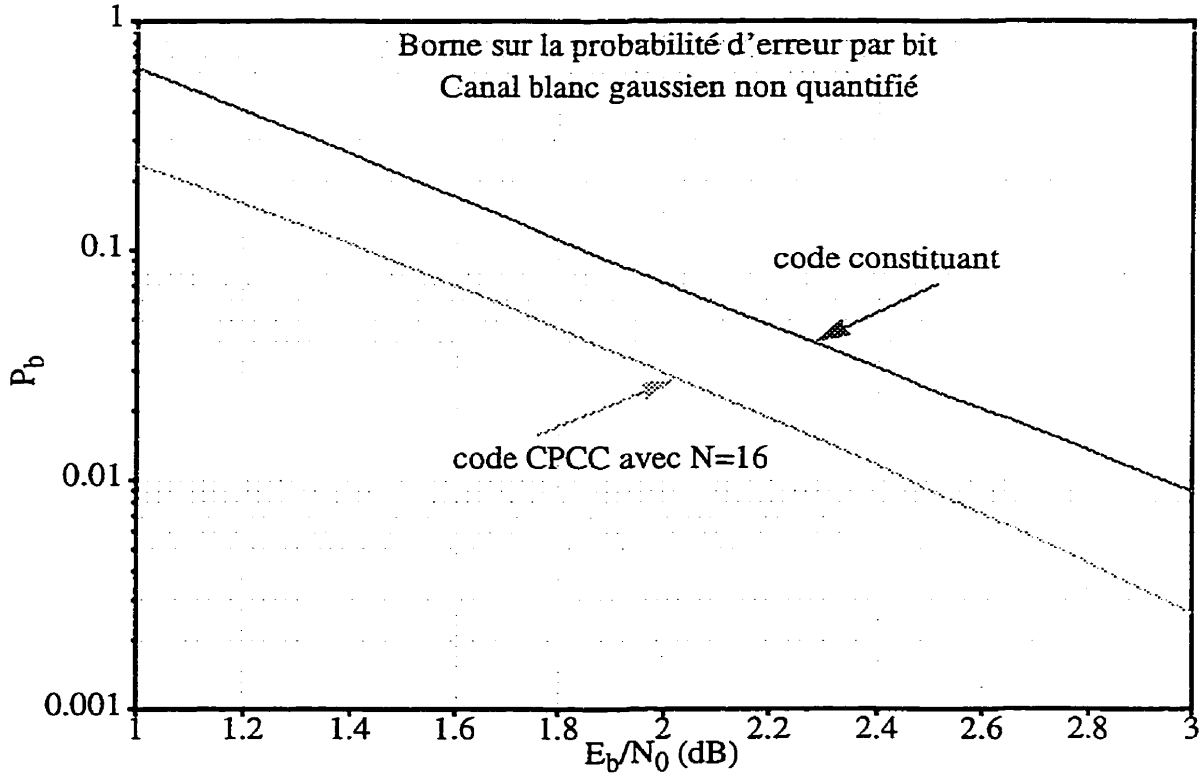


Figure 6.8: Comparaison du code origine et du code CPCC
avec codes récursifs systématiques R_1=R_2=1/2, K=3, G=(1,7/5)

La figure 6.8 montre les performances d'erreur du code constituant et du code CPCC engendré par ce code constituant. Nous constatons que même si la valeur de N est petite, le code CPCC donne de meilleurs performances d'erreur que le code constituant.

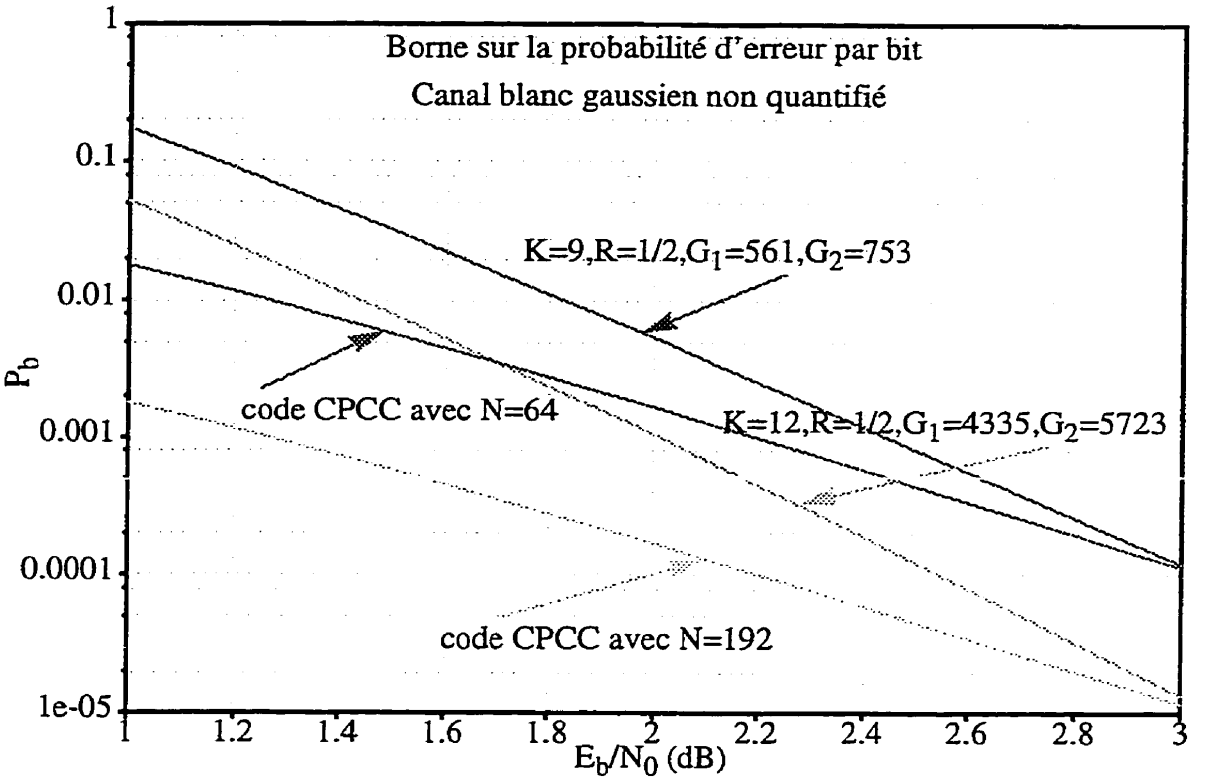


Figure 6.9: Comparaison des codes de K élevés
et du code CPCC avec codes récursifs systématiques R_1=R_2=1/2, K=3, G=(1,7/5)

La figure 6.9 illustre les performances des codes convolutionnels récursifs systématiques et des codes CPCC pour différentes valeurs de N. Nous rappelons que la probabilité d'erreur décroît exponentiellement avec la longueur de contrainte K du code. La comparaison entre un code ayant une longueur de contrainte K = 12, R = 1/2, G_1 = 4335, G_2 = 5723 et le code CPCC ayant R = 1/3, K = 3, G = (1, 7/5) avec N = 192 nous montre que le code CPCC offre de meilleurs performances et ce pour de faibles E_b/N_0 (de 1 à 3 dB).

Remarquons qu'à la figure 6.9, il y a un croisement entre la courbe de performance d'erreur du code ayant une longueur de contrainte $K = 12$, $R = 1/2$, $G_1 = 4335$, $G_2 = 5723$ et celle du code CPCC avec $R = 1/3$, $K = 3$, $G = (1, 7/5)$ avec $N = 64$. Ainsi, ces résultats montrent que les performances d'erreur du CPCC sont parfois inférieures aux performances d'erreur d'un code convolutionnel récursif systématique. En effet, lorsque le rapport signal à bruit E_b/N_0 est plus petit ou égal à 1.7 dB, il est clair que le CPCC avec $R = 1/3$, $K = 3$, $G = (1, 7/5)$ avec $N = 64$ donne de meilleures performances d'erreur en comparaison du code convolutionnel récursif systématique. D'autre part, pour E_b/N_0 supérieur à 1.7 dB, la technique utilisant un code convolutionnel récursif systématique de longueur de contrainte $K = 12$, un taux de codage $R = 1/2$ et deux générateurs spécifié par $G_1 = 4335$, $G_2 = 5723$ fournit des performances d'erreur supérieures à celles du code CPCC.

6.3 PERFORATION DES CODES CPCC

La technique de la perforation est utilisée pour obtenir des codes de taux de codage élevé et variable. Afin d'augmenter le taux de codage global R d'un code CPCC, initialement égal à 1/3, un perforateur est ajouté à la sortie du codeur CPCC. Le codeur ainsi obtenu est représenté à la figure 6.10.

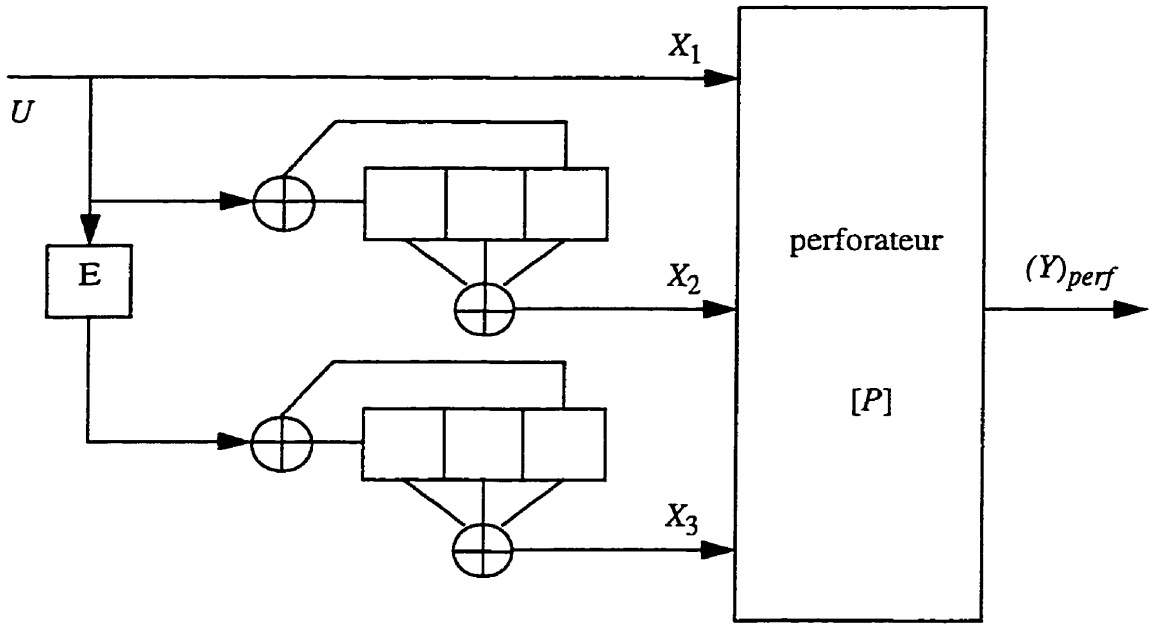


Figure 6.10: Schéma de principe de la perforation d'un codeur CPCC

La seule condition pour choisir la matrice de perforation P est que la première ligne de cette matrice soit formée uniquement de "1", afin que le code résultant demeure systématique. Il est à noter que cette exigence implique que le taux de codage global $R = b/\nu$ doit satisfaire à la condition $b \leq \nu - 2$. En effet, puisque tous les b bits d'information doivent être transmis et que chaque codeur doit transmettre au moins un symbole codé, alors ν est au moins égal à $b + 2$. Pour un code de taux de codage initial $R' = 1/\nu_0$ et de taux de codage global $R = b/\nu$, le nombre total de possibilités de matrices de perforation est égal à $(\nu_0 - 2)*b^2$. Par exemple, pour un taux de codage global $R = 4/6$, obtenu à partir du code CPCC de taux de codage initial $R = 1/3$, le patron de perforation pourrait être donné par la matrice suivante:

$$P = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Notons qu'il peut y avoir des patrons de perforation différents pouvant donner des codes perforés de même taux de codage, mais de performances d'erreur différentes. Cependant, le problème qui se pose est de déterminer parmi ces codes lesquels des patrons de perforation offrent les meilleures performances d'erreur. Les calculs des spectres et des performances de ces codes, nous donnent un moyen de déterminer les meilleurs patrons de perforation.

Pour faciliter notre étude, nous prenons le code CPCC de taux de codage global $R = 1/3$ résultant de la concaténation parallèle de deux codeurs convolutionnels récursifs systématiques de taux de codage $R_1 = R_2 = 1/2$, $K = 3$, $G = (1, 7/5)$ et utilisant un entrelaceur de longueur $N = 16$. Les taux de codage considérés sont $R = 2/4, 4/6, 6/8, 8/10$. Pour un taux de codage particulier, le spectre du code ainsi que la probabilité d'erreur par bit sont obtenus pour chaque patron de perforation possibles. Les performances d'erreur sont alors comparées entre eux, nous permettant ainsi de choisir la configuration de la matrice de perforation P qui minimise la probabilité d'erreur par bit. Cette procédure nous permet de déterminer les patrons de perforation suivants:

$$P_{2/4} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad P_{4/6} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix},$$

$$P_{6/8} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad P_{8/10} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Les performances d'erreur de ces codes perforés sont représentées à la figure 6.11. Comme c'est le cas pour les codes convolutionnels récursifs systématiques perforés, on observe une dégradation lorsque l'on passe d'un taux de codage $R = 2/4$ à $R = 8/10$.

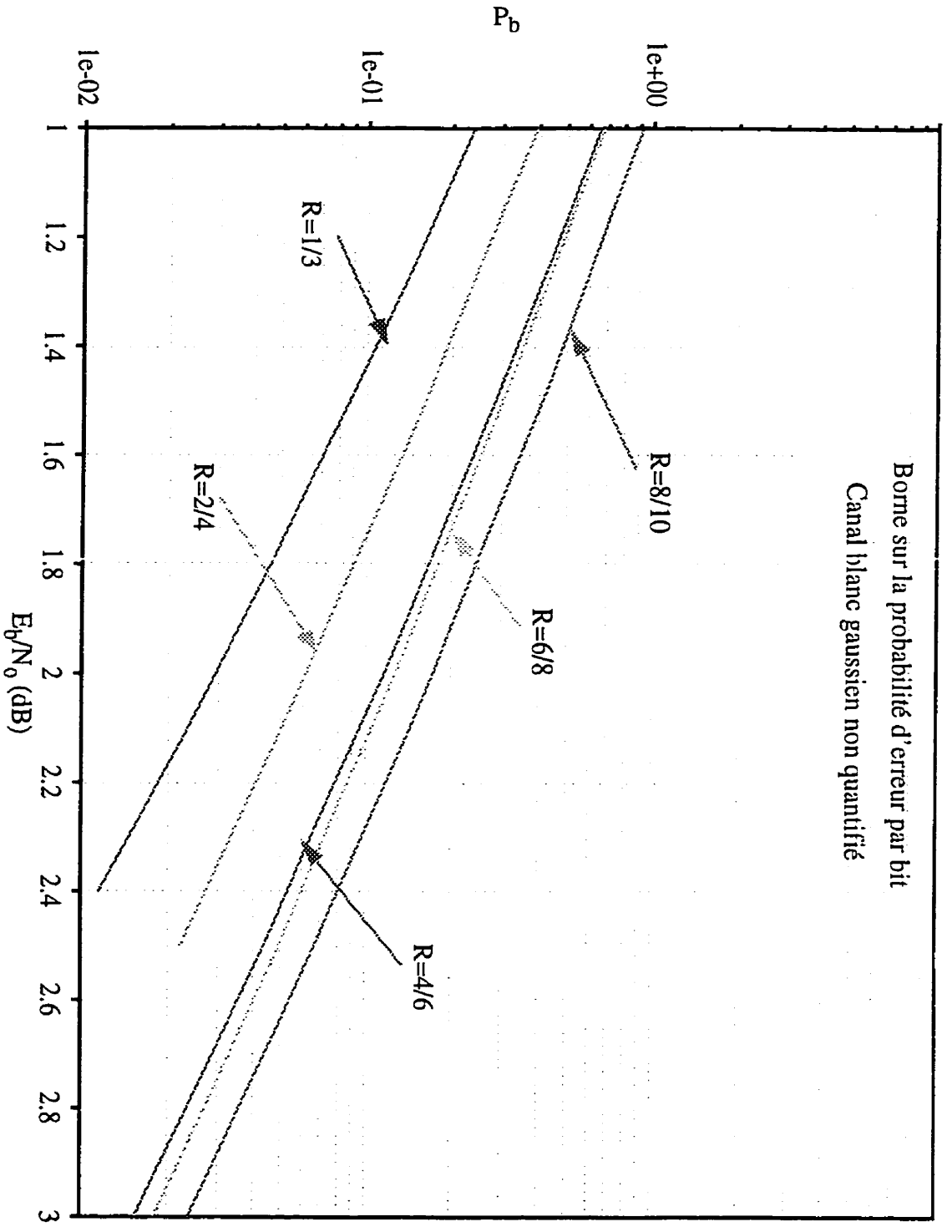


Figure 6.11: La probabilité d'erreur du code CPCC perforé avec codes récursifs systématiques R=1/2, K=3, G=(1,7/5), N=16

Les résultats que nous avons présenté nous montrent que la performance d'erreur de code CPCC est meilleure que celle de codes constituants, même si la longueur de l'entrelaceur N est petite. La performance d'erreur s'améliore aussi par l'augmentation de la longueur de l'entrelaceur N.

La technique de la perforation nous permet doncd'obtenir des codes de taux de codage élevé et variable. Elle rend donc la technique plus flexible.

CHAPITRE 7

CONCLUSION

Dans ce mémoire, nous avons présenté des méthodes pour déterminer le spectre des codes convolutionnels récursifs systématiques ainsi que celui des codes CPCC. En utilisant ces méthodes, nous avons pu évaluer les performances d'erreur de ces codes convolutionnels récursifs systématiques et CPCC.

L'analyse des performances d'erreur montre que l'utilisation de codes récursifs produit de très bonnes performances d'erreur comparées aux performances d'erreur des codes non récursifs lorsque le rapport signal à bruit est faible. En pratique, les deux codes peuvent être toutefois considérés comme équivalents car ils ont la même distance libre.

La perforation est une technique qui permet d'accroître la flexibilité des systèmes de communication numérique. L'utilisation de codes convolutionnels perforés permet des transmission avec des taux de codage variable et ce, en conservant la même complexité de décodage que celui du code origine.

Pour un code CPCC, l'entrelacement joue un rôle d'importance dans l'amélioration du gain de codage. Lorsque la taille de l'entrelaceur augmente, nous constatons une amélioration des performances d'erreur du code. Cette amélioration est proportionnelle à la taille de l'entrelaceur. Au chapitre 6, nous avons constaté que même si la valeur N de la taille de l'entrelaceur est faible, le code CPCC donne de meilleurs performances d'erreur que pour code constituant. En comparant les performances d'erreur

d'un code CPCC avec celles d'un code convolutionnel récursif systématique dont la longueur de contrainte, K, est supérieure à celle des codeurs constituants du code CPCC, nous nous apercevons que le codage CPCC offre parfois des performances d'erreur inférieure à celle du codage convolutionnel récursif systématique. Lorsque le rapport signal à bruit E_b/N_0 est faible, le meilleur choix est de prendre le codage CPCC. Par contre, quand E_b/N_0 est plus élevé, le codage convolutionnel récursif systématique peut être préféré au code CPCC.

A la fin du chapitre 6, nous avons procédé à la perforation des codes CPCC, conduisant à des codes de taux de codage variable. Comme c'est le cas pour les codes convolutionnels récursifs systématiques perforés, nous observons une dégradation lorsque l'on passe d'un taux de codage $R = 2/4$ à $R = 8/10$. Malgré une dégradation des performances qui semble être plus importante que pour les codes convolutionnels de taux de codage équivalents, les codes CPCC perforés demeurent très avantageux à des rapports signal à bruit relativement faibles.

Des travaux futures pourraient approfondir d'avantage l'analyse de cette technique de codage correcteur d'erreur. Comme mentionné au chapitre 6, le nombre de codeurs constituants formant le codeur CPCC peut être supérieur à deux. De plus, les codeur constituant peuvent être différents. Ainsi une étude pourrait porter sur le codage CPCC où le nombre de codeur constituants est supérieur à 2 et où ces codeurs sont différents. Dans ces travaux, des entrelaceurs de d'autres types (hélicoïdale, aléatoire, ...) pourraient être aussi considérés.

RÉFÉRENCES

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

[2] LIN, S. et COSTELLO, D. J. (1983). *Error Control Coding*, Prentice-Hall, Englewood Cliffs, N.J.

[3] VITREBI, A. J. (1979, April). *Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm*, IEEE Trans. Inform. Theory, vol. IT - 13, pp. 260 - 269.

[4] HACCOUN, D. (1986, Décembre). *Variablilité de calcul et débordements de décodeurs séquentiels à Pile*, Traitement du signal, Paris, France, vol. 3, No. 3, pp. 127 - 143.

[5] FORNEY, G. David. (1970, November). *Convolutional Codes I: Algebraic Structure*, IEEE Trans. Inform. Theory, vol. IT-16, pp. 720 - 738.

[6] BHARGAVA, V. K., HACCOUN, D., MATYAS, R. et NUSPL, P. P. (1981). *Digital Communications by Satellite*, John Wiley and Sons, New York.

[7] BENEDETTO, S. and MONTORSI, G. (1996, Mars). *Unveiling turbo codes:*

some results on parallel concatenated coding schemes, IEEE Trans. Inform.Theory, vol. 42, No. 2, pp. 409 - 429.

[8] PODEMSKI, R., HOLUBOWICZ, W., BERROU, C., GLAVIEUX, A. (1995, September). *Distance Spectrum of the Turbo-codes*, Proceedings 1995 IEEE International Symposium on Information Theory, Whisler, British Columbia, Canada, pp. 34.

[9] WOZENCRAFT, J. M. et JACOBS, I. M. (1965). *Principles of Communication Engineering*, John Wiley and Sons, New York.

[10] FANO, R. M. (1963, April). *A Heuristic Discussion of Probabilitic Decoding*, IEEE Trans. Inform. Theory, vol. IT-19, pp. 64 - 73.

[11] ZIGANGIROV, K. Sh. (1966). *Some Sequential Decoding Procedures*, Probl. Peredachi. Inform. vol. 2, pp. 13 - 25.

[12] JELINEK, F. (1969, November). *A Fast Sequential Decoding Algorithm Using a Stack*, IBM J. Res. Dev., vol. 13, pp. 675 - 685.

[13] BAHL, L., COCKE, J., JELINEK, F. et RAVIV, J. (1972). *Optimal decoding of linear codes for minimizing symbol error rate*, IEEE Int. Symp. Inform. Theory, pp. 90, Asilomar, CA.

[14] McADAM, P. L., WELCH, L. R. et WEBER, C. L. (1972). *M.A.P. bit decoding of convolutional codes*, IEEE Int. Symp. Inform. Theory, pp. 91, Asilomar, CA.

[15] BERROU, C., GLAVIEUX, A., THITIMAJSHIMA, P. (1993, May). *Near Shannon limit error - correcting coding and decoding: Turbo-codes*, Proceeding of IEEE International Conference on Communications, Genève, Suisse, pp. 1064-1070.

[16] ROBERTSON, P. (1994). *Improving decoder and code structure of parallel concatenated recursive systematic (turbo) codes*, Int. Conf. Universal Personal Commun., pp. 183-187, San Diego, CA.

[17] BERROU, C. et GLAVIEUX, A. (1996, Octobre). *Near optimum error correcting coding and decoding: turbo codes*, IEEE Trans. Comm., vol. 44, pp. 1261-1271.

[18] BOUZOUITA, N. (1997). *Sur le décodage itératif des codes turbo*, mémoire de maîtrise en sciences appliquées, École Polytechnique de Montréal.

[19] HACCOUN, D., *Théorie des communications -ELE 6703*, note de cours, École Polytechnique de Montréal

[20] MONTREUIL, P. (1987). *Algorithmes de détermination de spectres des codes convolutionnels*, mémoire de maîtrise en sciences appliquées, École Polytechnique de Montréal.

[21] PAQUIN, C. (1990). *Algorithmes de détermination de spectres des codes convolutionnels perforés*, mémoire de maîtrise en sciences appliquées, École Polytechnique de Montréal.

[22] OHASHI, M., YASUDA, Y., et HIRATA, Y. (1988, November). *Development of a variable rate syndrome sequential decoder based on a stack algorithm*, Proc. IEEE Globecom'88, Florida, pp. 5.2.1-5.2.5.

[23] LARSEN, K. L. (1973, May). *Short convolutional codes with maximal free distance for rates 1/2, 1/3 and 1/4*, IEEE Trans. Inform. Theory, vol. IT - 19, pp. 371 - 372.

[24] CEDERVALL, M. et JOHNNESSON R. (1989, November). *A fast algorithm for computing distance spectrum of convolutional codes*, IEEE Trans. Inform. Theory, vol. IT - 35, pp. 1146 - 1159.

[25] BEGIN, G., HACCOUN, D. and PAQUIN, C. (1990, November). *Further results on high-rate punctured convolutional codes for Viterbi and sequential decoding*, IEEE Trans. on Comm., vol. 38, pp. 1922 - 1928.

[26] CAIN, J. B., CLARK, G. C. and GEIST, J. M. (1979, January). *Punctured convolutional codes of rate (n-1)/n and simplified maximum likelihood decoding*, IEEE Trans. Inform. Theory, vol. IT - 25, pp. 97 - 100.

[27] HACCOUN, D. and BEGIN, G. (1987, Juin). *Codage et décodage séquentiel de codes convolutionnels perforés*, Proc. 11th GRETSI Conf., Nice, France, pp. 221 - 224.

[28] SVIRID, Y. V. (1995, September). *Weight distributions of Turbo-Codes*, Proceedings 1995 IEEE International Symposium on Information Theory, Whisler,

British Columbia, Canada, pp. 38.

[29] PEREZ, L., SEGHERS, J. et COSTELLO, D. J. (1996). *A distance spectrum interpretation of turbo codes*, IEEE Trans. Inform. Theory, vol. 42-6, pp. 1698 - 1709.

ANNEXE I

CALCUL DE FONCTION DE TRANSFERT

A.1.1 FONCTION DE TRANSFERT D'UN CODE CONVOLUTIONNEL RÉCURSIF SYSTÉMATIQUE

Un code convolutionnel récursif systématique avec le taux de codage R = 1/2, la longueur de contrainte K = 3, le vecteur de générateurs G = (1, 7/5) est donné à la figure A.1.1.1. Et son diagramme d'état est illustré à la figure A.1.1.2.

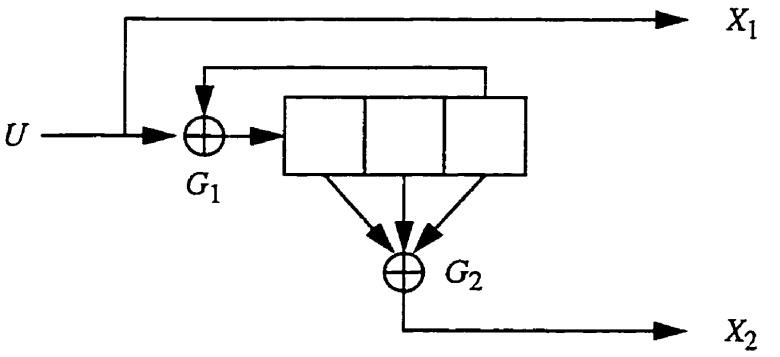


Figure A.1.1.1: Codeur convolutionnel récursif systématique

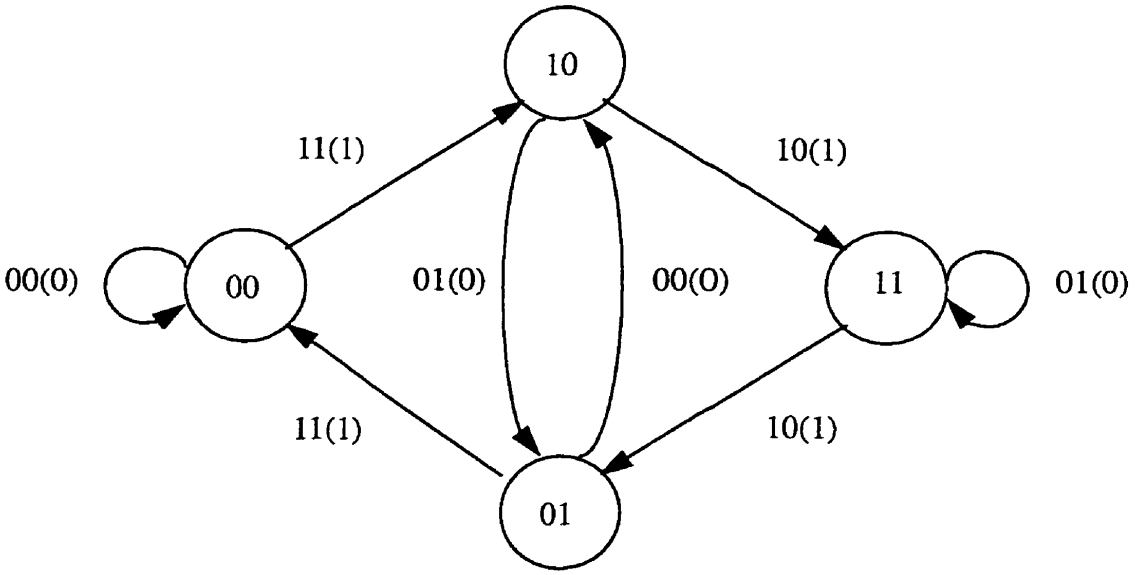


Figure A.1.1.2: Diagramme d'état du codeur de la figure A.1.1.1

Pour calculer la fonction de transfert, il faut modifier le diagramme d'état. Le diagramme d'état modifié de la figure A.1.1.2 peut être représentée comme tel:

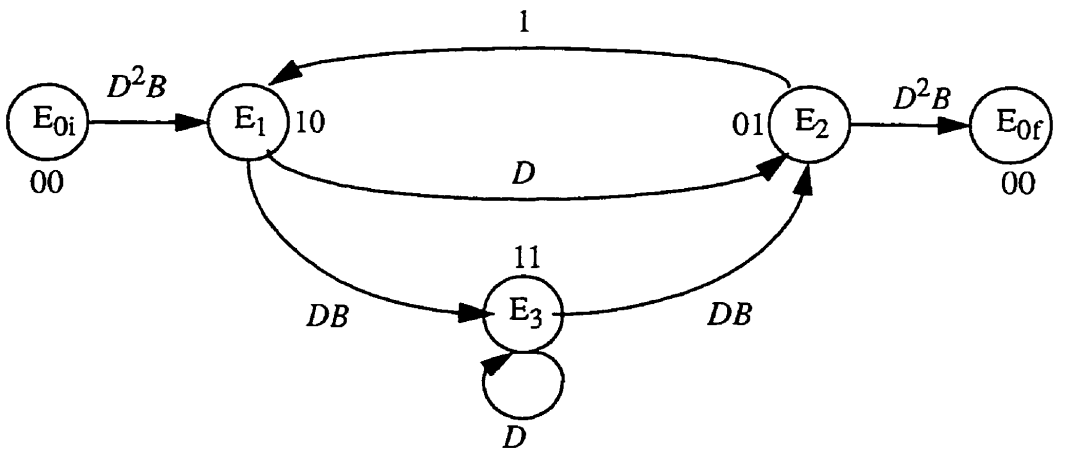
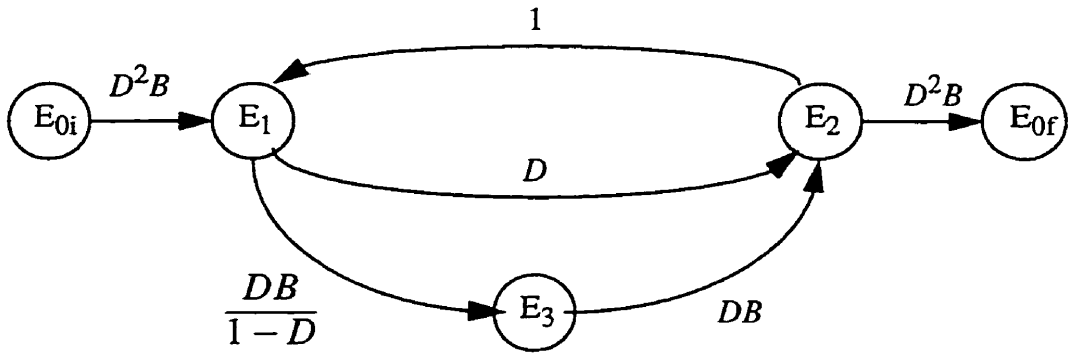
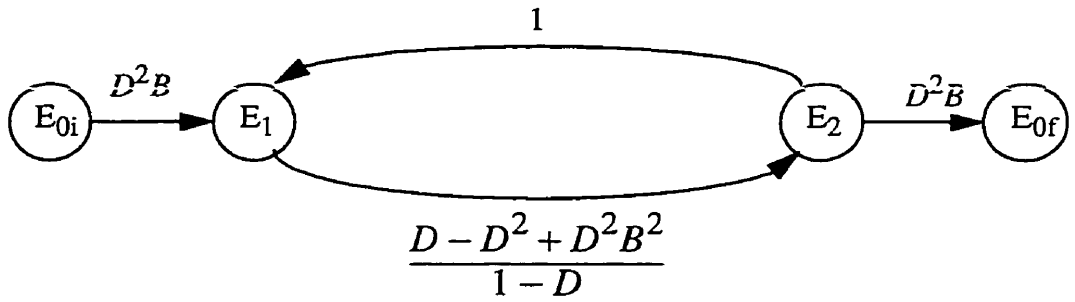
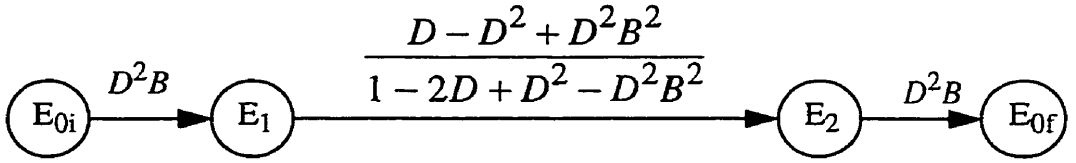


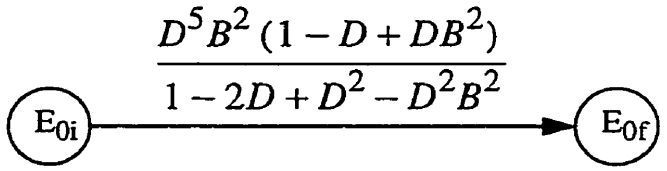
Figure A.1.1.3: Diagramme d'état modifié du codeur de la figure A.1.1.1

La défilé de calcul est présentée comme suit:









Donc, la fonction de transfert de la figure A.1.1.1 est:

$$T(D,B) = \frac{D^5B^2(1-D+DB^2)}{1-2D+D^2-D^2B^2}$$

A.1.2 FONCTION DE TRANSFERT D'UN CODE CONVOLUTIONNEL RÉCURSIF SYSTÉMATIQUE PERFORÉ

Un code convolutionnel récursif systématique perforé avec le taux de codage initial $R = 1/2$, la longueur de contrainte $K = 3$, le vecteur de générateurs $G = (1, 7/5)$ est donné à la figure A.1.2.1. Et son diagramme d'état est illustré à la figure A.1.2.2.

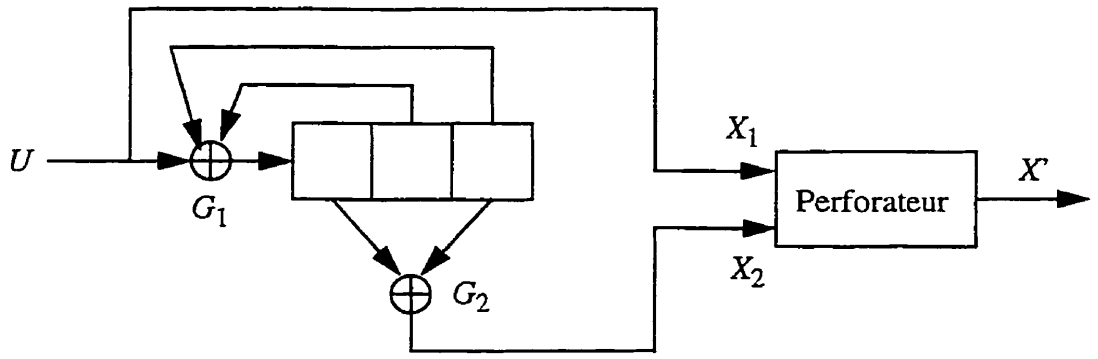


Figure A.1.2.1: Codeur convolutionnel récursif systématique perforé

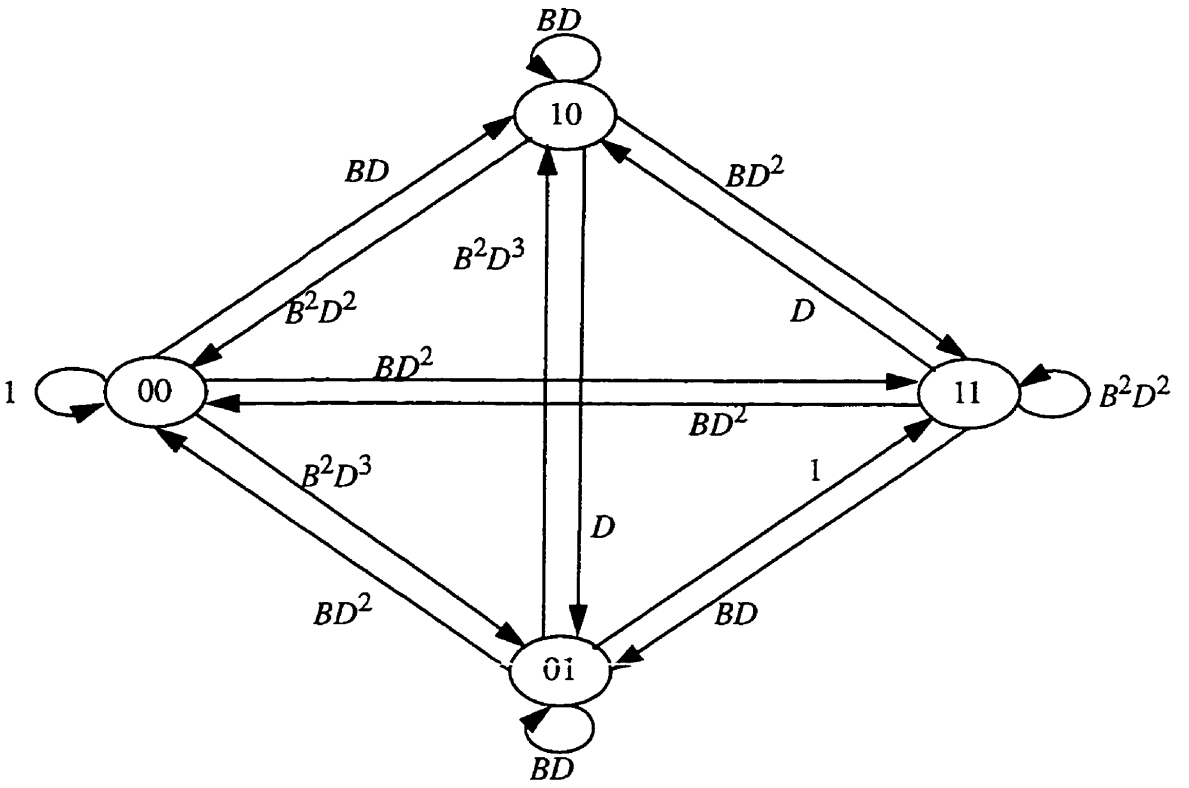


Figure A.1.2.2: Diagramme d'état du codeur de la figure A.1.2.1

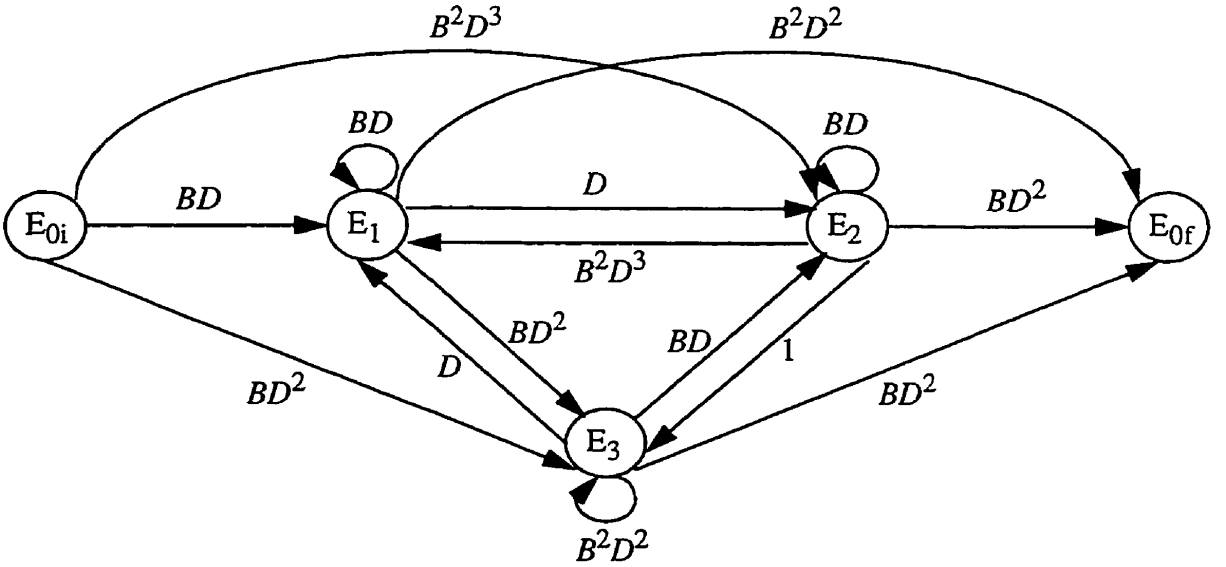


Figure A.1.2.3: Diagramme d'état modifié du codeur de la figure A.1.2.1

La figure A.1.2.3 présente le diagramme d'état modifié du codeur de la figure A.1.2.1. Avec la forme matricielle, la fonction de transfert peut obtenir par la façon suivante.

Les équations matricielles sont:

$$\begin{bmatrix} E_1 \\ E_2 \\ E_3 \end{bmatrix} = \begin{bmatrix} BD & B^2D^3 & D \\ D & BD & BD \\ BD^2 & 1 & B^2D^2 \end{bmatrix} \cdot \begin{bmatrix} E_1 \\ E_2 \\ E_3 \end{bmatrix} + \begin{bmatrix} BD \\ B^2D^3 \\ BD^2 \end{bmatrix} \cdot E_{0i}$$

$$E_{0f} = \begin{bmatrix} B^2D^2 & BD^2 & BD^2 \end{bmatrix} \cdot \begin{bmatrix} E_1 \\ E_2 \\ E_3 \end{bmatrix}$$

Posons:

$$A = \begin{bmatrix} BD & B^2D^3 & D \\ D & BD & BD \\ BD^2 & 1 & B^2D^2 \end{bmatrix}$$

$$F = \begin{bmatrix} BD \\ B^2D^3 \\ BD^2 \end{bmatrix}$$

$$G^T = \begin{bmatrix} B^2D^2 & BD^2 & BD^2 \end{bmatrix}$$

Ces deux équations peuvent s'écrire comme suivant:

$$E = AE + FE_{0i}$$

$$E_{0f} = G^T E$$

où E est le vecteur colonne contenant tous les états intermédiaires;

A, la matrice des transitions entre les états intermédiaires;

F, le vecteur colonne des transitions entre l'état E_{0i} et tous les états intermédiaires;

G, le vecteur colonne des transitions entre tous les états intermédiaires et l'état E_{of}.

Donc, la fonction de transfert s'écrit:

$$T(D,B) = \frac{E_{0f}}{E_{0i}} = G^T [I-A]^{-1} F$$

Quand la dimension de la matrice est grand, elle est difficile à calculer. Il est possible de faire l'inversion $[I-A]^{-1}$ en exprimant ce terme sous forme de série,

$$[I-A]^{-1} = I + A + A^2 + A^3 + \ldots$$

Dans ce cas, la fonction de transfert est obtenue:

$$T(D,B) = G^T F + G^T A F + G^T A^2 F + G^T A^3 F + \ldots$$

$$= D^3 B^3 + D^4 (3B^2 + B^4) + D^5 (13B^3 + B^5) + \ldots$$

ANNEXE II

SPECTRES DES CODES CONVOLUTIONNELS RÉCURSIFS SYSTÉMATIQUES

Dans cette annexe, nous présentons des résultats des spectres des codes convolutionnels récursifs systématiques.

Tableau A.2.1: Spectres de codes convolutionnels non systématiques et des codes convolutionnels récursifs systématiques, $R = 1/2$, $3 \leq K \leq 9$

K	G_1 G_2	d_{free}	(A_n) $[C_n]$ $\{Cr_n\}$
3	5 7	5	(1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384) [1, 4, 12, 32, 80, 192, 448, 1024, 2304, 5120, 11264, 24576, 53248, 114688, 245760] {2, 6, 14, 32, 72, 160, 352, 768, 1664, 3584, 7680, 16384, 34816, 73728, 155648}
4	15 17	6	(1, 3, 5, 11, 25, 55, 121, 267, 589, 1299, 2865, 6319, 13937, 30739, 67797) [2, 7, 18, 49, 130, 333, 836, 2069, 5060, 12255, 29444, 70267, 166726, 393635, 925334] {4, 9, 20, 51, 124, 303, 728, 1739, 4134, 9771, 22990, 53885, 125858, 293049, 680440}
5	23 35	7	(2, 3, 4, 16, 37, 68, 176, 432, 925, 2156, 5153, 11696, 26868, 62885, 145085) [4, 12, 20, 72, 225, 500, 1324, 3680, 8967, 22270, 57403, 142234, 348830, 867106, 2134239] {8, 12, 16, 84, 213, 406, 1156, 3104, 7021, 17372, 44427, 106518, 257200, 634556, 1537281}
6	53 75	8	(1, 8, 7, 12, 48, 95, 281, 605, 1272, 3334, 7615, 18131, 43197, 99210, 237248) [2, 36, 32, 62, 332, 701, 2342, 5503, 12506, 36234, 88576, 225685, 574994, 1400192, 3554210] {4, 34, 32, 62, 288, 604, 1884, 4430, 9926, 27794, 67380, 168606, 424768, 1025664, 2570672}
7	133 171	10	(11, 0, 38, 0, 193, 0, 1331, 0, 7275, 0, 40406, 0, 234969, 0, 1337714) [36, 0, 211, 0, 1404, 0, 11633, 0, 77433, 0, 502690, 0, 3322763, 0, 21292910] {60, 0, 223, 0, 1368, 0, 10963, 0, 66171, 0, 408918, 0, 2619965, 0, 16222096}
8	247 371	10	(1, 6, 12, 26, 52, 132, 317, 730, 1823, 4446, 10739, 25358, 60773, 146396, 350399) [2, 22, 60, 148, 340, 1008, 2642, 6748, 18312, 48478, 126364, 320062, 821350, 2102864, 5335734] {6, 28, 70, 182, 360, 984, 2530, 6156, 16308, 41924, 107014, 265396, 666098, 1677978, 4189876}
9	561 753	12	(11, 0, 50, 0, 286, 0, 1630, 0, 9639, 0, 55152, 0, 320782) [33, 0, 281, 0, 2179, 0, 15035, 0, 105166, 0, 692330, 0, 4580007] {67, 0, 349, 0, 2295, 0, 14575, 0, 96680, 0, 606538, 0, 3848633}

Tableau A.2.1 (suite): Spectre des codes convolutionnels non systématiques et des codes convolutionnels récursifs systématiques, $R = 1/2$, $10 \leq K \leq 16$

K	G_1 G_2	d_{free}	(A_n) $[C_n]$ $\{Cr_n\}$
10	1167 1545	12	(2, 8, 15, 35, 68, 170, 458, 1084, 2574, 6177, 14939, 36200, 86856) [14, 26, 74, 257, 496, 1378, 4122, 10832, 27988, 72209, 186920, 483102, 1234736] {14, 50, 96, 271, 552, 1428, 4106, 10192, 25644, 64627, 163408, 414532, 1038094}
11	2335 3661	14	(21, 0, 74, 0, 454, 0, 2687, 0, 15629, 0, 90518, 0, 526556) [94, 0, 463, 0, 3783, 0, 26711, 0, 181571, 0, 1207474, 0, 7919894] {148, 0, 585, 0, 4085, 0, 26881, 0, 172285, 0, 1086924, 0, 6849856}
12	4335 5723	15	(16, 31, 44, 129, 309, 697, 1713, 4175, 10158, 24508, 58600, 141960, 343347) [76. 180. 374. 1142. 2783. 6836. 18709. 49242. 128178. 329408. 836478. 2151230, 5497355] {122, 234, 360, 1134, 2875, 6854, 17637, 45154, 114532, 289116, 721920, 1818368, 4564999}
13	10533 17661	16	(33, 0, 111, 0, 779, 0, 4128, 0, 24173, 0, 142500, 0, 828402) [152, 0, 971, 0, 6933, 0, 45436, 0, 303435, 0, 2036131, 0, 13256560] {268, 0, 1049, 0, 7973, 0, 46662, 0, 296423, 0, 1892487, 0, 11821688}
14	21675 27123	16	(4, 17, 35, 76, 193, 454, 1047, 2624, 6138, 14944, 36179, 86640, 210568) [22, 99, 218, 608, 1724, 4404, 11108, 30438, 75942, 196714, 507232, 1289364, 3311290] {34, 143, 310, 722, 1890, 4708, 11420, 30044, 73432, 185730, 467800, 1163266, 2933186}
15	44735 63057	18	(26, 0, 165, 0, 845, 0, 4844, 0, 28513, 0, 166583, 0, 968884) [133, 0, 1321, 0, 7901, 0, 54864, 0, 370057, 0, 2450650, 0, 15893608] {229, 0, 1627, 0, 9191, 0, 57526, 0, 367507, 0, 2309798, 0, 14416882}
16	126723 152711	19	(30, 67, 54, 167, 632, 1402, 2812, 7041, 18178, 43631, 104526, 251134, 605947) [174, 420, 534, 1712, 5838, 14210, 32898, 87786, 237228, 609868, 1556396, 3945414, 10039681] {292, 666, 552, 1880, 7304, 16870, 35336, 91922, 245526, 612890, 1520392, 3779288, 9423360}

Tableau A.2.2: Spectres de codes convolutionnels non systématiques et des codes convolutionnels récursifs systématiques, $R = 1/3$, $3 \leq K \leq 9$

K	G_1 G_2 G_3	d_{free}	(A_n) $[C_n]$ $\{Cr_n\}$
3	5 7 7	8	(2, 0, 5, 0, 13, 0, 34, 0, 89, 0, 233, 0, 610, 0, 1597, 0, 4181, 0, 10946) [3, 0, 15, 0, 58, 0, 201, 0, 655, 0, 2052, 0, 6255, 0, 18687, 0, 54974, 0, 159765] {6, 0, 20, 0, 64, 0, 198, 0, 598, 0, 1774, 0, 5190, 0, 15016, 0, 43052, 0, 122502}
4	15 17 13	10	(3, 0, 2, 0, 15, 0, 24, 0, 87, 0, 188, 0, 557, 0, 1354, 0, 3713, 0, 9448) [6, 0, 6, 0, 58, 0, 118, 0, 507, 0, 1284, 0, 4323, 0, 11846, 0, 36009, 0, 100844] {10, 0, 8, 0, 70, 0, 128, 0, 523, 0, 1254, 0, 4089, 0, 10838, 0, 32207, 0, 88232}
5	25 33 37	12	(5, 0, 3, 0, 13, 0, 62, 0, 108, 0, 328, 0, 1051, 0, 2544, 0, 7197, 0, 20658) [12, 0, 12, 0, 56, 0, 320, 0, 693, 0, 2324, 0, 8380, 0, 23009, 0, 71016, 0, 222592] {20, 0, 14, 0, 72, 0, 368, 0, 711, 0, 2436, 0, 8368, 0, 21927, 0, 67312, 0, 205852}
6	47 53 75	13	(1, 3, 6, 4, 5, 12, 14, 33, 66, 106, 179, 317, 513, 766, 1297, 2251, 3964, 6721, 10969, 18818) [1, 8, 26, 20, 19, 62, 86, 204, 420, 710, 1345, 2606, 4343, 6790, 12305, 22356, 41090, 72820, 123901, 221886] {4, 14, 32, 20, 28, 70, 96, 232, 458, 772, 1356, 2576, 4290, 6684, 11798, 21132, 38692, 67378, 113524, 201222}
7	133 145 175	15	(3, 5, 5, 6, 11, 15, 25, 54, 92, 164, 274, 450, 758, 1290, 2142, 3567, 6089, 10403, 17749, 29715) [11, 16, 19, 28, 55, 96, 169, 338, 636, 1276, 2172, 3628, 6580, 12048, 20820, 36358, 65009, 115368, 204997, 356650] {13, 26, 35, 40, 71, 100, 173, 410, 726, 1354, 2330, 4002, 7020, 12282, 21088, 36284, 64243, 113084, 198575, 342770}
8	225 331 367	16	(1,0, 8, 0, 24, 0, 51, 0, 133, 0, 405, 0, 1129, 0, 3532, 0, 9754, 0, 28746) [1, 0, 24, 0, 113, 0, 287, 0, 898, 0, 3020, 0, 9436, 0, 32644, 0, 98472, 0, 318914] {4, 0, 50, 0, 164, 0, 390, 0, 1058, 0, 3506, 0, 10730, 0, 35496, 0, 105278, 0, 328492}
9	561 753 715	18	(4, 0, 13, 0, 28, 0, 81, 0, 235, 0, 646, 0, 1889, 0, 5608, 0, 16007, 0, 46407) [11, 0, 49, 0, 136, 0, 496, 0, 1652, 0, 5122, 0, 16388, 0, 53870, 0, 167364, 0, 529226] {23, 0, 91, 0, 218, 0, 612, 0, 2018, 0, 6046, 0, 18874, 0, 59700, 0, 181378, 0, 556894}

ANNEXE III

SPECTRES DES CODES CONVOLUTIONNELS RÉCURSIFS SYSTÉMATIQUES PERFORÉS

Un codeur convolutionnel récursif systématique perforé est illustré à la figure A.3.1. Le taux de codage de code origine est $R = 1/2$.

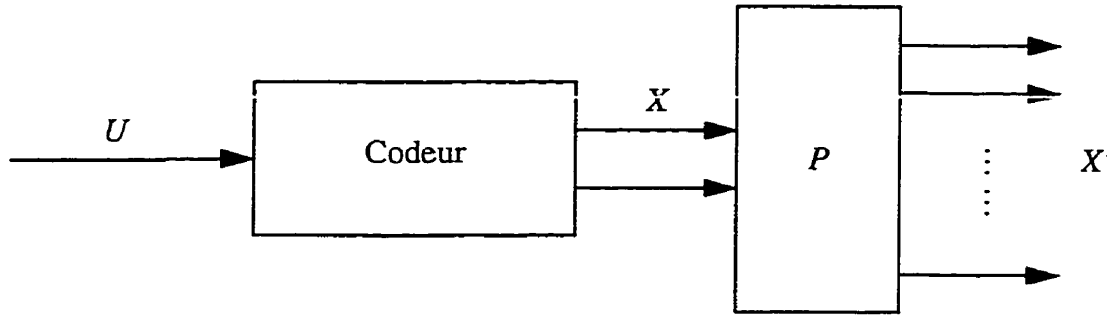


Figure A.3.1: Codeur convolutionnel récursif systématique perforé

Avec les différents patrons de perforation, on obtient les codes perforés de taux de codage variable. Voici les tableaux des spectres de codes perforés de taux de codage allant de 2/3 à 7/8.

Tableau A.3.1: Spectre des codes perforés: $R = 2/3$, $P = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$, $3 \leq K \leq 9$

codes origines		codes perforés $R = 2/3$	
K	G	d_f	(a_d) $[c_d]$
3	1, 5/7	3	(1, 4, 14, 40, 115, 331, 953, 2744, 7901, 22750, 65506, 188617, 543101, 1563797, 4502774) [3, 10, 44, 154, 521, 1724, 5609, 18008, 57201, 180106, 562944, 1748632, 5402681, 16615138, 50889898]
4	1, 17/15	4	(3, 11, 35, 114, 378, 1253, 4147, 13725, 45428, 150362, 497681, 1647267, 5452266, 18046379, 59731456) [10, 33, 146, 538, 2046, 7595, 27914, 101509, 366222, 1312170, 4674258, 16567175, 58462900, 205511847, 719960394]
5	1, 35/23	4	(1, 0, 27, 0, 345, 0, 4515, 0, 59058, 0, 772627, 0, 10107642, 0, 132230266) [3, 0, 106, 0, 1841, 0, 30027, 0, 471718, 0, 7201171, 0, 107686039, 0, 1585096699]
6	1, 53/75	6	(19, 0, 220, 0, 3089, 0, 42725, 0, 586592, 0, 8085210, 0, 111315783, 0, 1532925641) [82, 0, 1260, 0, 21530, 0, 354931, 0, 5643947, 0, 88444551, 0, 1364368734, 0, 20808088020]
7	1, 171/133	6	(1, 16, 48, 158, 642, 2435, 9174, 34701, 131533, 499312, 1891754, 7165914, 27160547, 102939934, 390103650) [3, 76, 269, 960, 4290, 18034, 74197, 303431, 1237276, 5030802, 20324463, 81763094, 328002734, 1311800821, 5231393084]
8	1, 371/247	6	(3, 0, 71, 0, 1032, 0, 14469, 0, 210819, 0, 3041149, 0, 43985963, 0, 635562375) [13, 0, 392, 0, 7081, 0, 118852, 0, 2008342, 0, 33041542, 0, 536313579, 0, 8595455776]
9	1, 753/561	7	(3, 9, 50, 190, 641, 2507, 9745, 37120, 142220, 544948, 2086538, 7988101, 30588852, 117140948, 448544740) [15, 50, 292, 1246, 4703, 20167, 84481, 346376, 1422291, 5809959, 23643588, 95854870, 387422765, 1561740153, 6279014364]

Tableau A.3.2: Spectre des codes perforés: R= 3/4, $P = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$, $3 \leq K \leq 9$

codes origines		codes perforés R= 3/4	
K	G	d_f	(a_d) $[c_d]$
3	1, 7/5	2	(1, 6, 21, 75, 256, 874, 2984, 10188, 34784, 118760, 405472, 1384368, 4726528, 16137376) [2, 12, 68, 282, 1136, 4464, 17240, 65686, 247568, 924808, 3429120, 12635144, 46305408, 168907072]
4	1, 17/15	3	(3, 14, 64, 276, 1181, 5076, 21818, 93770, 403030, 1732226, 7445147, 31999355, 137533780, 591122529) [8, 45, 249, 1271, 6286, 30630, 147212, 699524, 3293876, 15391804, 71461008, 329949003, 1516157153, 6937807892]
5	1, 35/23	3	(1, 7, 43, 204, 936, 4437, 21239, 101402, 483434, 2304342, 10985361, 52373300, 249692847, 1190417228) [3, 22, 176, 972, 5124, 27584, 147598, 779006, 4067691, 21075076, 108510514, 555667057, 2831949076, 14372743145]
6	1, 53/75	4	(1, 15, 65, 321, 1661, 8388, 42560, 215586, 1091757, 5533847, 28041728, 142076134, 719917788, 3647942648) [3, 59, 300, 1700, 10129, 57470, 322813, 1796614, 9916116, 54393318, 296572226, 1608781414, 8689784165, 46757907651]
7	1, 171/133	4	(3, 0, 149, 0, 3535, 0, 93326, 0, 2423904, 0, 63086012, 0, 1641535038, 0, 42714324205) [10, 0, 731, 0, 22405, 0, 730817, 0, 22568801, 0, 680930247, 0, 20151179191, 0, 587666565700]
8	1, 371/247	5	(4, 22, 132, 685, 3404, 17774, 93755, 488788,2546150, 13288084, 69343776, 361792069, 1887695732, 9849443753) [15, 104, 697, 4145, 23229, 134468, 778842, 4426092, 24963512, 140202159, 783428340, 4357851991, 24148250576, 133357459539]
9	1, 753/561	6	(10, 77, 303, 1599, 8565, 44820, 236282, 1236772, 6486761, 34044045, 178618742, 937256996, 4917567406, 25801447415) [52, 413, 1855, 11052, 65339, 376208, 2160054, 12230957, 69017434, 387700389, 2167946734, 12077865040, 67053023310, 371140197835]

Tableau A.3.3: Spectre des codes perforés: R= 4/5, $P = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$, $3 \leq K \leq 9$

codes origines		codes perforés R= 4/5	
K	G	d_f	(a_d) $[c_d]$
3	1, 5/7	2	(1, 15, 63, 261, 1084, 4493, 18633, 77266, 320406, 1328654, 5509642, 22847297, 94742814, 392878019) [2, 39, 208, 1047, 5136, 24553, 115366, 534550, 2449540, 11123380, 50130692, 224486675, 999757294, 4431323237]
4	1, 17/15	2	(1, 0, 93, 0, 2492, 0, 67375, 0, 1821412, 0, 49239974, 0, 1331151331, 0, 35986287679) [2, 0, 313, 0, 12104, 0, 428235, 0, 14307373, 0, 460599650, 0, 14447335914, 0, 444515048843]
5	1, 35/23	3	(4, 20, 137, 843, 4921, 29703, 179774, 1079279, 6485983, 39041796, 234880540, 1412702547, 8498248986) [11, 67, 554, 4116, 27890, 191449, 1301196, 8664735, 57177578, 374954161, 2441068349, 15795921674, 101722574261]
6	1, 53/75	4	(7, 54, 307, 2005, 12962, 83111, 532859, 3417085, 21921778, 140627199, 902063096, 5786457561, 37118404365) [22, 217, 1465, 11199, 82844, 597496, 4256276, 30025829, 210156840, 1460567718, 10090038729, 69350549839, 474537473615]
7	1, 171/133	4	(3, 24, 172, 1158, 7408, 48706, 319563, 2094852, 13737566, 90083445, 590747933, 3873778522, 25402176247) [10, 96, 848, 6552, 47789, 353697, 2575710, 18559164, 132683604, 942057640, 6649959820, 46702503877, 326551842734]
8	1, 247/371	4	(2, 15, 105, 716, 4763, 31439, 207325, 1375926, 9108602, 60276432, 399209237, 2642959760, 17498295231) [4, 56, 512, 3964, 30238, 224286, 1647094, 12023274, 86923048, 623362184, 4448352102, 31565323588, 222996659970]
9	1, 753/561	5	(8, 63, 418, 2731, 18066, 121050, 805626, 5366503, 35743953, 238127232, 1586235000, 10566977218) [37, 317, 2409, 18093, 134318, 994728, 7265477, 52678793, 379419665, 2717687374, 19369590288, 137466797748]

Tableau A.3.4: Spectre des codes perforés: R= 5/6, $P = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$, $3 \leq K \leq 9$

codes origines		codes perforés R= 5/6	
K	G	d_f	(a_d) $[c_d]$
3	1, 7/5	2	(6, 20, 125, 537, 2543, 11581, 53529, 246036, 1133245, 5215601, 24011238, 110528930, 508810109, 2342224891) [12, 40, 452, 2098, 12314, 63536, 334840, 1714858, 8734868, 43998460, 220137510, 1094080968, 5408535126, 26609243310]
4	1, 15/17	2	(3, 9, 124, 571, 3942, 22423, 139674, 832631, 5068241, 30544540, 184957531, 1117445884, 6758497057) [6, 18, 472, 2260, 20540, 128040, 929686, 6113002, 41416110, 272532432, 1797004930, 11718051394, 76161743182]
5	1, 23/35	2	(2, 6, 99, 486, 3713, 24203, 166853, 1119504, 7605354, 51389289, 348067467, 2355007217, 15941219634) [4, 12, 384, 1924, 19824, 140282, 1131802, 8378774, 63345462, 467806530, 3449875718, 25206336042, 183372631506]
6	1, 53/75	4	(19, 171, 1251, 9573, 75097, 584394, 4543202, 35354659, 275053493, 2139791948, 16647596153, 129517325265, 1007630096839) [64, 730, 6311, 56329, 504412, 4408689, 38045121, 325388978, 2759573175, 23243056930, 194639020569, 1621703112560, 13452410524783]
7	1, 171/133	3	(1, 17, 136, 1143, 8717, 69488, 550980, 4362319, 34545004, 273640159, 2167394710, 17166768465, 135969904852) [3, 61, 591, 5900, 52293, 474356, 4213541, 36953475, 321117927, 2769163611, 23719401170, 202015463354, 17121201139960]
8	1, 371/247	3	(2, 10, 87, 902, 6409, 52777, 407216, 3288190, 25790336, 205819300, 1625226521, 12916035300, 102250258947) [6, 37, 392, 4749, 39393, 365955, 3170564, 28260558, 243263979, 2109634230, 18009782102, 153748856818, 1301852217433]
9	1, 753/561	4	(4, 22, 235, 1930, 15238, 123828, 997012, 8048187, 64967995, 524339542, 4232340197, 34160984107, 275726621389) [14, 90, 1198, 11520, 103313, 941421, 8411183, 74598836, 656274294, 5732956268, 49796748995, 430358783789, 3703047821477]

Tableau A.3.5: Spectre des codes perforés: R= 6/7, $P = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$, $3 \leq K \leq 9$

codes origines		codes perforés R= 6/7	
K	G	d_f	(a_d) $[c_d]$
3	1, 5/7 ou 7/5		code catastrophe
4	1, 17/15	2	(1, 26, 192, 1388, 10052, 72644, 525350, 3799172, 27473567, 198676562, 1436739394, 10389842596, 75134610829) [2, 71, 676, 6006, 51799, 434158, 3571194, 28947553, 231911059, 1840340640, 14489164500, 113316980434, 881199400986]
5	1, 35/23	3	(12, 118, 958, 7932, 66326, 551950, 4601210, 38354282, 319689503, 2664700520, 22211085307, 185136119920, 1543165517265) [33, 415, 4192, 41251, 400222, 3800203, 35546642, 328557809, 3007418496, 27308395871, 246301067529, 2208674858883, 19707619362592]
6	1, 53/75	3	(6, 58, 505, 4513, 40512, 362645, 3245628, 29043555, 259911997, 2325966150, 20815077135, 186274374639, 1666972494258) [17, 206, 2217, 23644, 246703, 2516513, 25280464, 250902052, 2466193353, 24046564408, 232880209004, 2242333051415, 21483174010247]
7	1, 171/133	3	(2, 32, 310, 2767, 25617, 235749, 2170952, 19982803, 183932576, 1693280392, 15587431026, 143489127969, 1320889297039) [6, 115, 1383, 14653, 157453, 1650726, 17044626, 173880868, 1756984105, 17615022556, 175413117271, 1736806934501, 17111725494696]
8	1, 371/247	3	(1, 10, 166, 1569, 14215, 132651, 1248662, 11729248, 110005979, 1031979124, 9682806892, 90849887311, 852394665491) [3, 35, 736, 8298, 87244, 926917, 9790066, 101975775, 1050197245, 10731821151, 108950666199, 1099708134201, 110446356903423]
9	1, 753/561	4	(4, 89, 834, 7475, 72569, 681900, 6525520, 61666545, 587936915, 5570911133, 53008105139, 502953593408) [14, 397, 4407, 46353, 510406, 5383760, 57060265, 592380594, 6148586707, 63045996598, 645148914108, 6552655547977]

Tableau A.3.6: Spectre des codes perforés: R= 7/8, $P = \begin{bmatrix} 1\,1\,1\,1\,1\,1\,1 \\ 1\,0\,0\,0\,0\,0\,0 \end{bmatrix}$, $3 \leq K \leq 9$

codes origines		codes perforés R= 7/8	
K	G	d_f	(a_d) $[c_d]$
3	1, 5/7	2	(15, 42, 410, 1961, 12098, 66509, 384606, 2174619, 12418031, 70602653, 402187161, 2289101114, 13033625952) [30, 84, 1544, 7746, 61526, 376318, 2510540, 15745260, 99754332, 620283852, 3844047560, 23624661706, 144514376724]
4	1, 15/17	2	(7, 20, 468, 2220, 24773, 160767, 1443537, 10577108, 87775767, 674166699, 5430888198, 42475780875) [14, 40, 1824, 8832, 136216, 938676, 10178870, 80477068, 757601224, 6285107710, 55629010856, 467484667268]
5	1, 35/23	2	(1, 24, 244, 2314, 21681, 203474, 1912257, 17970939, 168865859, 1586787123, 14910712052, 140112652575) [2, 68, 890, 10369, 115818, 1260624, 13476145, 141975232, 1478134269, 15243020779, 155953205581, 1584964195405]
6	1, 53/75	3	(9, 104, 1158, 11631, 117903, 1207526, 12336826, 125986874, 1286984086, 13146417589, 134286850613, 1371711244728) [26, 374, 5138, 61913, 730299, 8527353, 97850461, 1108869028, 12446576835, 138573662105, 1532278354033, 16844853957311]
7	1, 171/133	3	(2, 42, 468, 4939, 52778, 567636, 6073705, 65082861, 697229127, 7470272515, 80030957513, 857429385531) [6, 149, 2084, 26181, 326066, 4000330, 48097368, 572029038, 6735452087, 78668779079, 912491762357, 10522775452036]
8	1, 371/247	3	(5, 43, 489, 5315, 56796, 608559, 6518056, 69776612, 747112342, 7999939742, 85659510275, 917197140316) [15, 160, 2210, 28699, 356335, 4348194, 52245945, 620027996, 7289109124, 85014545924, 984860745282, 11343784611595]
9	1, 753/561	4	(19, 175, 1816, 20390, 223482, 2438515, 26634212, 290976737, 3178919625, 34730774828, 379438499089, 4145414355723) [68, 783, 9802, 127513, 1593091, 19515293, 236425671, 2837396147, 33778150788, 399404710184, 4695317957775, 54921533970859]

ANNEXE IV

SPECTRES DES CODES CPCC

Le schéma de principe de codage CPCC est illustré à la figure A.4.1. Ce codeur consiste à utiliser deux codeurs convolutionnels récursifs systématiques séparés par un entrelaceur. Les codes conctituants sont identiques, de taux de codage R = 1/2, de la longueur de contrainte K = 3, et de générateur G = (1, 7/5).

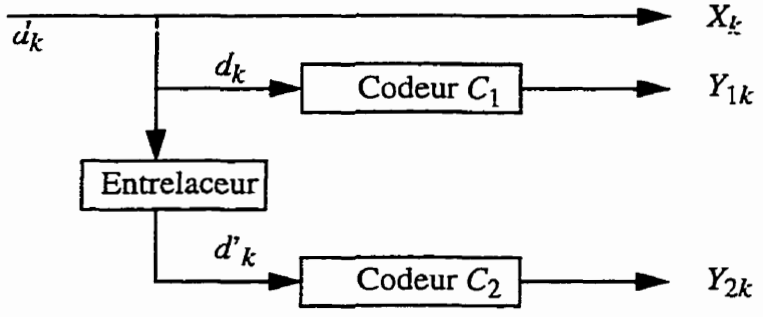


Figure A.4.1: Codeur CPCC

Tableau A.4.1: Spectre des codes CPCC: R= 1/3, , $3 \leq K \leq 9$, $N = 16$

K	G	d_f	(n) [w]
3	(1, 7/5)	6	(1, 4, 12, 37, 114, 351, 1081, 3328, 10199, 30594, 86937, 225072, 513784, 1012077, 1699492, 2417156, 2901315, 2930736, 2483477) [2, 10, 36, 124, 422, 1424, 4768, 15860, 52258, 167782, 507250, 1386016, 3309536, 6760006, 11677222, 16963096, 20660674, 21048106, 17880350]
4	(1, 17/15)	7	(1, 7, 25, 77, 252, 853, 2863, 9351, 28980, 82571, 210790, 475172, 941898) [4, 25, 86, 289, 1074, 3988, 14353, 49810, 162664, 484627, 1287332, 3012695, 6195251]
5	(1, 35/23)	8	(2, 14, 55, 184, 612, 2013, 6309, 18427, 49447, 120601, 265641, 527459) [8, 59, 235, 781, 2703, 9749, 33558, 104865, 294208, 743079, 1695783, 3502967]
6	(1, 75/53)	9	(1, 15, 109, 505357, 1793, 4949, 11972, 277455, 63300, 138085, 277455, 11953743) [4, 62, 458, 2192, 7722, 22114, 57194, 144592, 359536, 843378, 1803202, 3472302]
7	(1, 171/133)	11	(11, 110, 555, 1957, 5604, 14091, 32232, 68050, 133501, 243734) [60, 621, 3264, 11950, 35050, 88987, 204744, 438552, 885385, 1683394]
8	(1, 371/247)	11	(1, 14, 104, 544, 2191, 7041, 18483, 40623, 77095, 131093) [6, 76, 525, 2679, 10997, 36944, 102452, 238935, 483353, 880702]
9	(1, 753/561)	13	(11, 125, 686, 2490, 6983, 16673, 35447, 67275, 112918, 167975) [67, 763, 4199, 15338, 43632, 106808, 234796, 462760, 808841, 1257405]

Tableau A.4.2: Spectre des codes CPCC: R= 1/3, , $3 \leq K \leq 9$, $N = 32$

K	G	d_f	(n) [w]
3	(1, 7/5)	6	(1, 4, 12, 37, 114, 351, 1081, 3329, 10252, 31572, 97229, 299426, 922111, 2839729, 8745217, 26931731, 82938675, 255409989, 786375942) [2, 10, 36, 124, 422, 1424, 4768, 15862, 52478, 172788, 566538, 1850710, 6025882, 19562594, 63340840, 204598480, 659441584, 2121186950, 6809472250]
4	(1, 17/15)	7	(1, 7, 25, 77, 252, 853, 2876, 9644, 32325, 108429, 363778, 1220376, 4093856) [4, 25, 86, 289, 1074, 3988, 14447, 51867, 185799, 663524, 2360431, 8368183, 29580992]
5	(1, 35/23)	8	(2, 14, 55, 184, 617, 2120, 7455, 26751, 96448, 344878, 1222999, 4327683) [8, 59, 235, 781, 2734, 10409, 40645, 156996, 595660, 2231393, 8317056, 30990000]
6	(1, 75/53)	9	(1, 15, 109, 518, 1828, 5460, 16562, 57802, 222285, 863278, 3260304, 11905519) [4, 62, 458, 2196, 7896, 24892, 83536, 324504, 1347342, 5521300, 21801458, 83072534]
7	(1, 171/133)	11	(11, 110, 568, 2188, 7688, 27038, 96029, 341752, 1226068, 4464182) [60, 621, 3340, 13309, 47555, 169133, 615886, 2288709, 8674537, 33461818]
8	(1, 371/247)	11	(1, 14, 107, 612, 2948, 12625, 49709, 185117, 669531, 2400818) [6, 76, 545, 3144, 16284, 76590, 326960, 1289762, 4856362, 18017652]
9	(1, 753/561)	13	(11, 125, 718, 3062, 12123, 48542, 194263, 766332, 2978414, 11337004) [67, 763, 4424, 19355, 79920, 335073, 1400834, 5765589, 23332468, 92161440]

Tableau A.4.3: Spectre des codes CPCC: R= 1/3, , $3 \leq K \leq 9$, $N = 64$

K	G	d_f	(n) [w]
3	(1, 7/5)	6	(1, 4, 12, 37, 114, 351, 1081, 3329, 10252, 31572, 97229, 299426, 922111, 2839729, 8745217, 26931732, 82938844, 255418101, 786584466) [2, 10, 36, 124, 422, 1424, 4768, 15862, 52478, 172788, 566538, 1850710, 6025882, 19562594, 63340840, 204598482, 659442404, 2121238456, 6811021446]
4	(1, 17/15)	7	(1, 7, 25, 77, 252, 853, 2876, 9644, 32325, 108429, 363778, 1220376, 4093856) [4, 25, 86, 289, 1074, 3988, 14447, 51867, 185799, 663524, 2360431, 8368183, 29580992]
5	(1, 35/23)	8	(2, 14, 55, 184, 617, 2120, 7455, 26751, 96448, 344881, 1223112, 4329989) [8, 59, 235, 781, 2734, 10409, 40645, 156996, 595660, 2231417, 8317968, 31009128]
6	(1, 75/53)	9	(1, 15, 109, 518, 1828, 5460, 16562, 57802, 222285, 863346, 3262865, 11953743) [4, 62, 458, 2196, 7896, 24892, 83536, 324504, 1347342, 5521866, 21822924, 83480414]
7	(1, 171/133)	11	(11, 110, 568, 2188, 7688, 27038, 96035, 341946, 1229446, 4506285) [60, 621, 3340, 13309, 47555, 169133, 615942, 2290525, 8706251, 33857983]
8	(1, 371/247)	11	(1, 14, 107, 612, 2948, 12625, 49710, 185166, 670682, 2418352) [6, 76, 545, 3144, 16284, 76590, 326968, 1290175, 4866410, 18174760]
9	(1, 753/561)	13	(11, 125, 718, 3062, 12125, 48606, 195320, 778451, 3088116, 12180903) [67, 763, 4424, 19355, 79934, 335521, 1408379, 5855380, 24187177, 99115316]

Tableau A.4.4: Spectre des codes CPCC: R= 1/3, , $3 \leq K \leq 9$, $N = 128$

K	G	d_f	(n) [w]
3	(1, 7/5)	6	(1, 4, 12, 37, 114, 351, 1081, 3329, 10252, 31572, 97229, 299426, 922111, 2839729, 8745217, 26931732, 82938844, 255418101, 786584466) [2, 10, 36, 124, 422, 1424, 4768, 15862, 52478, 172788, 566538, 1850710, 6025882, 19562594, 63340840, 204598482, 659442404, 2121238456, 6811021446]
4	(1, 17/15)	7	(1, 7, 25, 77, 252, 853, 2876, 9644, 32325, 108429, 363778, 1220376, 4093856) [4, 25, 86, 289, 1074, 3988, 14447, 51867, 185799, 663524, 2360431, 8368183, 29580992]
5	(1, 35/23)	8	(2, 14, 55, 184, 617, 2120, 7455, 26751, 96448, 344881, 1223112, 4329989) [8, 59, 235, 781, 2734, 10409, 40645, 156996, 595660, 2231417, 8317968, 31009128]
6	(1, 75/53)	9	(1, 15, 109, 518, 1828, 5460, 16562, 57802, 222285, 863346, 3262865, 11953743) [4, 62, 458, 2196, 7896, 24892, 83536, 324504, 1347342, 5521866, 21822924, 83480414]
7	(1, 171/133)	11	(11, 110, 568, 2188, 7688, 27038, 96035, 341946, 1229446, 4506285) [60, 621, 3340, 13309, 47555, 169133, 615942, 2290525, 8706251, 33857983]
8	(1, 371/247)	11	(1, 14, 107, 612, 2948, 12625, 49710, 185166, 670682, 2418352) [6, 76, 545, 3144, 16284, 76590, 326968, 1290175, 4866410, 18174760]
9	(1, 753/561)	13	(11, 125, 718, 3062, 12125, 48606, 195320, 778451, 3088116, 12180903) [67, 763, 4424, 19355, 79934, 335521, 1408379, 5855380, 24187177, 99115316]

Tableau A.4.5: Spectre des codes CPCC: R= 1/3, , $3 \leq K \leq 9$, N = 192

K	G	d_f	(n) [w]
3	(1, 7/5)	6	(1, 4, 12, 37, 114, 351, 1081, 3329, 10252, 31572, 97229, 299426, 922111, 2839729, 8745217, 26931732, 82938844, 255418101, 786584466) [2, 10, 36, 124, 422, 1424, 4768, 15862, 52478, 172788, 566538, 1850710, 6025882, 19562594, 63340840, 204598482, 659442404, 2121238456, 6811021446]
4	(1, 17/15)	7	(1, 7, 25, 77, 252, 853, 2876, 9644, 32325, 108429, 363778, 1220376, 4093856) [4, 25, 86, 289, 1074, 3988, 14447, 51867, 185799, 663524, 2360431, 8368183, 29580992]
5	(1, 35/23)	8	(2, 14, 55, 184, 617, 2120, 7455, 26751, 96448, 344881, 1223112, 4329989) [8, 59, 235, 781, 2734, 10409, 40645, 156996, 595660, 2231417, 8317968, 31009128]
6	(1, 75/53)	9	(1, 15, 109, 518, 1828, 5460, 16562, 57802, 222285, 863346, 3262865, 11953743) [4, 62, 458, 2196, 7896, 24892, 83536, 324504, 1347342, 5521866, 21822924, 83480414]
7	(1, 171/133)	11	(11, 110, 568, 2188, 7688, 27038, 96035, 341946, 1229446, 4506285) [60, 621, 3340, 13309, 47555, 169133, 615942, 2290525, 8706251, 33857983]
8	(1, 371/247)	11	(1, 14, 107, 612, 2948, 12625, 49710, 185166, 670682, 2418352) [6, 76, 545, 3144, 16284, 76590, 326968, 1290175, 4866410, 18174760]
9	(1, 753/561)	13	(11, 125, 718, 3062, 12125, 48606, 195320, 778451, 3088116, 12180903) [67, 763, 4424, 19355, 79934, 335521, 1408379, 5855380, 24187177, 99115316]

ANNEXE V

PERFORMANCES DES CODES CONVOLUTIONNELS

Dans cette annexe, nous présentons des performances des codes convolutionnels non systématiques, des codes convolutionnels récursifs systématiques, des codes convolutionnels récursifs systématiques perforés, des codes CPCC, des codes CPCC perforés.

A.5.1 PERFORMANCES DES CODES CONVOLUTIONNELS NON SYSTÉMATIQUE ET DES CODES CONVOLUTIONNELS RÉCURSIFS SYSTÉMATIQUES

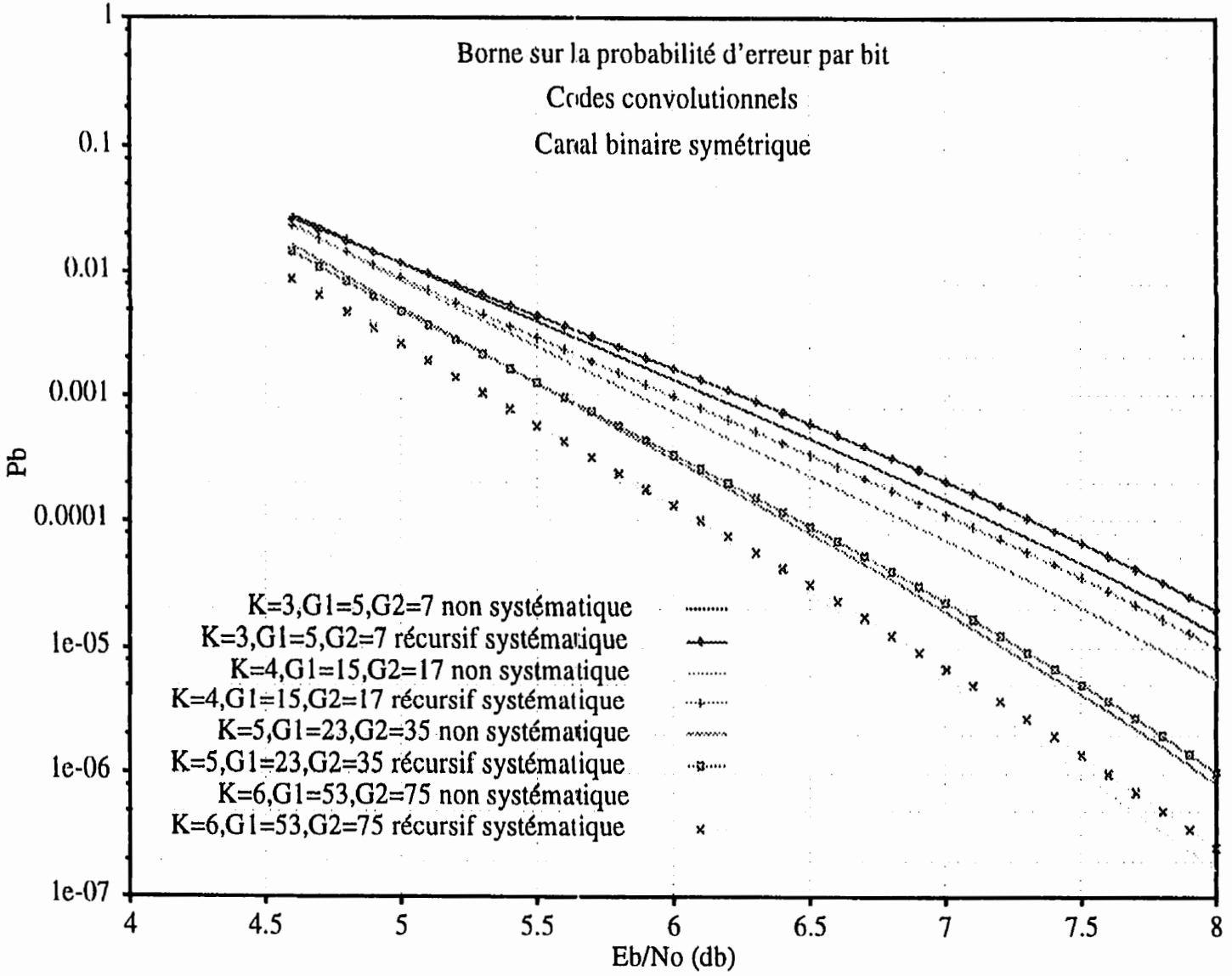


Figure A.5.1.1: Probabilité d'erreur des codes convolutionnels sous canal binaire symétrique, K = 3, 4, 5, 6

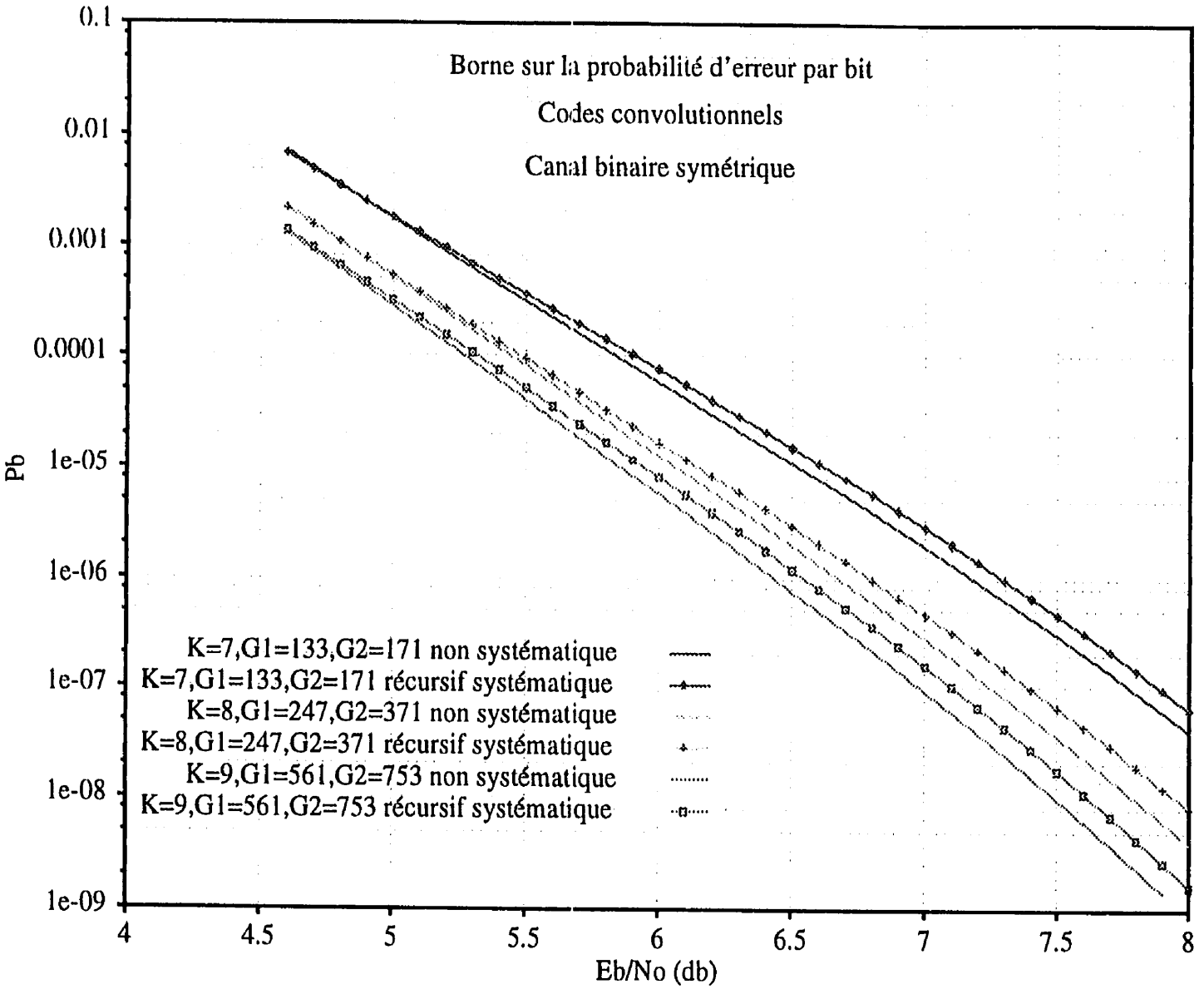


Figure A.5.1.2: Probabilité d'erreur des codes convolutionnels sous canal binaire symétrique, K = 7, 8, 9

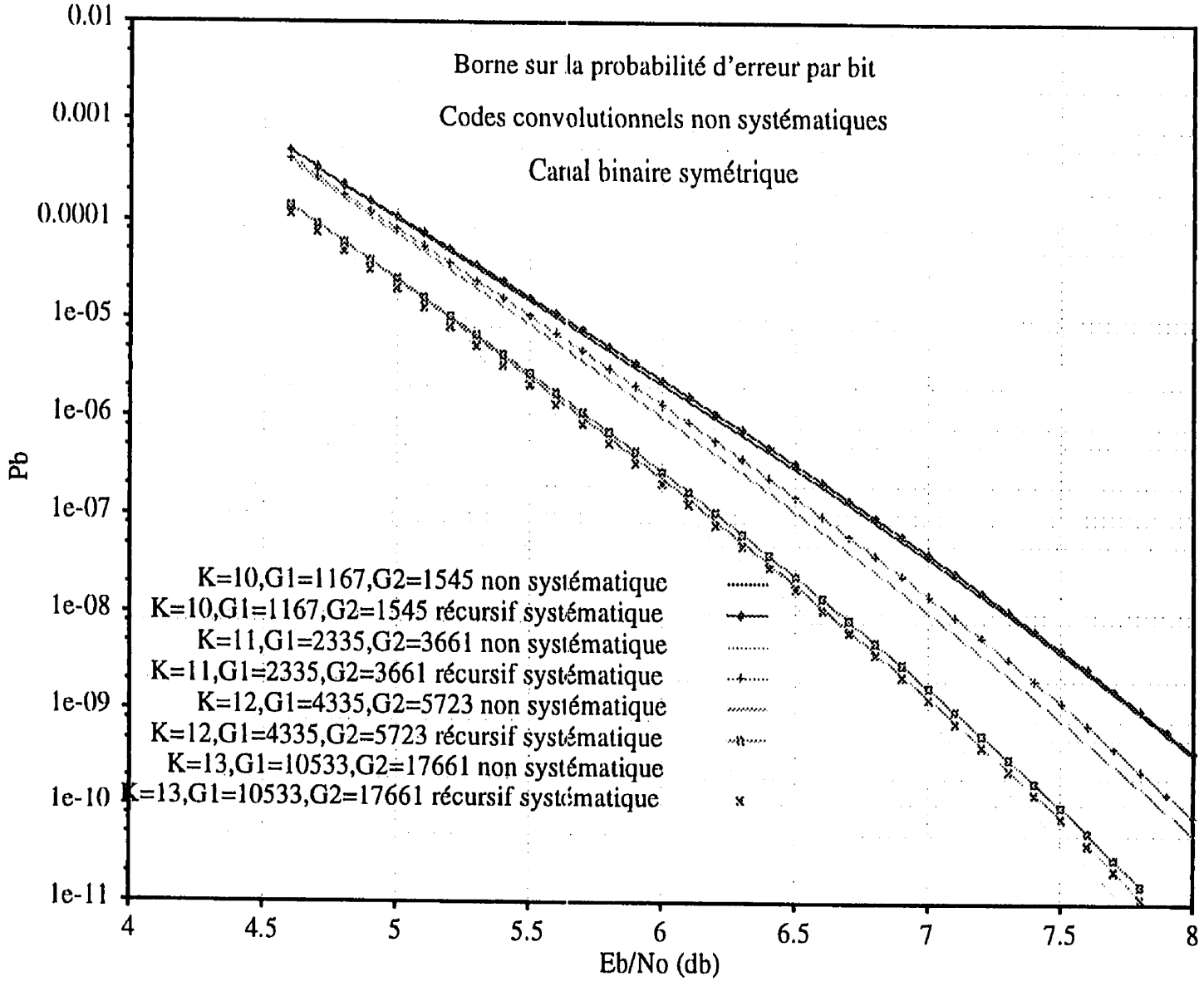


Figure A.5.1.3: Probabilité d'erreur des codes convolutionnels sous canal binaire symétrique, K = 10, 11, 12, 13

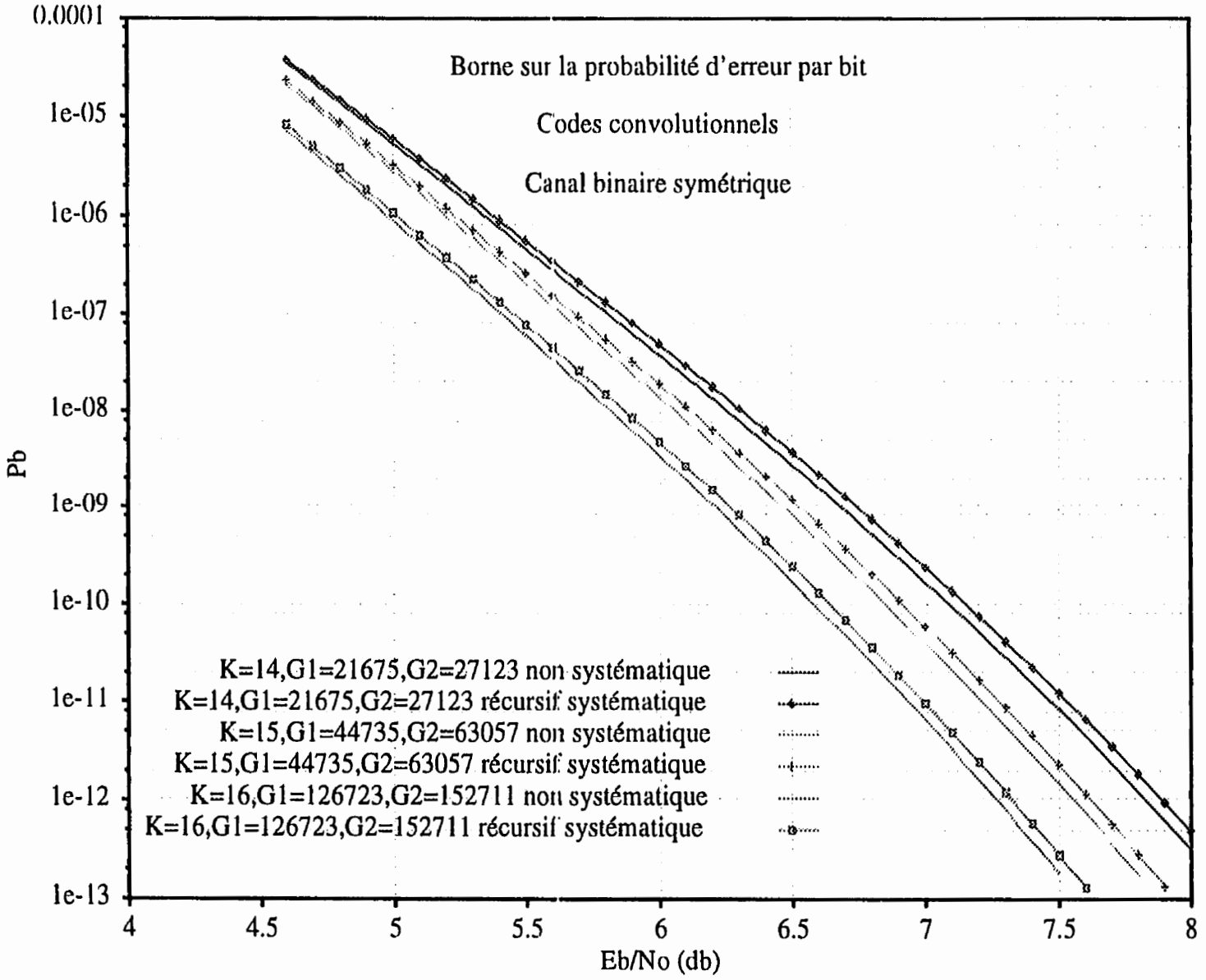


Figure A.5.1.4: Probabilité d'erreur des codes convolutionnels sous canal binaire symétrique, K = 14, 15, 16

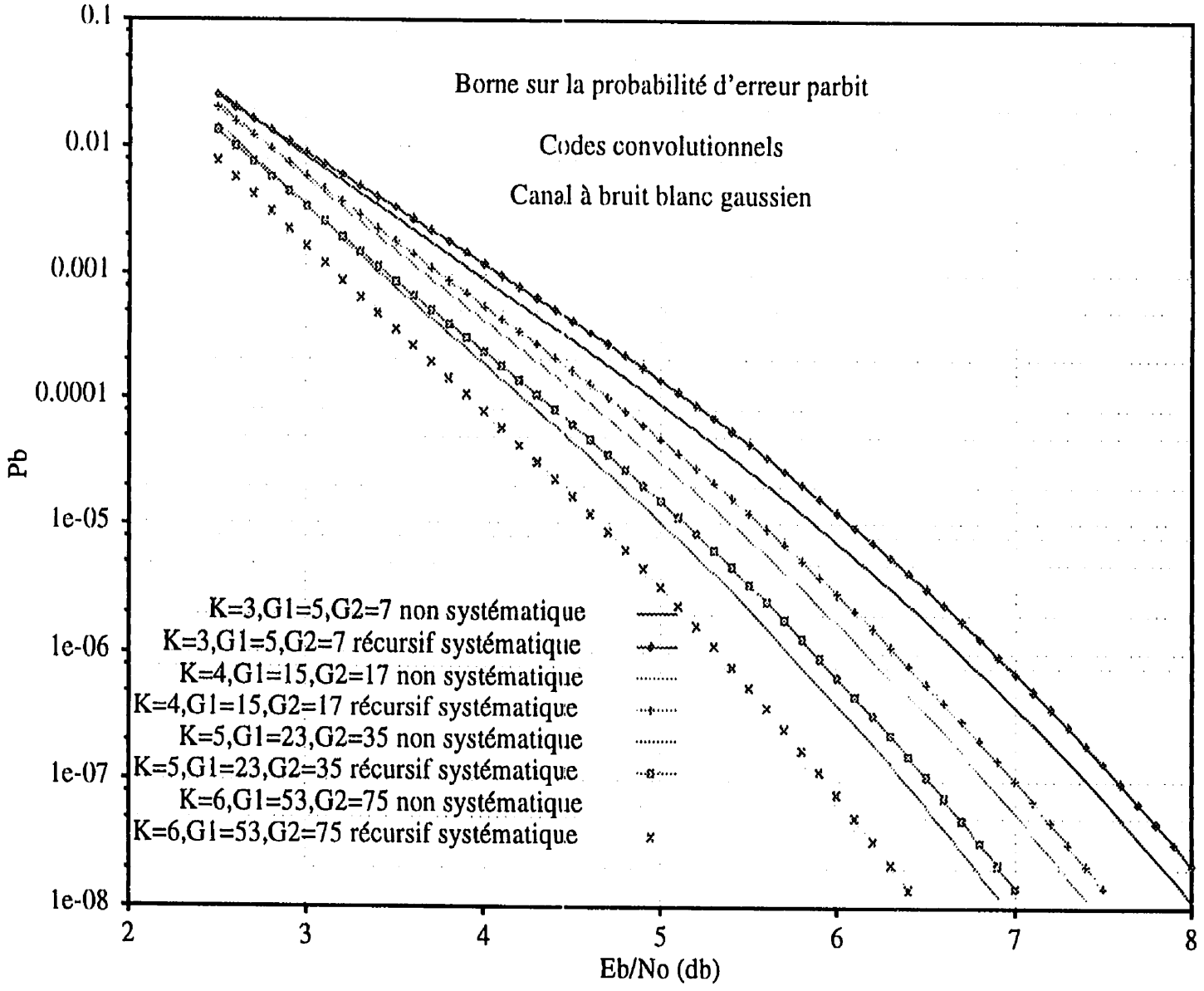


Figure A.5.1.5: Probabilité d'erreur des codes convolutionnels sous canal à bruit blanc gaussien, K = 3, 4, 5, 6

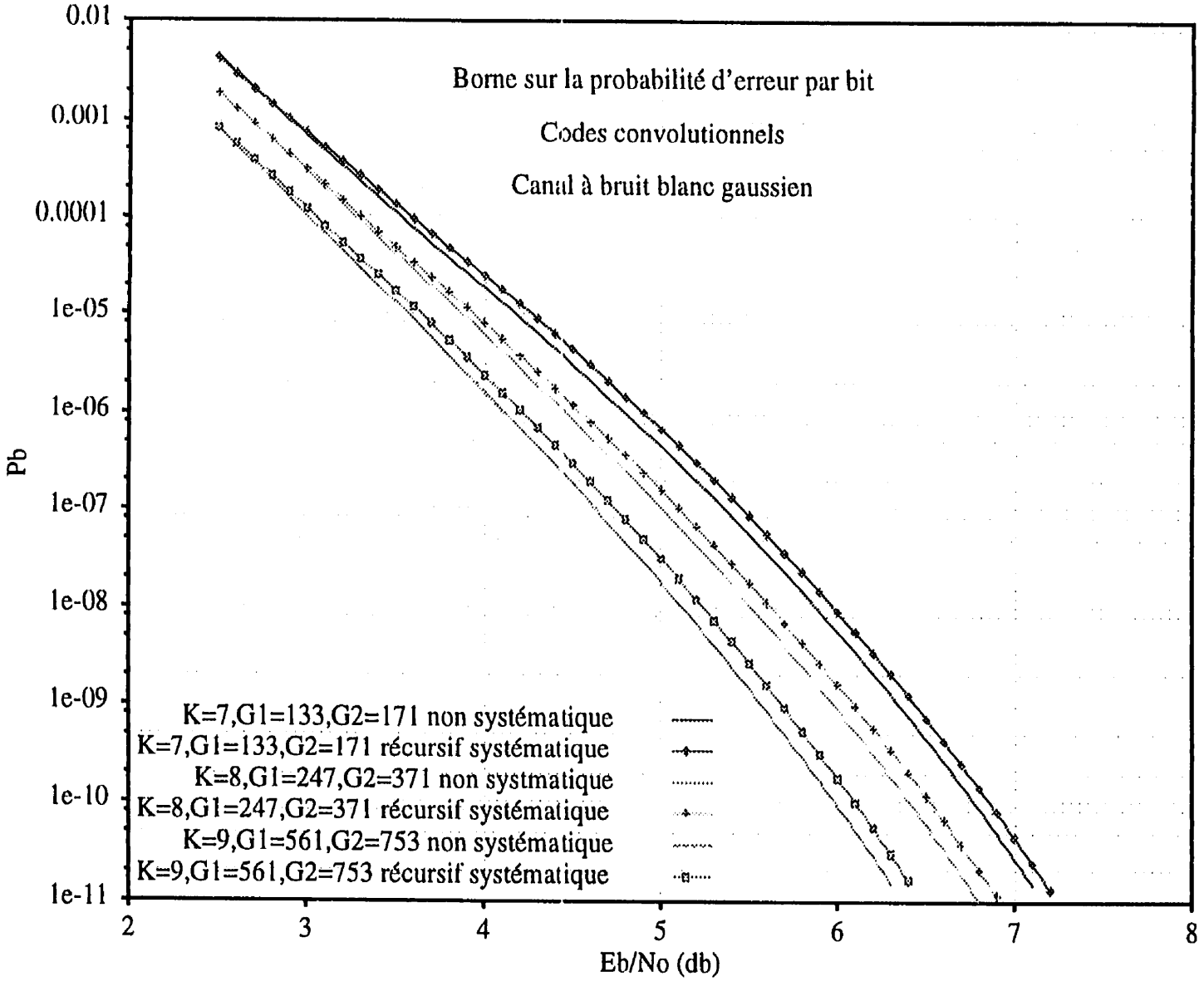


Figure A.5.1.6: Probabilité d'erreur des codes convolutionnels sous canal à bruit blanc gaussien, K = 7, 8, 9

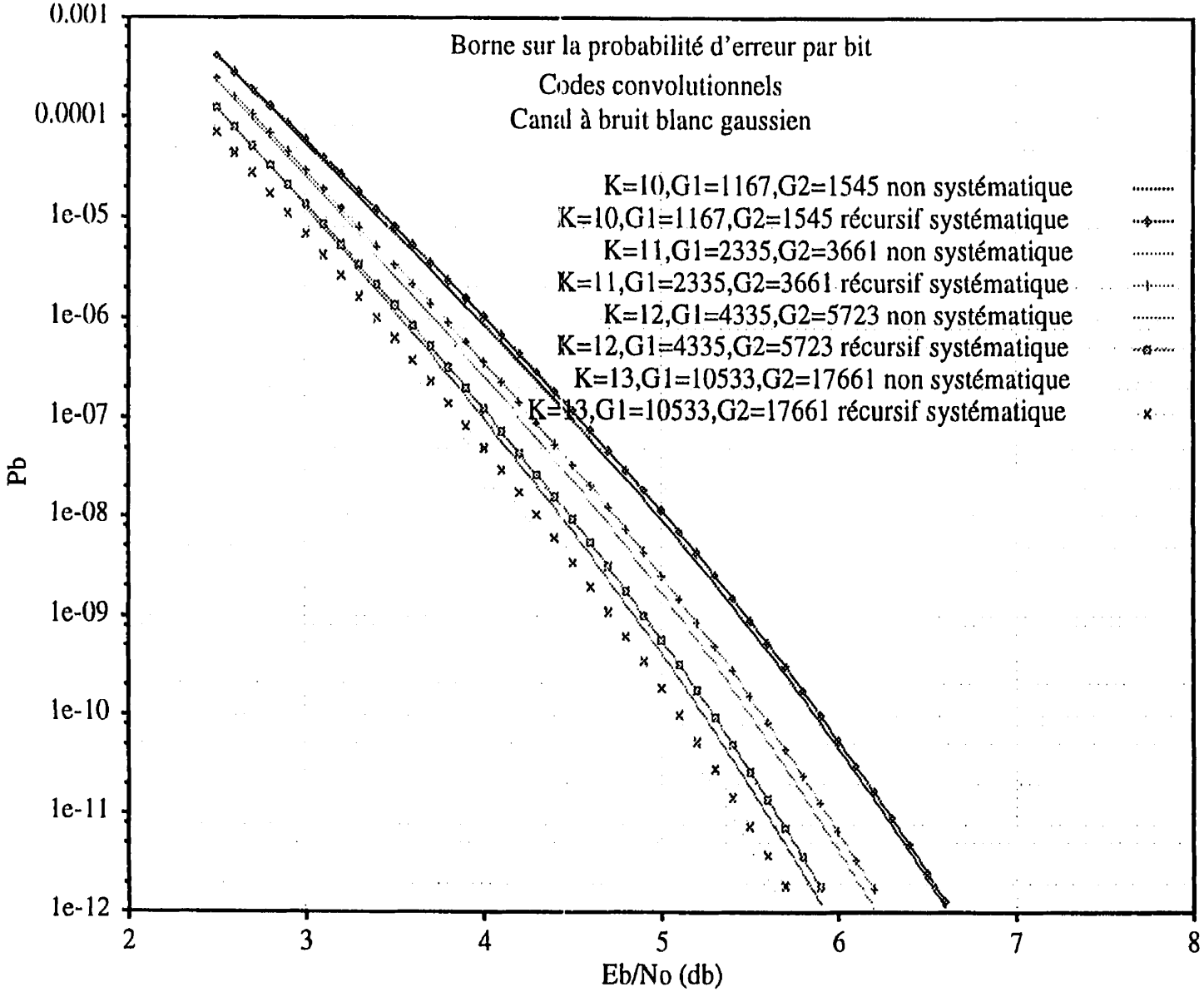


Figure A.5.1.7: Probabilité d'erreur des codes convolutionnels sous canal à bruit blanc gaussien, K = 10, 11, 12, 13

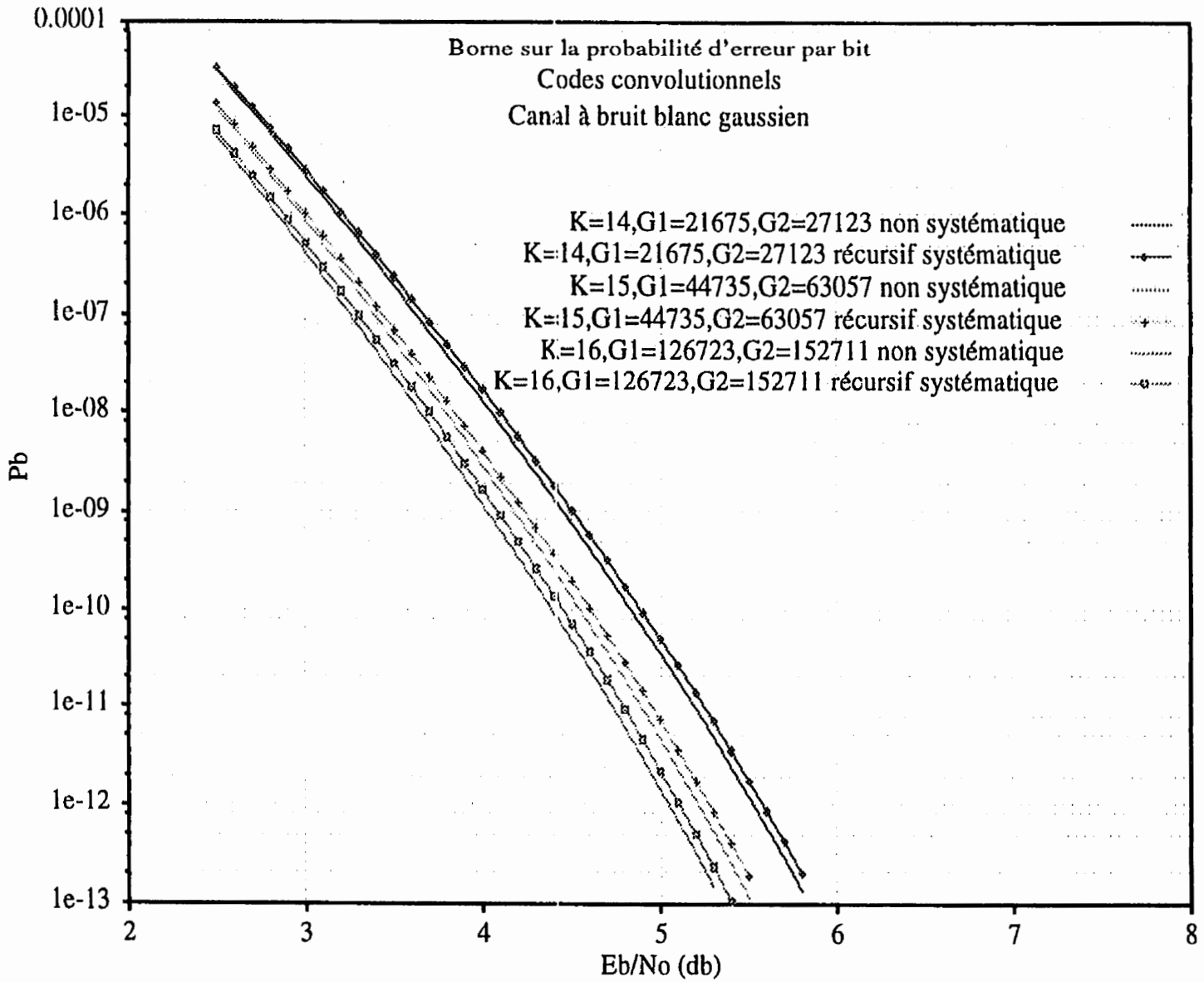


Figure A.5.1.8: Probabilité d'erreur des codes convolutionnels sous canal à bruit blanc gaussien, K = 14, 15, 16

A.5.2 PERFORMANCES DES CODES CONVOLUTIONNELS RÉCURSIFS SYSTÉMATIQUES PERFORÉS

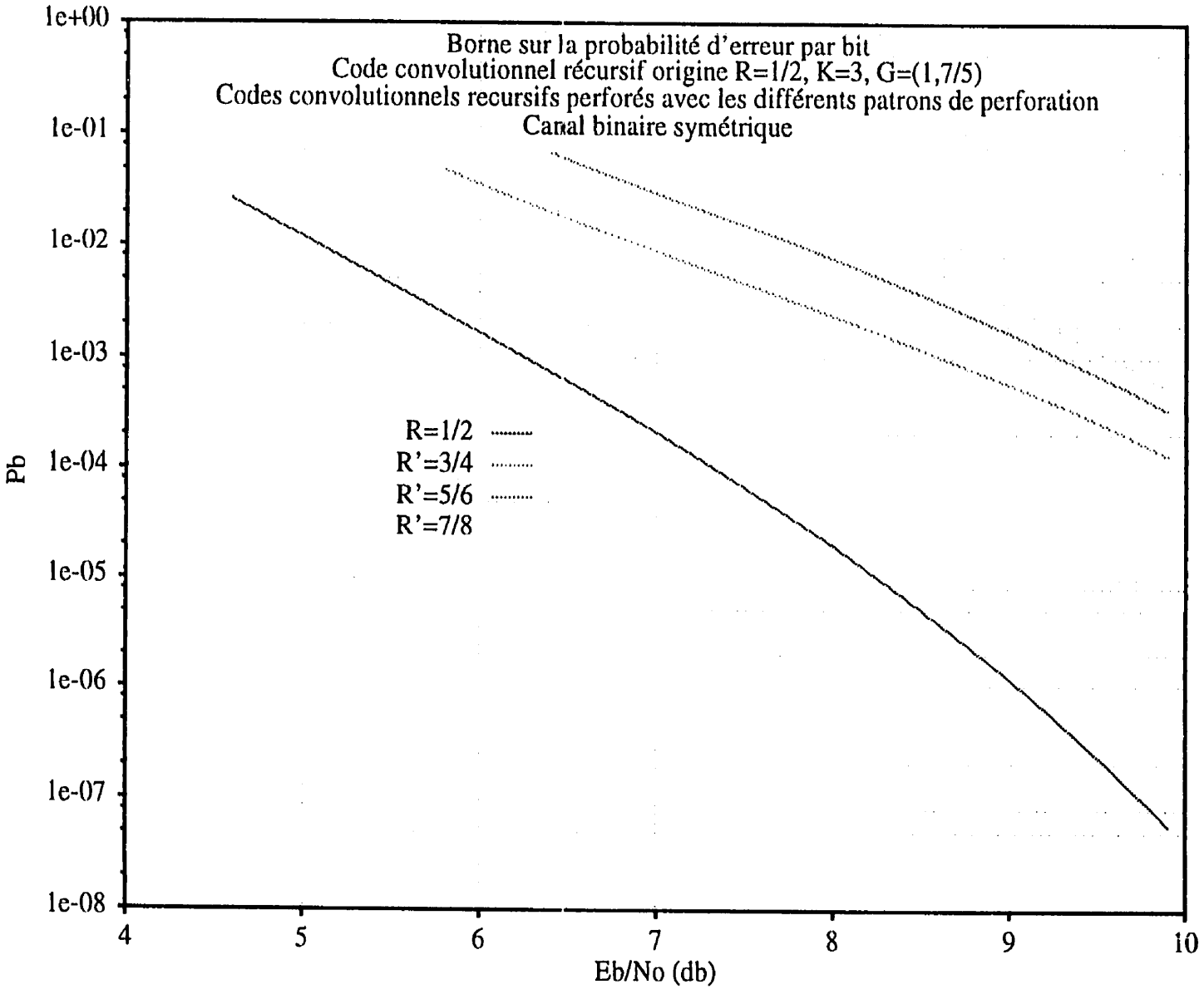


Figure A.5.2.1: Probabilité d'erreur des codes convolutionnels récursifs systématiques perforés sous canal binaire symétrique, K = 3, G = (1, 7/5), R = 1/2, 3/4, 5/6, 7/8

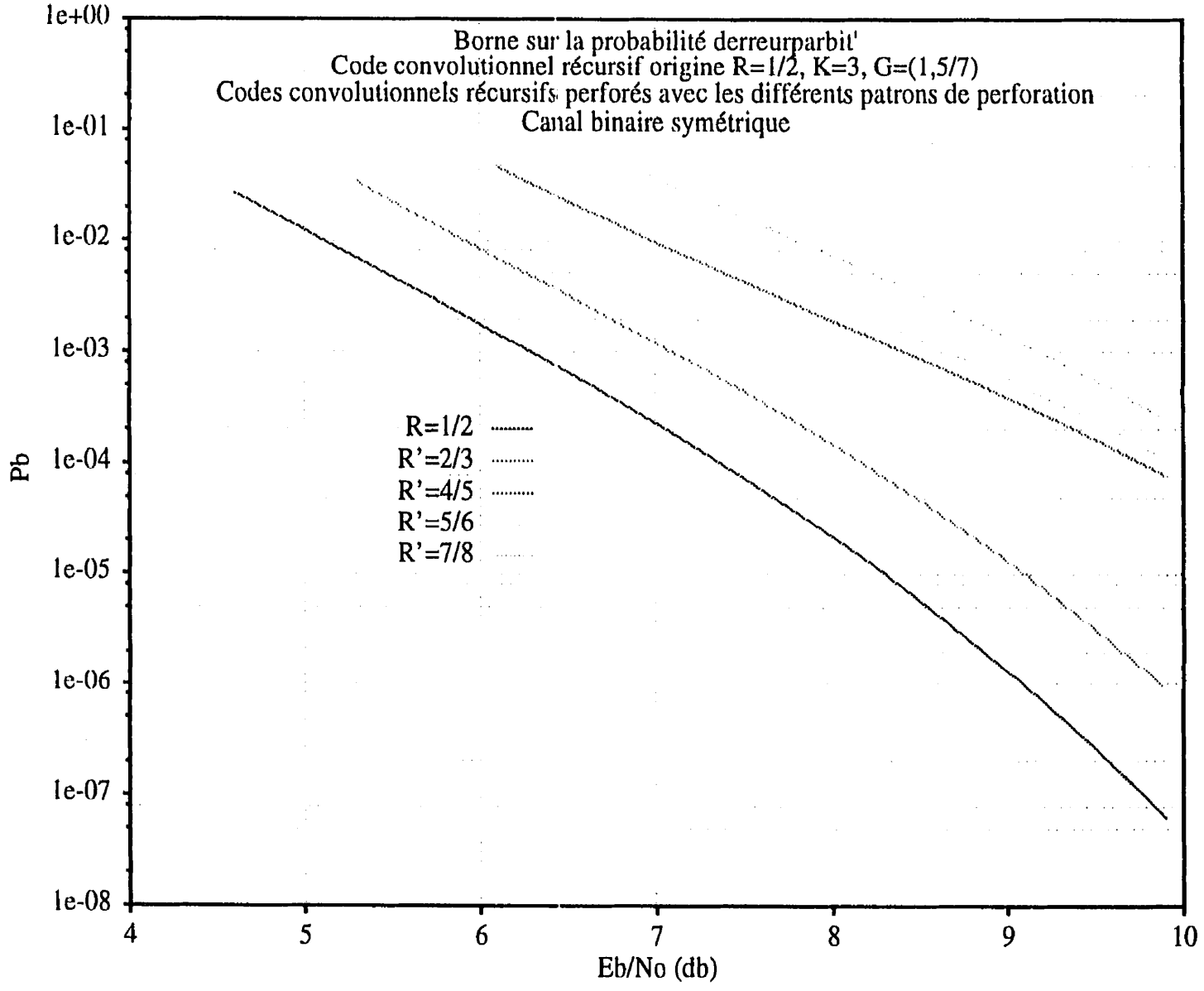


Figure A.5.2.2: Probabilité d'erreur des codes convolutionnels récursifs systématiques perforés sous canal binaire symétrique, K = 3, G = (1, 5/7), R = 1/2, 2/3, 4/5, 5/6, 7/8

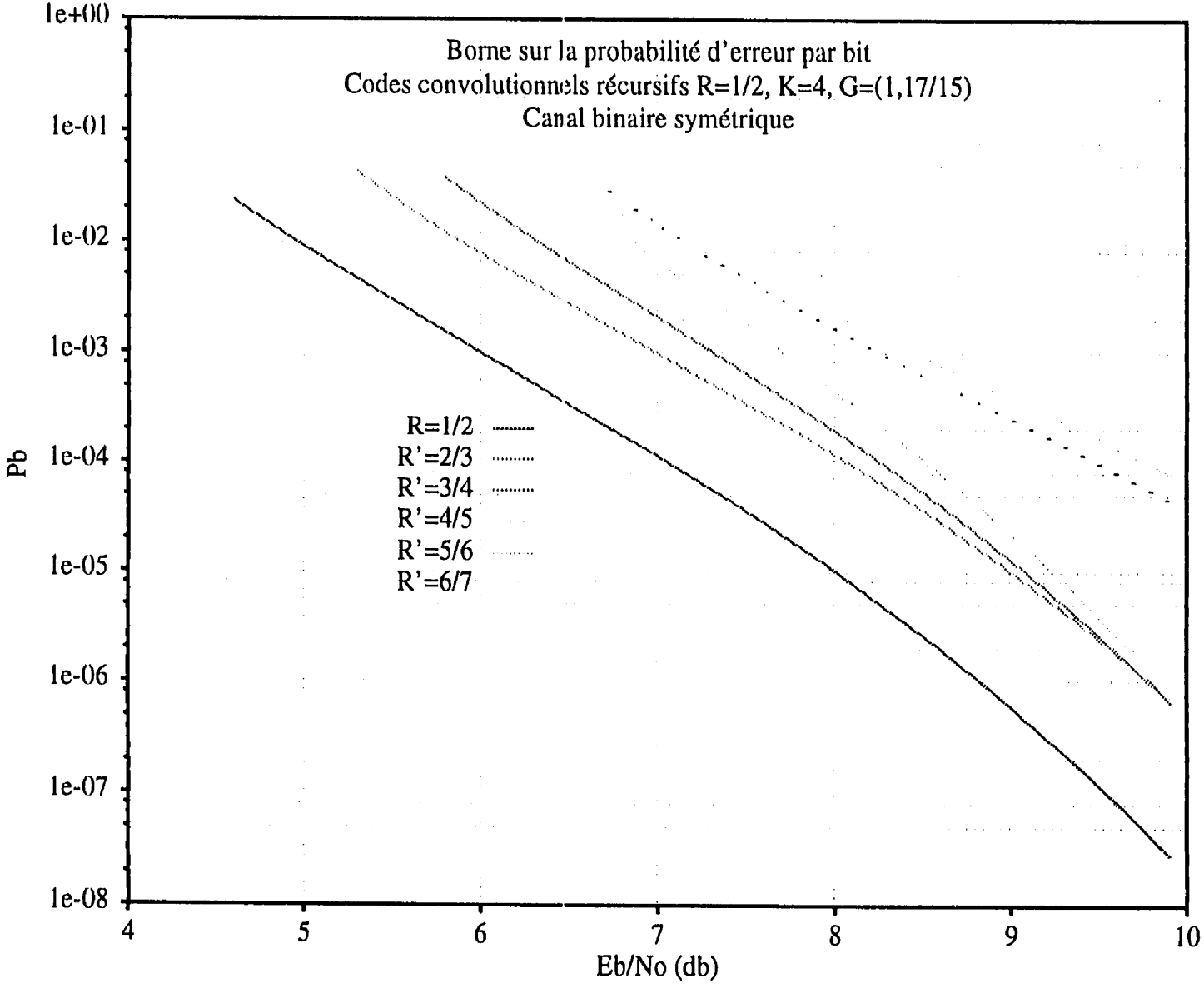


Figure A.5.2.3: Probabilité d'erreur des codes convolutionnels récursifs systématiques perforés sous canal binaire symétrique, K=4, G=(1, 17/15), R=1/2, 2/3, 3/4, 4/5, 5/6, 6/7

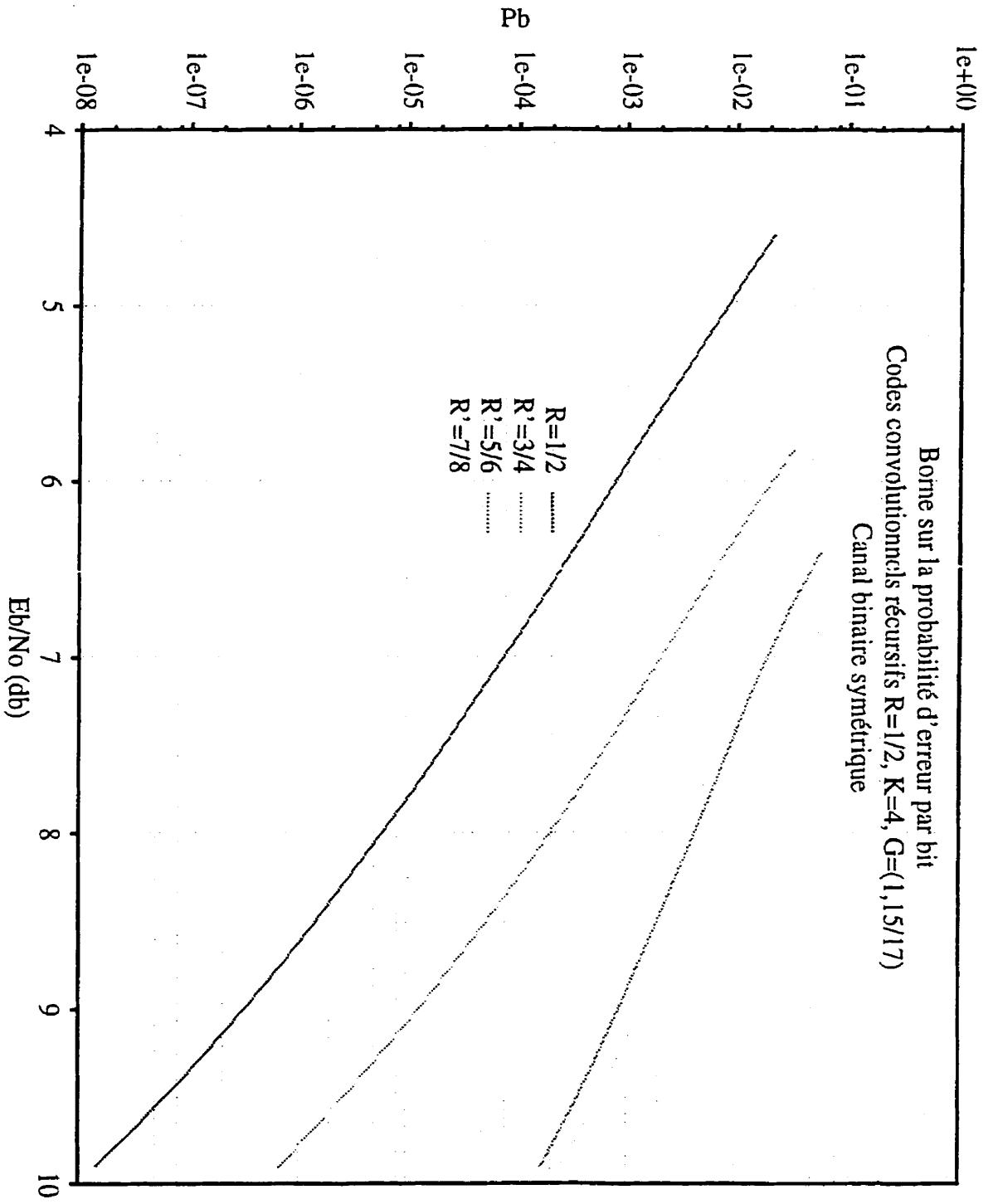


Figure A.5.2.4: Probabilité d'erreur des codes convolutionnels récursifs systématiques perforés sous canal binaire symétrique, K=4, G=(1, 15/17), R=1/2, 3/4, 5/6, 7/8

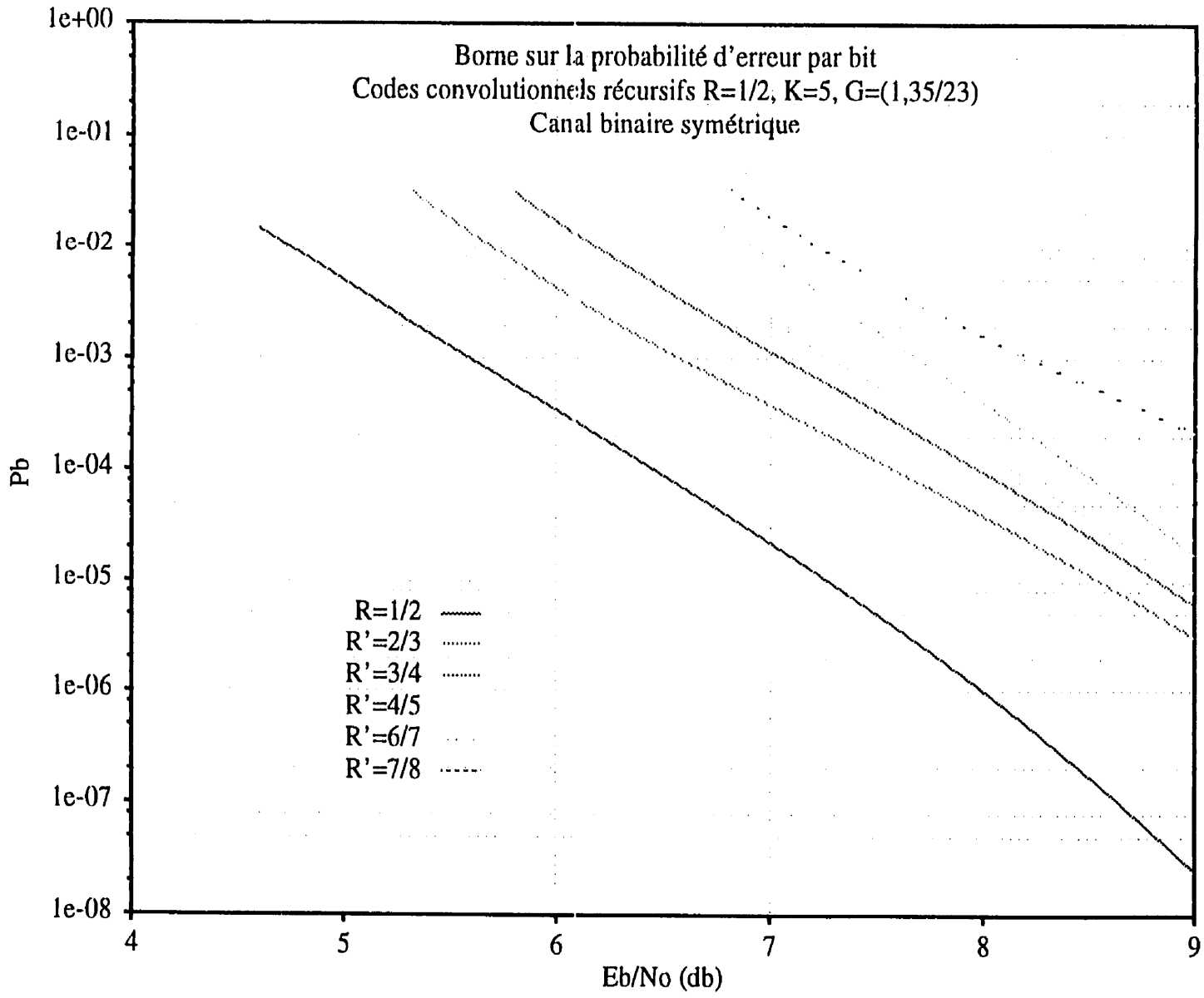


Figure A.5.2.5: Probabilité d'erreur des codes convolutionnels récursifs systématiques perforés sous canal binaire symétrique, K=5, G=(1, 35/23), R=1/2, 2/3, 3/4, 4/5, 6/7, 7/8

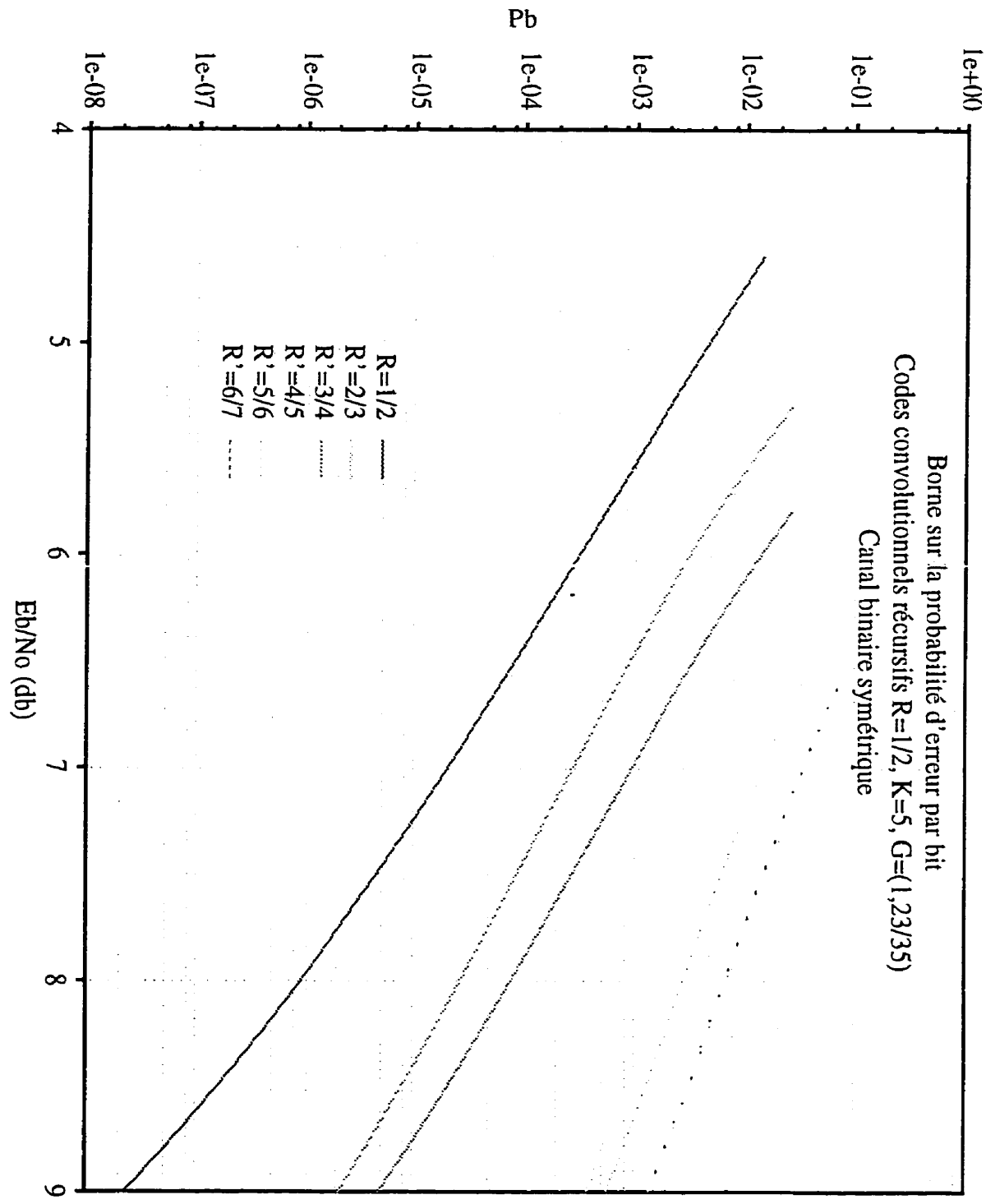


Figure A.5.2.6: Probabilité d'erreur des codes convolutionnels récursifs systématiques perforés sous canal binaire symétrique, K=5, G=(1, 23/35), R=1/2, 2/3, 3/4, 4/5, 5/6, 6/7

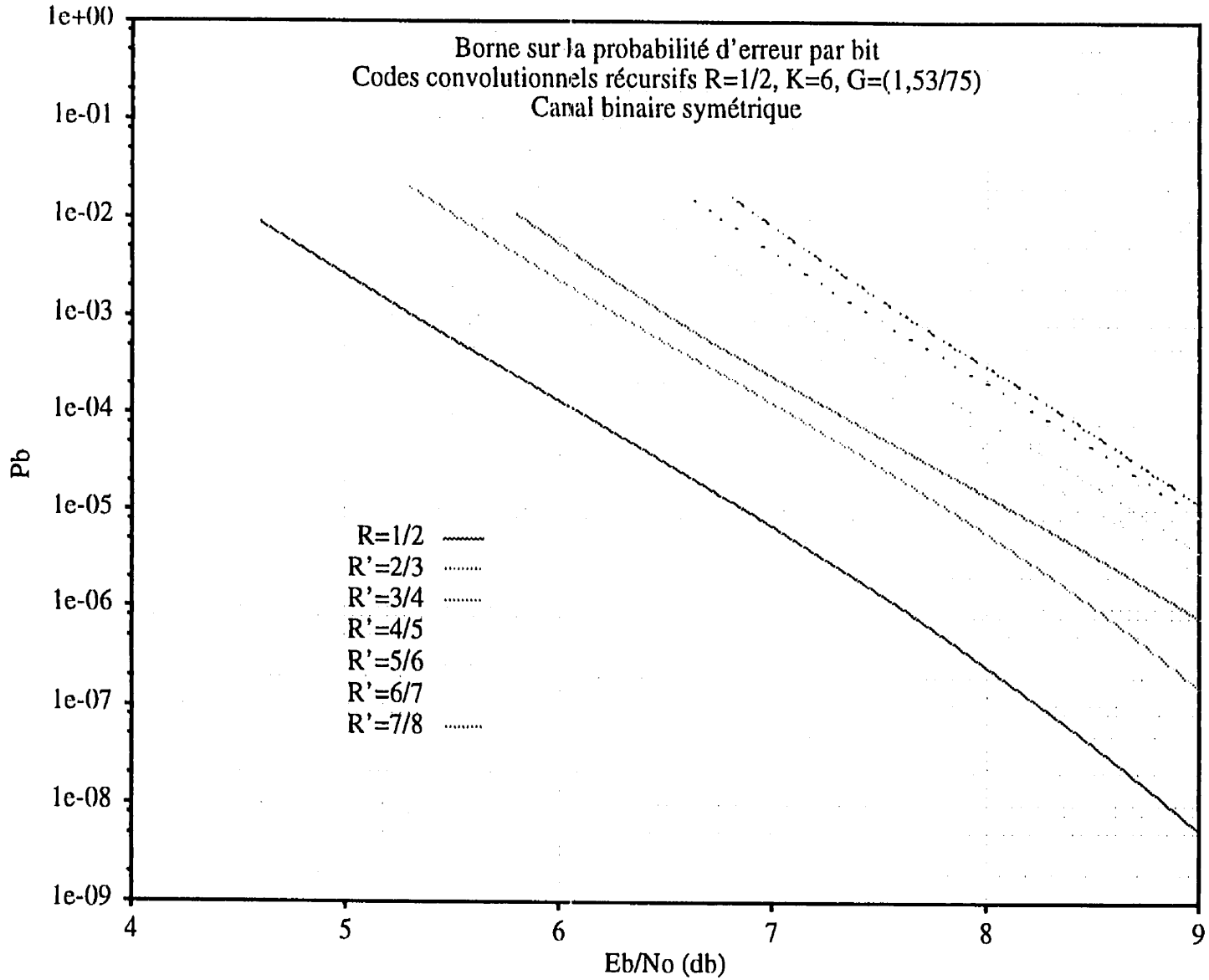


Figure A.5.2.7: Probabilité d'erreur des codes convolutionnels récursifs systématiques perforés sous canal binaire symétrique, K=6, G=(1,53/75), R=1/2,2/3,3/4,4/5,5/6,6/7,7/8

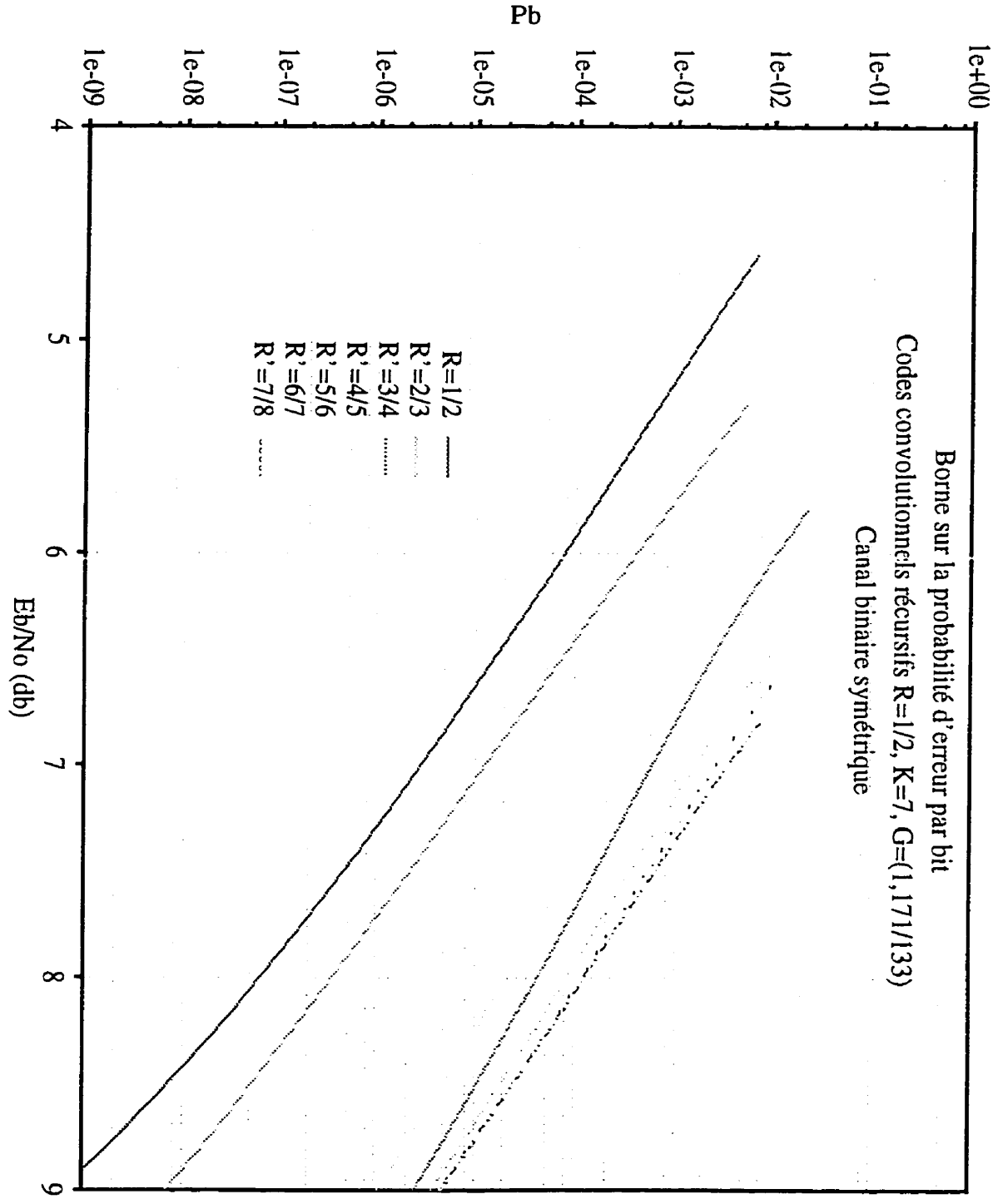


Figure A.5.2.8: Probabilité d'erreur des codes convolutionnels récursifs systématiques perforés sous canal binaire symétrique, K=7,G=(1,171/133),R=1/2,2/3,3/4,4/5,5/6,6/7,7/8

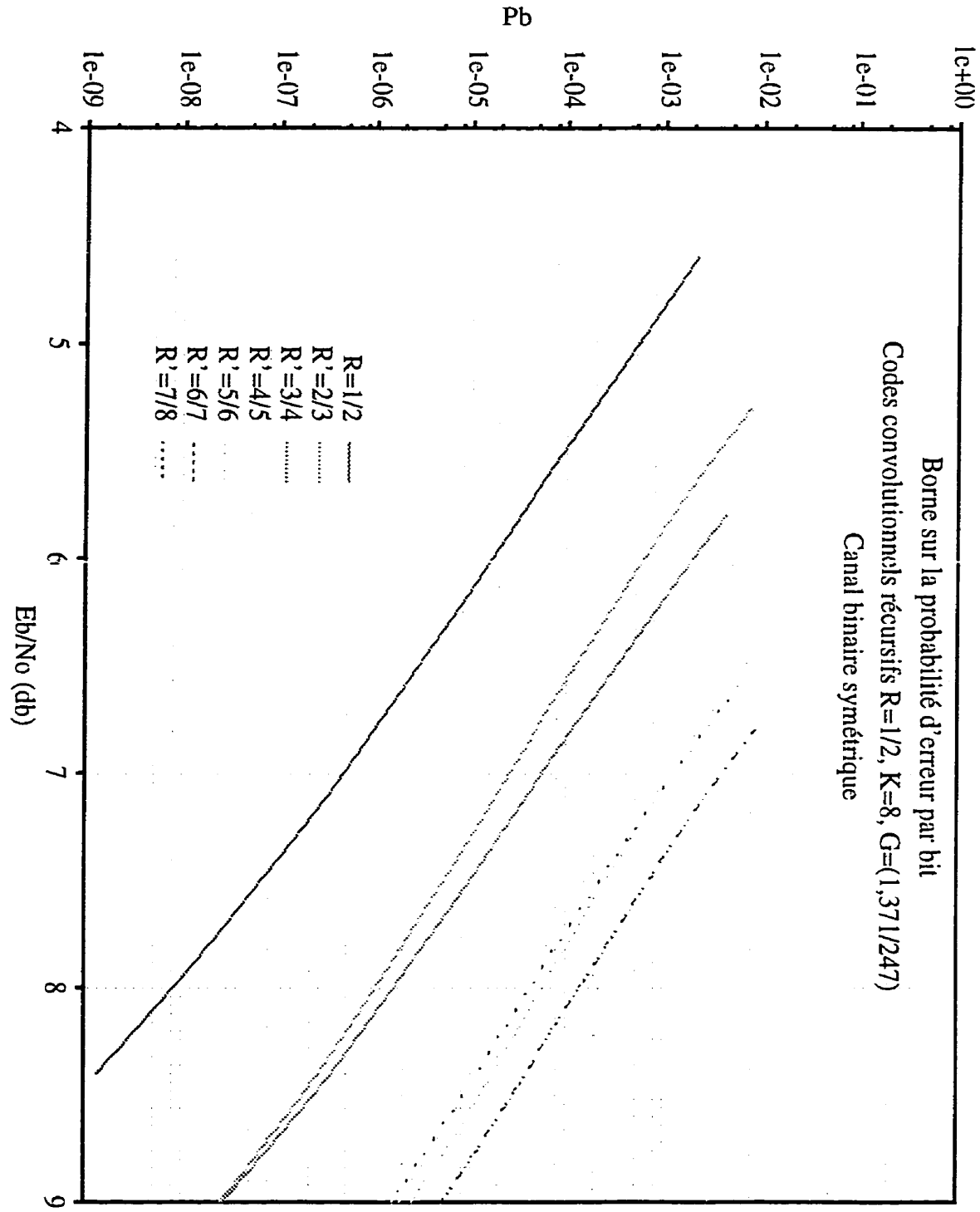


Figure A.5.2.9: Probabilité d'erreur des codes convolutionnels récursifs systématiques perforés sous canal binaire symétrique, K=8,G=(1,371/247),R=1/2,2/3,3/4,4/5,5/6,6/7,7/8

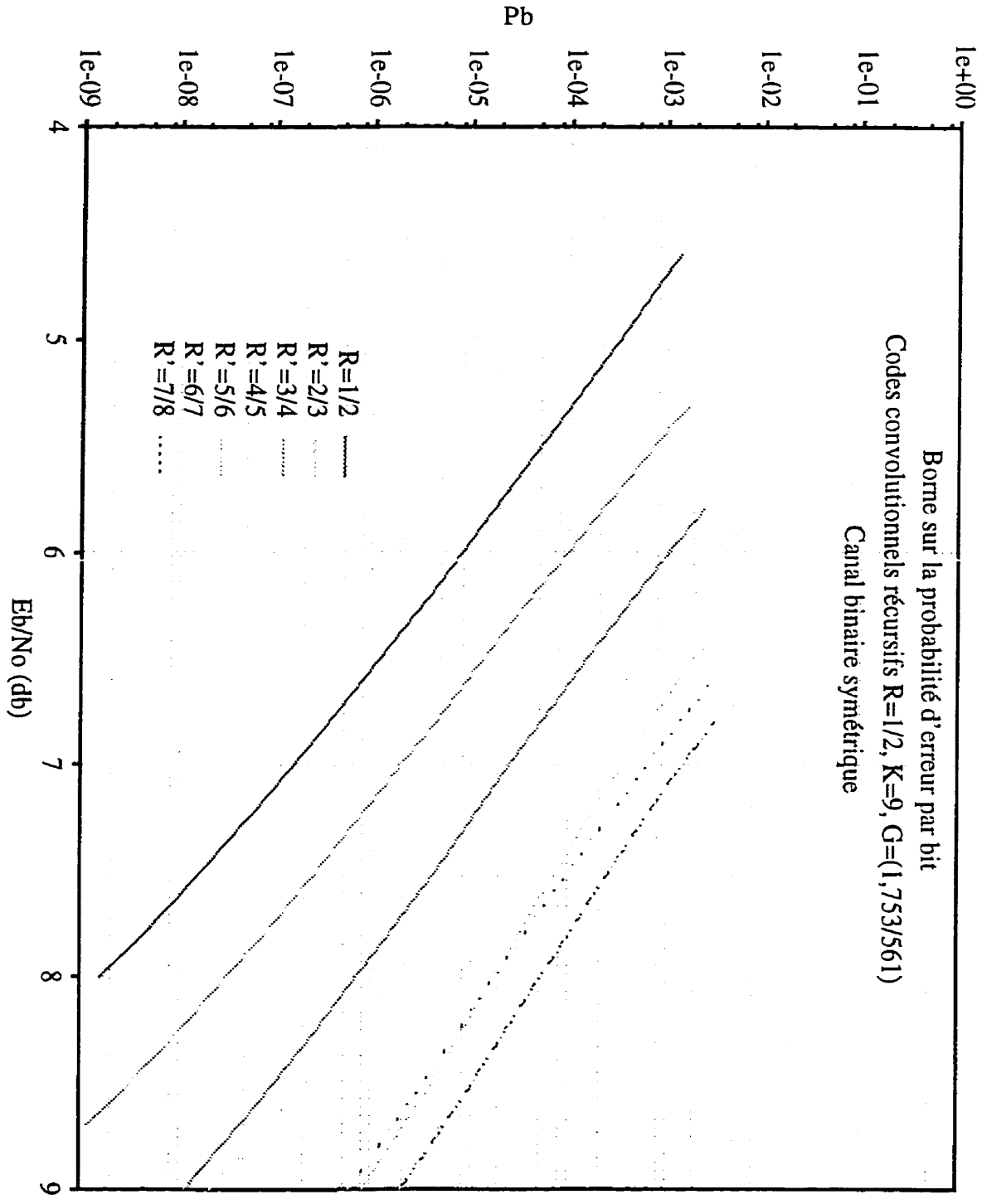


Figure A.5.2.10: Probabilité d'erreur des codes convolutionnels récursifs systématiques perforés sous canal binaire symétrique, K=9,G=(1,753/561),R=1/2,2/3,3/4,4/5,5/6,6/7,7/8

A.5.3 PERFORMANCES DES CODES CPCC

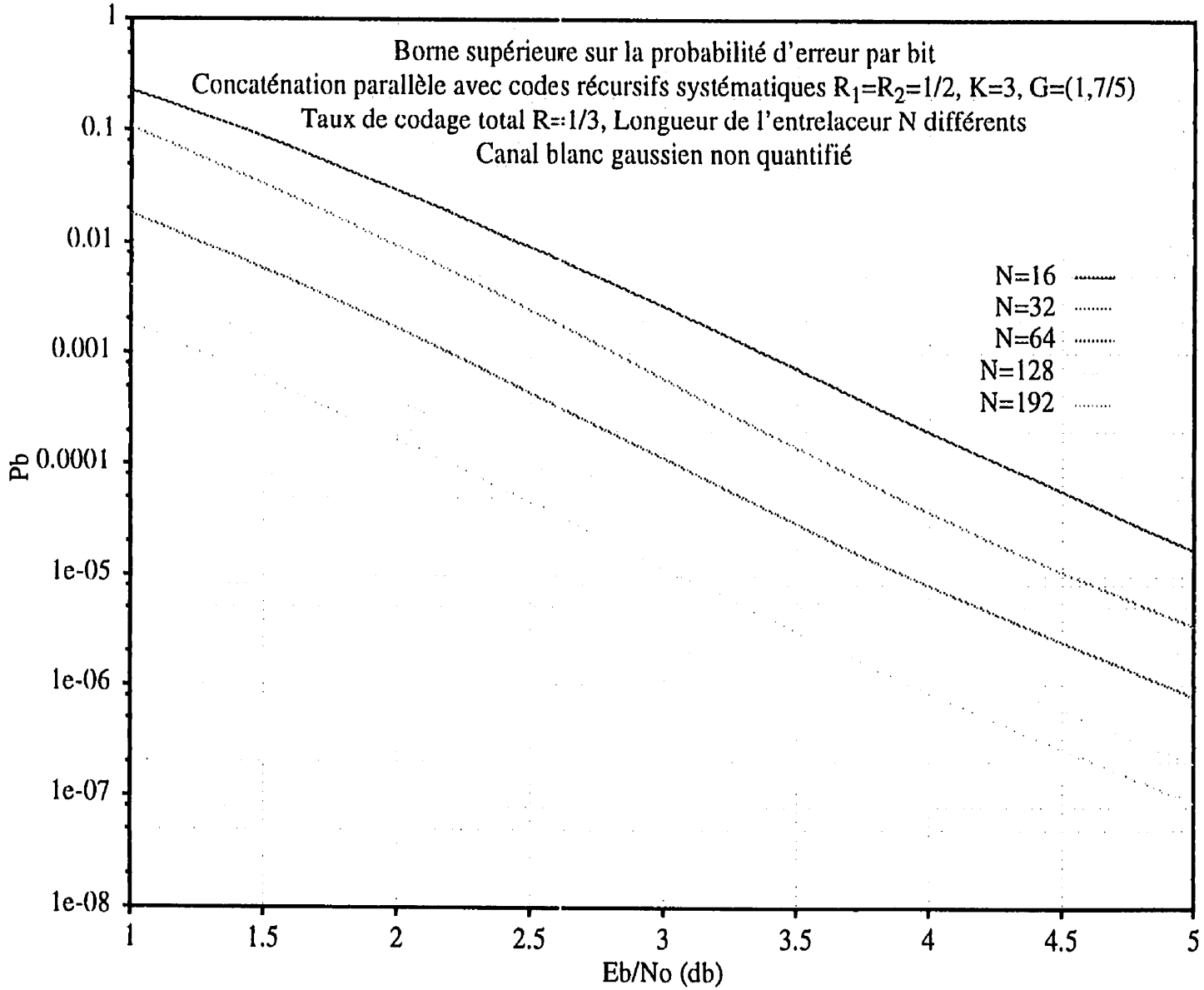


Figure A.5.3.1: Probabilités d'erreur du code CPCC avec deux codes récursifs systématiques $R_1 = R_2 = 1/2$, K = 3, G = (1, 7/5), différentes valeurs de *N*

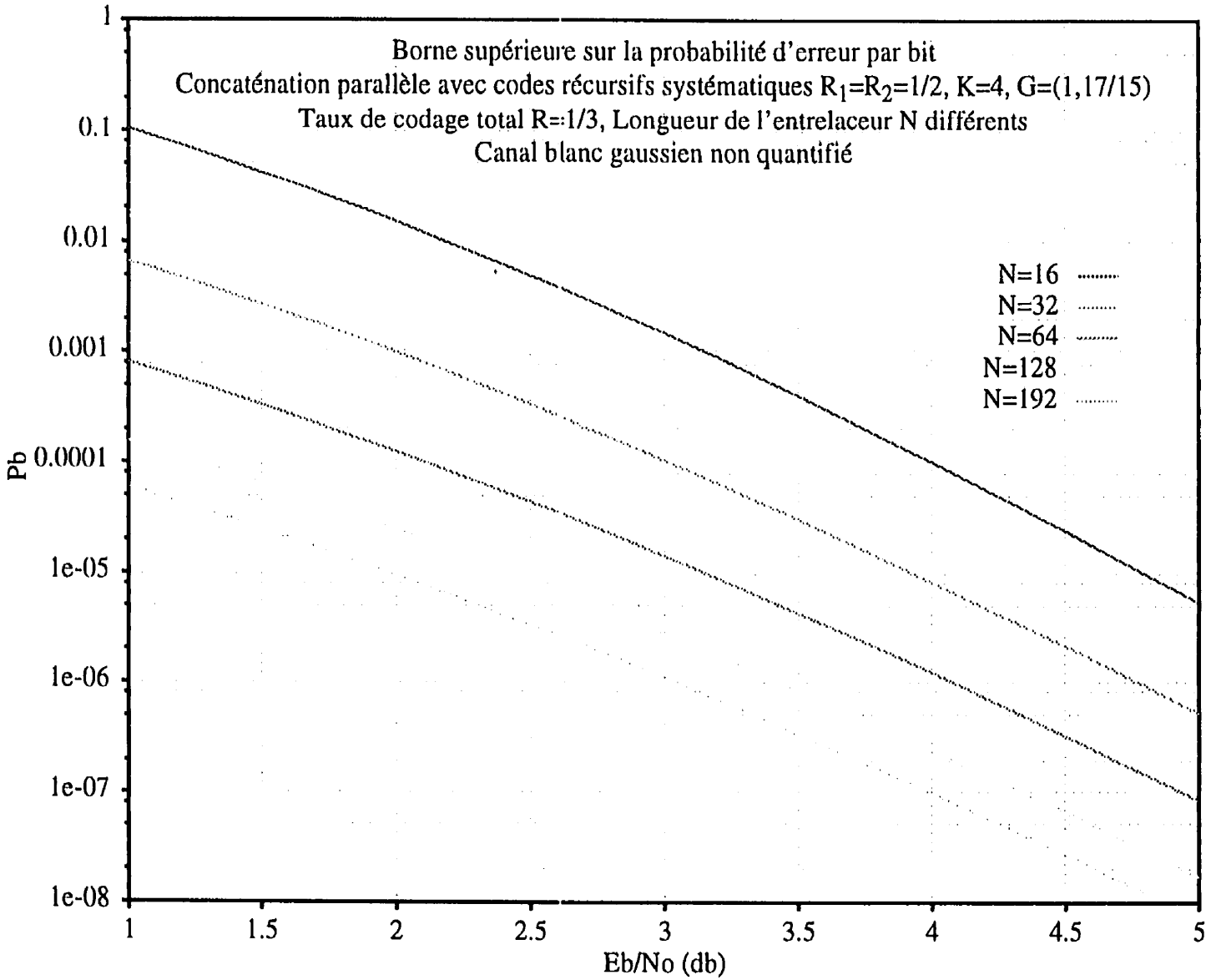


Figure A.5.3.2: Probabilités d'erreur du code CPCC avec deux codes récursifs systématiques $R_1 = R_2 = 1/2$, K = 4, G = (1, 17/15), différentes valeurs de *N*

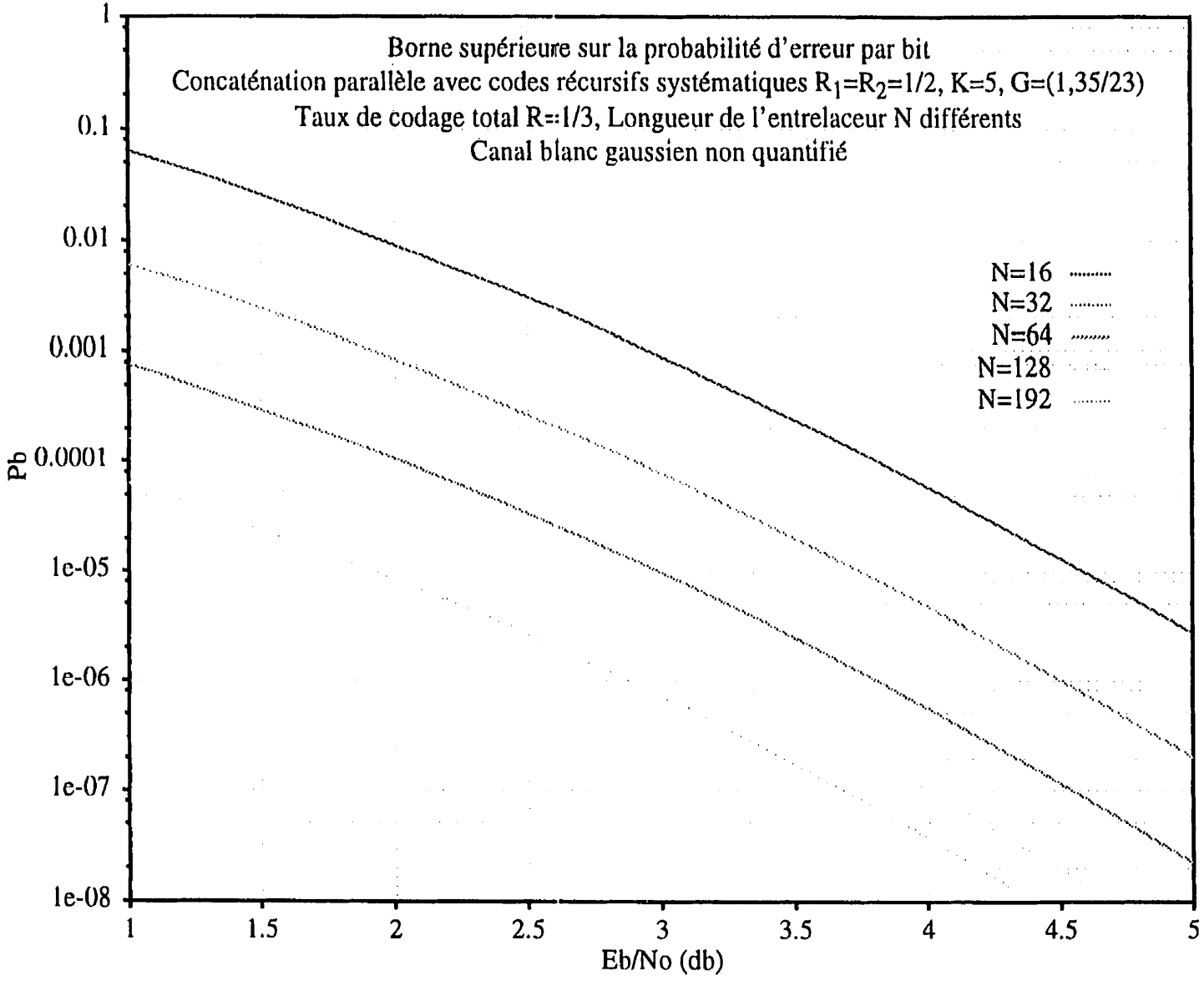


Figure A.5.3.3: Probabilités d'erreur du code CPCC avec deux codes récursifs systématiques $R_1 = R_2 = 1/2$, $K = 5$, $G = (1, 35/23)$, différentes valeurs de N

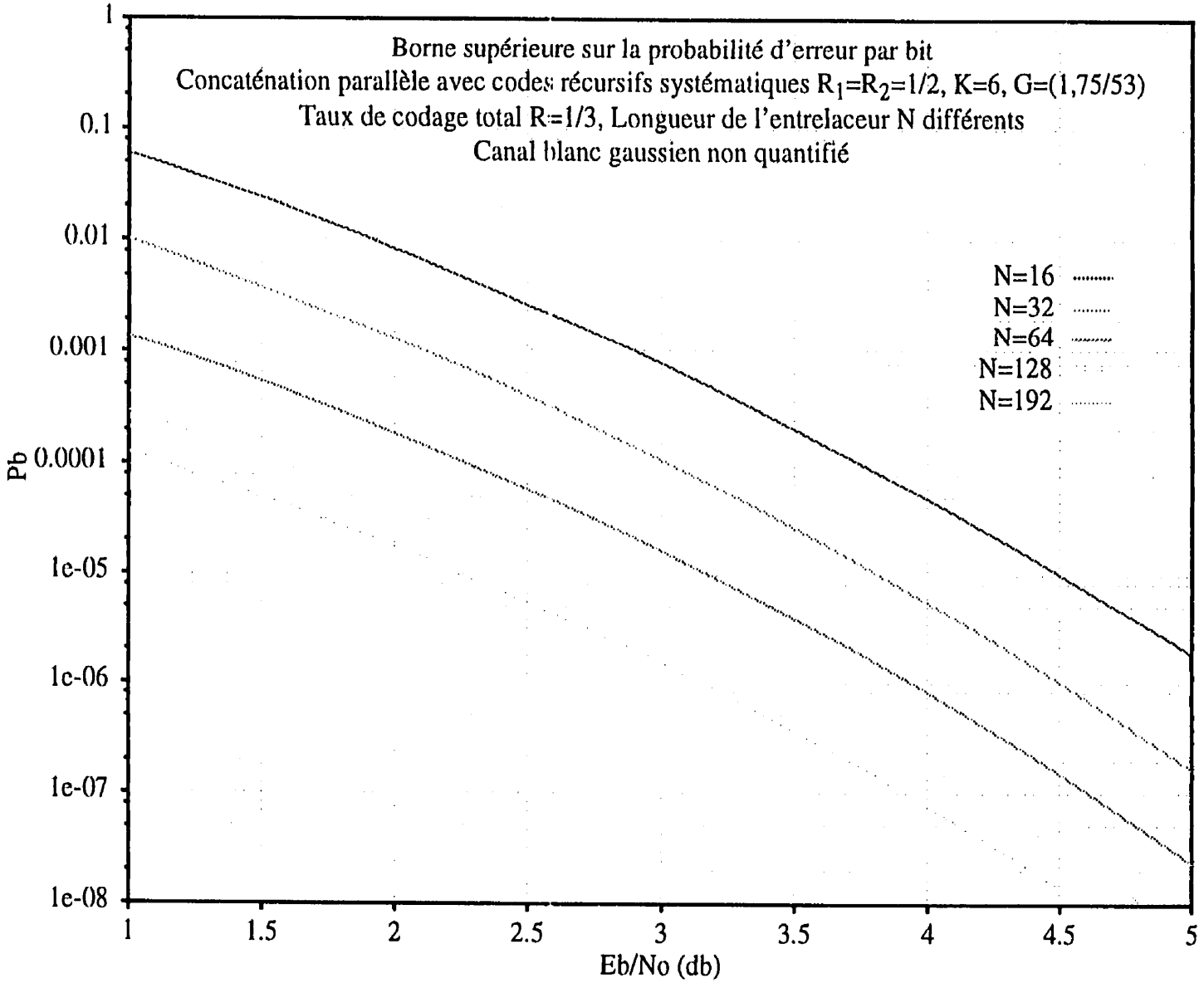


Figure A.5.3.4: Probabilités d'erreur du code CPCC avec deux codes récursifs systématiques $R_1 = R_2 = 1/2$, K = 6, G = (1, 75/53), différentes valeurs de *N*

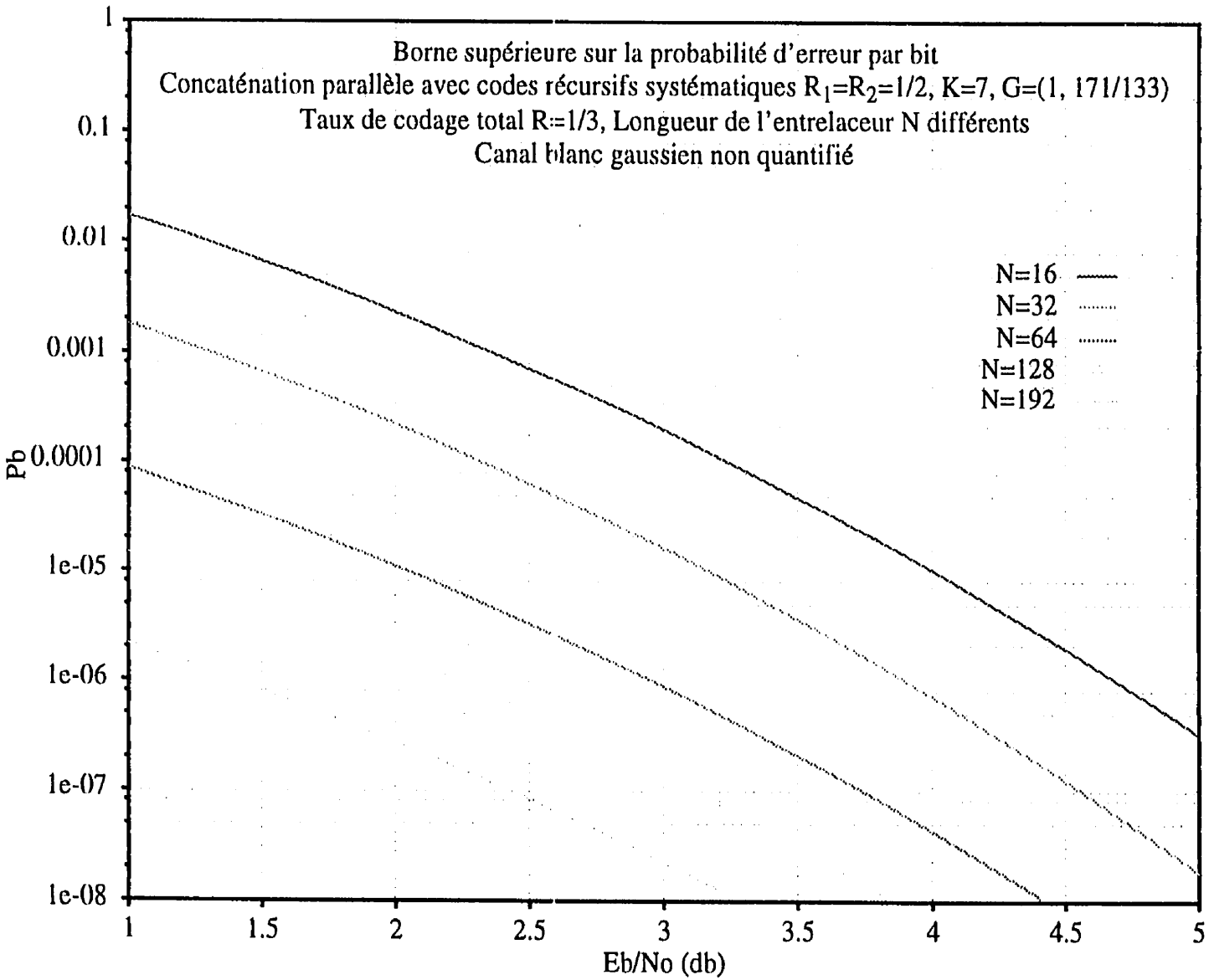


Figure A.5.3.5: Probabilités d'erreur du code CPCC avec deux codes récursifs systématiques.$R_1 = R_2 = 1/2$, K = 7, G = (1, 171/133), différentes valeurs de *N*

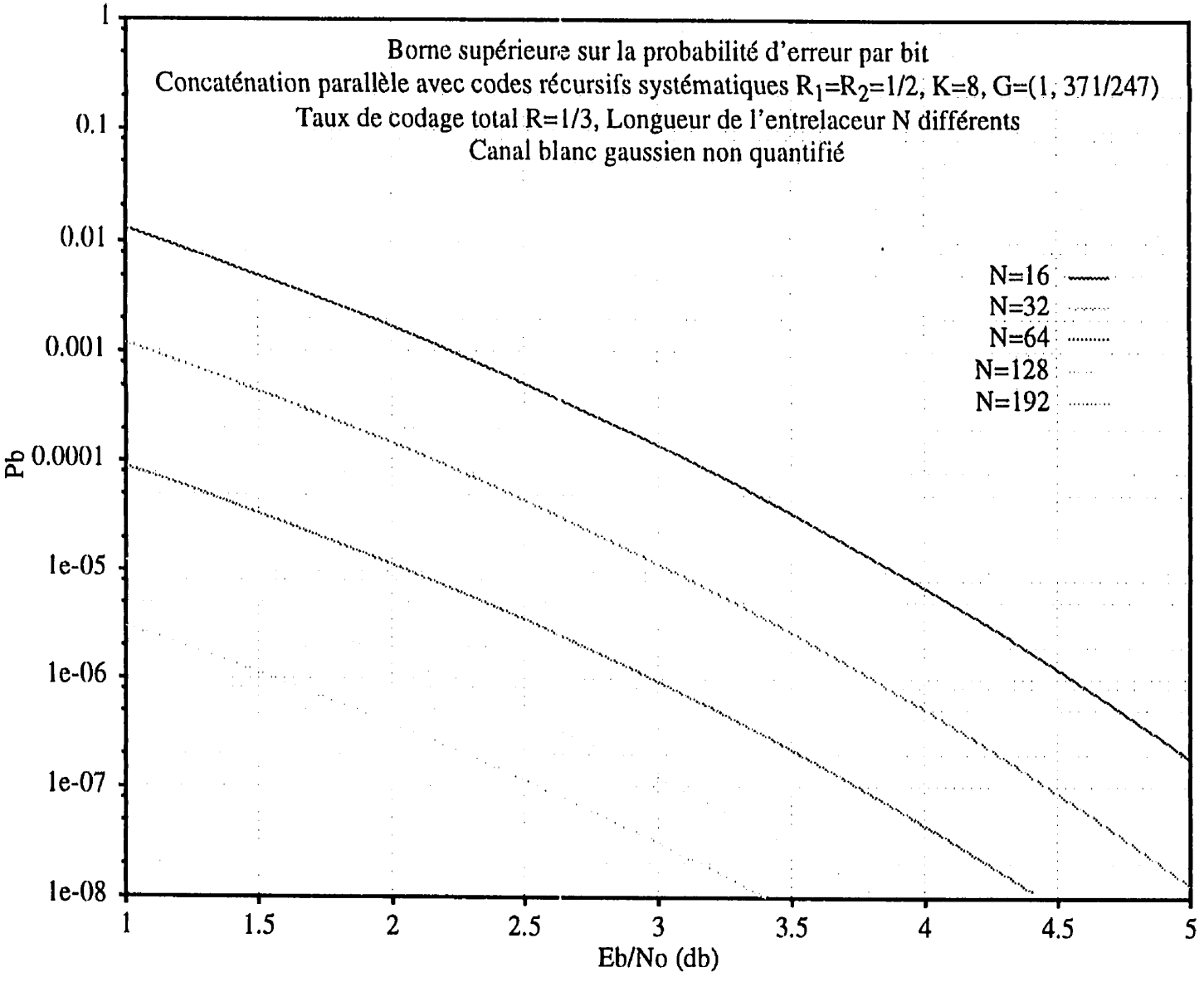


Figure A.5.3.6: Probabilités d'erreur du code CPCC avec deux codes récursifs systématiques $R_1 = R_2 = 1/2$, K = 8, G = (1, 371/247), différentes valeurs de *N*

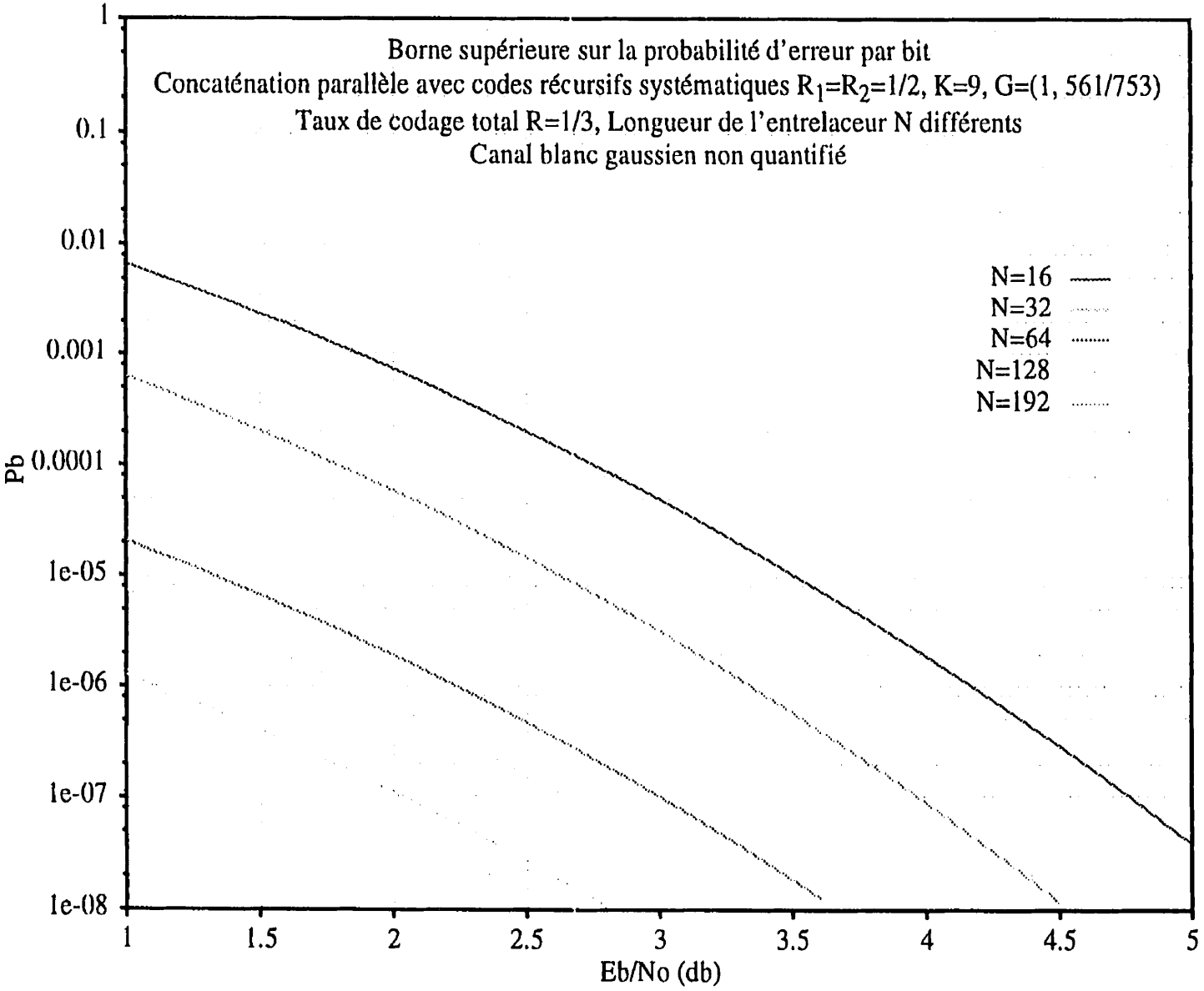


Figure A.5.3.7: Probabilités d'erreur du code CPCC avec deux codes récursifs systématiques $R_1 = R_2 = 1/2$, K = 9, G = (1, 561/753), différentes valeurs de *N*

ANNEXE VI

ALGORITHME POUR LE CALCUL DU SPECTRE DE DISTANCE DES CODES CONVOLUTIONNELS RÉCURSIFS SYSTÉMATIQUES

Conformément á la relation entre les codes convolutionnels non systématiques et les codes convolutionnels récursifs systématiques, cet alorithme peut calculer les coefficients C_d des codes convolutionnels récursifs systématiques, noté C_{rd}, en même temps que les coefficients A_d et C_d des codes convolutionnels non systématiques.

Les coefficients C_{rd} sont calculés à partir de l'arbre du code convolutionnel non systématique par l'introduction d'une variable b_s, qui représente le nombre de symboles "1" cumulé dans chaque noeud de l'arbre pour le code convolutionnel récursif systématique.

Le calcul des coefficients C_{rd} des codes convolutionnels récursifs systématiques est le suivant:

Données:

M,

$G = (G^1, G^2)$,

la distance libre d_{free},

la distance considérée $d = d_{free} + j + 1$

Préparation:

calcul de générateur en binaire $G_k = (g^k_M, g^k_{M-1}, \ldots, g^k_0)$, $k = 1, 2$

calcul du profil de distance $\boldsymbol{d} = (d_0, d_1, \ldots, d_{M-1})$

Début:

F1 (Initialisation):

$A(.) \leftarrow 0$, $C(.) \leftarrow 0$, $C_r(.) \leftarrow 0$,

$S \leftarrow (1, 0, 0, \ldots, 0)$,

$W \leftarrow d - d_0$, $m \leftarrow 1$,

$b \leftarrow 1$, $b_s \leftarrow 1$

F2 (Noeuds suivants):

Calcul de S_0, S_1, W_0 et W_1

$$b_0 = \sum_{i=1}^{M} g^1_{M-i} \cdot u_i \qquad b_1 = 1 + \sum_{i=1}^{M} g^1_{M-i} \cdot u_i$$

$S_0 = (0, u_1, \ldots, u_{M-1})$, $W_0 = W - w_0$

$S_1 = (1, u_1, \ldots, u_{M-1})$, $W_1 = W - w_1$

Si m < M, aller en F6

F3 (Retour à l'état zéro):

Si (W_0 0), alors

$A(d - W_0) \leftarrow A(d - W_0) + 1$

$C(d - W_0) \leftarrow C(d - W_0) + b$

$C_r(d - W_0) \leftarrow C_r(d - W_0) + b_s$

F4 (Extension de la branche "1"?)

Si ($W_1 < d_{M-1}$ ou $W < d_M$), alors aller en F5

Si non

sélectionner le noeud S_1: $S = S_1$, $W = W_1$, $b \leftarrow b + 1$, $b_s \leftarrow b_s + b_1$

$m \leftarrow 1$, aller en F2

F5 (Tester la pile):

Si la pile est vide, alors

fin de l'algorithme avec les résultats dans $A(.)$, $C(.)$, $C_r(.)$

Si non

lire les données (W, S, b, b_s) du dernier noeud de la pile,

$m \leftarrow 1$, aller en F2

F6 (Extension de la branche "0"?)

Si ($W_0 < d_{M\text{-}m\text{-}1}$), alors aller en F4

F7 (Enregistrer le noeud 1?)

Si ($\bar{W}_1$ $d_{M\text{-}1}$ et $W < d_M$), alors

empiler les données du noeud S_1: (W_1, S_1, $b + 1$, $b_s + b_1$)

sélectionner toujours le noeud S_0: $S = S_0$, $W = W_0$, $b_s \leftarrow b_s + b_0$

$m \leftarrow m + 1$, aller en F2

FIN

Les incrémentations de la variable b_s pour traverser la branche "0" et la branche "1" sont respectivement $b_s \leftarrow b_s + b_0$ et $b_s \leftarrow b_s + b_1$ où b_0 et b_1 sont respectivement les symboles à l'entrée du codeur convolutionnel récursif systématique pour atteindre les états S_0 et S_1.

Le code convolutionnel récursif systématique, les symboles b_0 et b_1 sont alors les premiers symboles du mot de code de la branche "0" et de la branche "1" à connaître pour calculer leur poids par les relations:

$$b_0 = \sum_{i=1}^{M} g^1_{M-i} \cdot u_i \qquad \text{pour la branche "0'}$$

$$b_1 = 1 + \sum_{i=1}^{M} g^1_{M-i} \cdot u_i \qquad \text{pour la branche "1'}$$

Les données à empiler dans ce cas sont alors $(W_1, S_1, b + 1, b_s + b_1)$.

Les coefficients C_{rd}, du code convolutionnel récursif systématique se trouvent à la fin de l'algorithme dans $C_r(.)$ que l'algorithme enregistre à chaque fois que le parcours atteind l'état zéro par:

$$C_r(d - W_0) \leftarrow C_r(d - W_0) + b_s$$

Les coefficients de spectre A_d et C_{rd} du code convolutionnel récursif systématique et A_d et C_d du code convolutionnel non systématique sont donc obtenus en même temps par cet algorithme.