Titre: Title:	Décodage à seuil itératif sans entrelacement des codes convolutionnels doublement orthogonaux
Auteur: Author:	Christian Cardinal
Date:	2001
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Cardinal, C. (2001). Décodage à seuil itératif sans entrelacement des codes convolutionnels doublement orthogonaux [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8913/

UNIVERSITÉ DE MONTRÉAL

DÉCODAGE À SEUIL ITÉRATIF SANS ENTRELACEMENT DES CODES CONVOLUTIONNELS DOUBLEMENT ORTHOGONAUX

CHRISTIAN CARDINAL

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET INFORMATIQUE

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION

DU DIPLÔME DE PHILOSOPHIAE DOCTOR (Ph.D.)

(GÉNIE ÉLECTRIQUE ET INFORMATIQUE)

JUIN 2001

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée:

DÉCODAGE À SEUIL ITÉRATIF SANS ENTRELACEMENT DES CODES CONVOLUTIONNELS DOUBLEMENT ORTHOGONAUX

Présentée par CARDINAL Christian

en vue de l'obtention du grade de: Philosophiae Doctor (Ph.D.)

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

M. CONAN Jean, Ph.D., président

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

M. GAGNON François, ing., Ph.D., membre et codirecteur de recherche

M. FORTIER Paul, ing., Ph.D., membre

M. DESPINS Charles L., ing., Ph.D., membre

REMERCIEMENTS

Je tiens d'abord à remercier mon directeur de recherche, Dr David Haccoun, pour son encadrement exeptionnel et ses conseils judicieux. Grâce à lui, ce travail a pu être accompli dans une atmosphère très enrichissante. Je le remercie aussi pour son aide financière qu'il m'a accordé pendant les premières années de mes études au doctorat.

Je remercie également mon co-directeur de recherche, Dr François Gagnon à qui revient l'idée maîtresse sur laquelle cette recherche s'appuie. Par les nombreuses discussions, toujours constructives, il a su me guider efficacement dans ce travail. Je le remercie également pour avoir contribué à mon aide financière pendant les premières années de mes études au doctorat.

J'aimerais également adresser mes remerciements à tous mes collègues de travail de l'École polytechnique qui ont su créer une ambiance dynamique au sein du laboratoire. Je pense tout particulièrement à Brice Baechler, Frédéric Dru et Khaled Lajnef. Ces remerciements s'adressent aussi à mes collèges de travail du LACIME à l'École de technologie supérieure. J'aimerais aussi remercier Ahmed Nouha pour les discussions fructueuses que nous avons eues.

Enfin, je désire remercier chaleureusement mes proches pour leur soutien et leur encouragement. Je pense particulièrement à ma mère, Thérèse, qui m'a constamment encouragé à poursuivre ce travail.

RÉSUMÉ

L'objectif de cette thèse est d'explorer une technique de codage convolutionnel avec décodage itératif sans entrelacement afin de proposer des solutions simples aux problèmes de la complexité et de la latence du codage et du décodage Turbo conventionnel.

Le décodage Turbo conventionnel exige, à chaque itération, l'utilisation d'entrelaceurs et d'un algorithme de décodage symbole par symbole. Les algorithmes de décodage symbole par symbole habituellement utilisés dans le décodage Turbo sont basés sur l'algorithme du maximum *a posteriori* et sont généralement complexes. De plus, la latence engendrée par le décodage Turbo augmente avec la longueur des entrelaceurs et du nombre d'itérations requis pour atteindre une probabilité d'erreur par bit donnée. Le décodage à seuil est aussi un algorithme de décodage symbole par symbole mais, il offre une faible complexité comparativement à d'autres algorithmes de décodage symbole par symbole. Le décodage à seuil constitue alors une alternative de choix afin de réduire la complexité du décodage Turbo. Une solution proposée dans cette thèse repose sur une méthode de décodage itératif basée sur le décodage à seuil.

La présence de l'entrelacement au codage et au décodage Turbo conventionnel permet de maintenir l'indépendance des observables dans tout le processus de décodage. Afin de réduire la latence du décodage Turbo, la solution proposée dans cette thèse consiste à éliminer ces entrelaceurs dans tout le processus de codage et de décodage Turbo. Cependant, tout en considérant un décodage à seuil à chaque itération, la contrainte d'indépendance des observables doit être satisfaite. Pour résoudre ce problème, des codes

convolutionnels doublement orthogonaux sont définis. En utilisant ces nouveaux codes, les équations de contrôle de parité générées par le décodage à seuil itératif sont orthogonales sur deux itérations consécutives et par conséquent, les observables sont indépendants sur les deux premières itérations. On définit au sens large des codes convolutionnels doublement orthogonaux lorsque la condition d'indépendance des observables n'est pas complètement satisfaite. Avec une définition au sens strict, les codes convolutionnels doublement orthogonaux satisfont entièrement cette contrainte d'indépendance.

L'analyse du décodage à seuil itératif pour des codes convolutionnels doublement orthogonaux définis au sens large apporte des solutions permettant d'améliorer les performances d'erreur. Une solution retenue consiste à pondérer les équations de contrôle de parité par des coefficients appropriés qui minimisent la probabilité d'erreur par bit à la dernière itération. Pour de forts rapports signal à bruit, les résultats de simulation montrent que ces coefficients peuvent tous être ajustés à une même valeur sans engendrer une importante dégradation des performances d'erreur comparativement à des ajustements où les coefficients ne sont pas identiques. Pour la plupart des codes convolutionnels doublement orthogonaux définis au sens large, les résultats de simulation démontrent que des gains de codage additionnels d'environ 0.5 à 1.0 dB sont possibles en utilisant des valeurs de coefficients de pondération identiques. Cette même analyse s'applique au décodage à seuil itératif des codes convolutionnels doublement orthogonaux définis au sens strict. De cette analyse, on y conclut qu'à de forts rapports signal à bruit, une

pondération des équations de contrôle de parité n'apporte aucune amélioration des performances d'erreur. Cependant, pour de plus faibles rapports signal à bruit, des améliorations substantielles des performances d'erreur sont possibles avec une pondération adéquate des équations de contrôle de parité.

Une analyse des performances d'erreur du décodage à seuil itératif pour des codes convolutionnels multiplement orthogonaux définis au sens strict indique qu'au-delà de la double orthogonalité, des améliorations marginales des performances d'erreur sont obtenues. Cette analyse montre aussi qu'après seulement trois itérations, les codes convolutionnels doublement orthogonaux définis au sens strict atteignent le gain asymptotique de codage.

Des codes convolutionnels doublement orthogonaux récursifs sont aussi définis. Une méthode de décodage itératif appropriée est développée et explorée. Des résultats de simulation montrent que l'utilisation de cette technique de codage et décodage procure de très bonnes performances d'erreur. Par exemple, les résultats de simulation indiquent, qu'avec des codes convolutionnels doublement orthogonaux récursifs définis au sens strict de taux de codage égal à 1/2, une probabilité d'erreur par bit d'environ 10^{-5} avec un rapport signal à bruit de 1.3 dB est atteignable.

Des comparaisons des techniques de codage et de décodage développées dans cette thèse avec les techniques de codage et de décodage Turbo conventionnelles montrent que, pour des rapports signal à bruit supérieurs à environ 3 dB, les codes convolutionnels

doublement orthogonaux offrent de nombreux avantages. Pour une même probabilité d'erreur par bit, le décodage itératif de ces nouveaux codes est moins complexe et sa latence est généralement plus faible comparativement à d'autres techniques de codage et décodage Turbo. Les techniques de codage et de décodage développées au cours de cette thèse sont particulièrement appropriées pour des systèmes de communication où de fortes contraintes de délai sont imposées.

ABSTRACT

The object of this thesis is to explore a new coding and iterative decoding technique without the use of interleaving and propose some practical solutions that circumvent the complexity and the latency shortcomings of the usual coding and Turbo decoding techniques.

The conventional Turbo decoding techniques use interleavers and symbol by symbol decoding algorithm. The usual symbol by symbol decoding algorithms such as the maximum *a posteriori* or MAP decoding algorithms suffer from a substantial computational complexity. Furthermore, the Turbo decoding latency increases with the length of interleaving and the number of iterations needed to achieved a given bit error probability. Threshold decoding is also another type of symbol by symbol decoding that is less complex than the other type of symbol by symbol decoding algorithms. Hence, it appears to be a very attractive alternative to the other type of symbol by symbol decoding algorithms. In this thesis, a proposed solution to the complexity shortcomings of usual Turbo decoding is to use the threshold decoding technique as the heart of the iterative decoding procedure.

One of the key aspects of Turbo decoding is the use of interleaving at both coding and decoding. The presence of interleaving ensures that the observables used in the iterative decoding procedure are independent. In order to decrease the latency of conventional Turbo decoding, the solution proposed in this thesis is to eliminate the interleavers at both encoding and at decoding. However, using threshold decoding at each iteration step, the

independence constraint on the observables must be satisfied. This problem is solved by defining the new convolutional self doubly orthogonal codes which are threshold decodable. The parity check equations generated at each iteration step of the decoding procedure are then orthogonal over two consecutive iteration steps. This means that the observables used at the second decoding step are independent. The convolutional self doubly orthogonal codes are defined in the wide sense if the condition allowing the independence of observables is not completely satisfied at the second iteration step. However, with the use of convolutional self doubly orthogonal codes defined in strict sense, all observables used at the second iteration step are independent.

From the analysis of the iterative threshold decoding for convolutional self doubly orthogonal codes defined in the wide sense, some modifications of this decoding procedure improving the error performances are proposed. One of these modifications consists in the use of weighting each parity check equation by appropriate coefficients which minimize the bit error probability at the last iteration step. The simulation results show that for high signal to noise ratio, all coefficients can take the same value without degradation of error performances as compared to the use of different coefficient values. For many convolutional self doubly orthogonal codes defined in wide sense, coding gain improvements of about 0.5 to 1.0 dB are obtained using this modification. Furthermore, for convolutional self doubly orthogonal codes defined in the strict sense, at high signal to noise ratio, the use of weighting does not improve the error performances but, for lower signal to noise ratio values, substantial improvements in error performances are also possible with the use of appropriate weighting coefficients.

The error performance analysis of iterative threshold decoding without interleaving using the convolutional self multi-orthogonal codes defined in the strict sense is presented. This analysis shows that beyond the double orthogonality, only marginal improvements in error performances are obtained. This analysis also indicates that for convolutional self doubly orthogonal codes defined in the strict sense, only three iterations are needed to achieve the asymptotic coding gain.

The recursive convolutional self doubly orthogonal codes are also defined. An appropriate iterative decoding method is developed and explored. Some simulation results show that the use of the recursive convolutional self doubly orthogonal codes defined in the strict sense provides very good error performances. For example, a bit error probability of about 10^{-5} at a signal to noise ratio of 1.3 dB may be achieved with a recursive convolutional self doubly orthogonal codes defined in the strict sense having a coding rate of 1/2.

Some comparisons of the coding and iterative decoding techniques developed in this thesis with usual coding and Turbo decoding techniques show that for signal to noise ratio values higher than 3 dB, the use of convolutional self doubly orthogonal codes provide many advantages. For the same bit error probability, the decoding procedure of these new codes is less complex and its latency is lower than the other Turbo decoding techniques. The new coding and iterative decoding techniques developed in this thesis are particularly appropriate for many communication systems where severe delay constraints are imposed.

TABLE DES MATIÈRES

REMERCIEMENTS iv

SOMMAIRE v

ABSTRACT ix

LISTE DES TABLEAUX xvii

LISTE DES FIGURES xix

LISTE DES SIGLES ET ABRÉVIATIONS xxiii

Chapitre 1 Introduction 1

1.1 Définition du projet 5

1.1.1 Motivations et problématiques de la recherche 5

1.1.2 Objectifs principaux 6

1.2 Contributions 9

1.3 Organisation de la thèse 10

Chapitre 2 Préliminaires 13

2.1 Décodage à seuil en quantification ferme pour les codes convolutionnels 13

2.1.1 Codage 16

2.1.2 Décodage 20

2.2 Décodeur à seuil à sortie pondérée (ou à sortie non-quantifiée) 26

2.2.1 Modèle du canal de transmission 27

2.2.2 Décodage à seuil et probabilité a posteriori (PAP) 30

2.2.3 Calcul des poids 34

2.2.4 Définition de l'opération «add-min» 38

2.2.5 Décodeur à seuil et valeur extrinsèque 41

Chapitre 3 Processus itératif de décodage à seuil à sortie pondérée sans entrelacement 44

3.1 Décodage à seuil itératif à sortie non-quantifiée 46

3.2 Définition des Codes Convolutionnels Doublement Orthogonaux 48

3.2.1 Définition des codes convolutionnels doublement orthogonaux au sens large 49

3.2.2 Construction des codes CSO^2C-WS 51

3.3 Décodage itératif des codes CSO^2C-WS 53

3.3.1 Première structure de décodage (Structure 1) : sans la condition 53

3.3.2 Deuxième structure de décodage (Structure 2) : avec la condition 56

3.4 Amélioration de l'algorithme de décodage à seuil itératif pour des CSO^2C-WS. 58

3.4.1 Analyse en quantification ferme du processus de décodage itératif 61

3.4.2 Méthode expérimentale 67

3.4.3 Approche par réseaux de neurones 71

3.4.4 Algorithme de rétropropagation de l'erreur adapté au décodage itératif à seuil 73

3.4.5 Sensibilité des performances d'erreur aux imprécisions des coefficients de pondération 79

3.4.6 Performance de l'algorithme de rétropropagation de l'erreur 82

3.5 Définition au sens strict des codes convolutionnels doublement orthogonaux . . 84

3.5.1 Construction des codes convolutionnels doublement orthogonaux au sens strict 87

3.6 Décodage à seuil itératif des CSO^2C-SS 88

3.7 Discussion et conclusion 92

Chapitre 4 Analyse des performances d'erreur du décodage à seuil itératif sans entrelacement 97

4.1 Probabilité d'erreur par bit du décodage à seuil à sortie non-quantifiée pour des CSOC 99

4.1.1 Fonction de densité de probabilité d'un opérateur addmin à deux entrées. . 104

4.1.2 Généralisation de la fonction de densité de probabilité pour un opérateur addmin à J entrées 110

4.1.3 Evaluation de la probabilité d'erreur du décodage à seuil pour des CSOC . 120

4.2 Extension : Évaluation de la probabilité d'erreur pour M itérations de décodage 126

4.2.1 Comparaison des résultats analytiques et de simulation 127

4.3 Conclusion 130

Chapitre 5 Codes convolutionnels doublement orthogonaux récursifs (R-CSO^2C) 131

5.1 Équivalence entre les codes à faible densité de parité et les codes doublement orthogonaux 133

5.1.1 Codes en blocs à faible densité de parité 133

5.1.2 Codes en blocs doublement orthogonaux 136

5.2 Définition et décodage des codes R-CSO^2C 137

5.2.1 Codes convolutionnels orthogonaux (d'ordre 1) récursifs (R-CSOC) 139

5.2.2 Codes convolutionnels doublement orthogonaux (ordre 2) récursifs au sens large 146

5.2.3 Codes convolutionnels doublement orthogonaux (d'ordre 2) récursifs au sens strict 154

5.2.4 Définition des codes R-CSO^{2}C-SS 155

5.3 Conclusion 171

Chapitre 6 Analyse de la complexité et de la latence du décodage itératif sans entrelacement 173

6.1 Calculs de la complexité et de la latence du décodage itératif des CSO^{2}C sans entrelacement 174

6.1.1 Complexité et latence du décodage itératif des CSO^{2}C-WS 175

6.1.1.1 Calcul du nombre d'opérations add-min à deux entrées par itération ... 175

6.1.1.2 Calcul du nombre d'additions de deux valeurs réelles 176

6.1.1.3 Calcul du nombre de multiplications de valeurs réelles 176

6.1.2 Complexité et latence du décodage itératif à seuil des CSO^{2}C-SS 177

6.1.3 Complexité et latence du décodage itératif des R-CSO^{2}C-SS 179

6.2 Comparaisons avec d'autres techniques de décodage itératif avec entrelacement 181

6.2.1 Comparaisons entre le processus de décodage à seuil itératif sans entrelacement et le processus de décodage à seuil Turbo avec entrelacement 182

6.2.2 Comparaisons entre le processus de décodage à seuil itératif sans entrelacement et le décodage Turbo conventionnel. 185

6.3 Conclusion 196

Chapitre 7 Conclusion 198

7.1 Recommandations pour recherches futures 204

RÉFÉRENCES 208

Annexe I LOGARITHME DU RAPPORT DE VRAISEMBLANCE (LRV) DE L'ERREUR DE DÉCISION 216

Annexe II DÉRIVATION DE L'OPÉRATEUR ADD-MIN 218

Annexe III RÉSULTATS DE SIMULATION - MÉTHODE EXPÉRIMENTALE .. 221

Annexe IV RÉSULTATS DE SIMULATION - OPTIMISATION PAR RESEAUX DE NEURONES 228

Annexe V RÉSULTATS DE SIMULATION DES CODES CSO^2C-SS 233

Annexe VI PROBABILITÉ D'ERREUR DU PROCESSUS DE DÉCODAGE À SEUIL ITÉRATIF EN QUANTIFICATION FERME 242

Annexe VII ALGORITHME DE RÉTROPROPAGATION DE L'ERREUR ADAPTÉ AU DÉCODAGE À SEUIL ITÉRATIF 246

Annexe VIII ALGORITHME «BELIEF PROPAGATION» OU DE PEARL 254

Annexe IX DÉCODAGE DES CODES À FAIBLE DENSITÉ DE PARITÉ 259

LISTE DES TABLEAUX

Tableau 3.1 Résultats de la construction de codes CSO^2-WS [21] 52

Tableau 3.2 Coefficients de pondération pour le CSO^2-WS, J = 8 de la Figure 3.10 . . 79

Tableau 3.3 Durée d'exécutions de l'algorithme d'optimisation 83

Tableau 3.4 Comparaison entre sens large (WS) et sens strict (SS) pour E_b/N_0 = 3.5 dB. 93

Tableau 3.5 Comparaison entre sens large (WS) et sens strict (SS) pour E_b/N_0 = 2.5 dB. 94

Tableau 6.1 Décodage à seuil à rétroaction de la décision pour des CSO^2C-WS 177

Tableau 6.2 Décodage à seuil à rétroaction de la décision pour des CSO^2C-SS 178

Tableau 6.3 Complexité de l'algorithme de décodage itératif pour les codes R-CSO^2C-SS 180

Tableau 6.4 Comparaisons entre un décodage Turbo CSOC avec entrelaceur et CSO^2C-WS 183

Tableau 6.5 Comparaisons entre un décodage Turbo CSOC avec entrelaceur et CSO^2C-SS 184

Tableau 6.6 Complexité de l'algorithme log-MAP [54]. 186

Tableau 6.7 Comparaisons de la complexité et de la latence de quelques codes CSO^2C avec celles d'un code CPCCSR, m=4, N=400. 189

Tableau 6.8 Comparaisons de la complexités et de la latence du code R-CSO^2C-SS, J=5 avec celles d'un code Turbo m=4, N=65536. 195

Tableau III.1 Coefficient de pondération opt. J=6, 4 itérations opt. à E_b/N_0=3.5 dB . 228

Tableau III.2 J=7, 4 itérations, opt. à E_b/N_0 = 3.5 dB 229

Tableau III.3 J=8, 4 Itérations, opt. à E_b/N_0 = 3.0 dB 230

Tableau III.4 Coefficients de pondération pour deux itérations 231

Tableau III.5 Coefficients de pondération pour trois itérations 231

Tableau III.6 Coefficients de pondération pour quatre itérations 232

Tableau III.7 Coefficients de pondération pour cinq itérations 232

Tableau VII.1 Règles d'initialisation pour l'algorithme de Pearl 258

LISTE DES FIGURES

Figure 1.1 Schéma de principe d'un codeur CPCCSR 2

Figure 1.2 Schéma de principe du décodage «Turbo» 3

Figure 2.1 Modèle simplifié du système de communication numérique avec canal BSC 14

Figure 2.2 Exemples de codeurs CSOC, r_c=1/2. a) m=3, J=3, b) m=6, J=4 19

Figure 2.3 Schéma de principe du calcul des symboles de syndrome 21

Figure 2.4 Schéma de principe d'un décodeur à seuil en quantification ferme 26

Figure 2.5 Modèle d'un système de communication avec un canal non-quantifié ... 27

Figure 2.6 Modèle du canal équivalent gaussien 28

Figure 2.7 Décodeur à seuil non quantifié utilisant l'opération add-min 41

Figure 3.1 Schéma de principe du décodage à seuil itératif à sortie non-quantifiée sans entrelacement pour M itérations 47

Figure 3.2 Comparaison des performances d'erreur entre des codes J=6, CSOC et CSO^{2}C pour les deux premières itérations 55

Figure 3.3 Performances d'erreur pour un code J=6 CSO^{2}C-WS utilisant les structures 1 et 2 57

Figure 3.4 Résultats de simulation pour un code CSO^2-WS, J=6, $a_j^{(\mu)}$ =1, j=0,..., 6, μ=1,..., 4 59

Figure 3.5 Structure en quantification ferme du processus de décodage à seuil itératif 63

Figure 3.6 (a) Seuils fixes $T^{(k)} = \lfloor J/2 \rfloor$, k=1, 2, ..., 10, (b) Seuils décroissants 66

Figure 3.7 Variation du coefficient a pour un code CSO^{2}C-WS, J = 8 à E_b/N_0 = 3 dB 69

Figure 3.8 Variation du coefficient a pour un code CSO^2C-WS, $J = 8$ à $E_b/N_0 = 4$ dB 69

Figure 3.9 Comparaison entre le processus itératif sans l'utilisation des coefficients et celui utilisant le coefficient optimal $a^* = 0.2$ 70

Figure 3.10 Résultats de simulation pour un code CSO^2-WS, $J = 8$ utilisant des coefficients de pondération obtenus par l'algorithme de rétropropagation de l'erreur 78

Figure 3.11 Sensibilité des performances d'erreur aux variations des coefficients de pondération pour un CSO^2C-WS, J=6, CV=28 % 80

Figure 3.12 Sensibilité des performances d'erreur aux variations des coefficients de pondération pour un CSO^2C-WS, J=6, CV=39.48 % 81

Figure 3.13 Sensibilité des performances d'erreur aux variations des coefficients de pondération pour un CSO^2C-WS, J=6, CV=70.5 % 82

Figure 3.14 Exemple d'un codeur convolutionnel doublement orthogonal au sens strict, J=3, $r_c = 3/6$ 85

Figure 3.15 Variation des coefficients de pondération a pour un code J=8 CSO^2C-SS, ... 89

Figure 3.16 Résultats de simulation pour J=8, CSO^2C-SS, $r_c = 8/16$ 90

Figure 3.17 Variation des coefficients de pondération a pour un code J=8 CSO^2C-SS, ... 91

Figure 3.18 Comparaison entre les performances d'erreur des codes CSO^2C-WS et CSO^2C-SS en fonction de leur latence totale après convergence, E_b/N_0=3.5 dB 93

Figure 3.19 Comparaison entre les performances d'erreur des codes CSO^2C-WS et CSO^2C-SS en fonction de leur latence totale après convergence, $E_b/N_0 = 2.5$ dB 95

Figure 4.1 Simulation de la FDP de λ d'un code J=6 CSOC, E_b/N_0=2.0dB, x^u=1.0 102

Figure 4.2 Représentation graphique dans le plan z_1z_2 de l'ensemble 106

Figure 4.3 Représentation graphique de dans le plan $z_1 z_2$ 107

Figure 4.4 Opérateur addmin à trois entrées réalisé à partir d'une cascade de deux opérateurs addmin à deux entrées 114

Figure 4.5 Comparaisons entre les résultats de simulation et la fonction de densité de probabilité donnée par (4.50) pour J=2, où la moyenne et la variance des variables z_i sont égales à 1 119

Figure 4.6 Comparaisons entre les résultats de simulation et la fonction de densité de probabilité donnée par (4.50) pour J=3, où la moyenne et la variance des variables z_i sont égales à 1 120

Figure 4.7 Comparaisons entre les résultats de simulation et les deux méthodes de calcul pour un code CSOC, J=6 125

Figure 4.8 Comparaisons entre les résultats de simulation et les deux méthodes de calcul pour un code CSOC, J=10 126

Figure 4.9 Comparaison entre les résultats analytiques obtenus pour un CSO^{M}C-SS, J=5 et les résultats de simulation pour un code CSO^{2}C-SS, J=5 128

Figure 4.10 Comparaison entre les résultats analytiques obtenus pour un CSO^{M}C-SS, J=10 et les résultats de simulation pour un code CSO^{2}C-SS, J=10 129

Figure 5.1 Résultats de simulation d'un code R-CSO^{2}C-WS, J=4, r_c=1/2 obtenu avec un décodage défini PAP itératif 152

Figure 5.2 Résultats de simulation d'un code R-CSO^{2}C-WS, J=4, r_c=1/2 obtenu avec une procédure itérative de décodage PAP avec rétroaction de la décision ... 153

Figure 5.3 Exemple d'un codeur R-CSO^{2}C-SS, J=2, r_c=1/2 156

Figure 5.4 Résultats de simulation pour le code R-CSO^{2}C-SS, J=5, r_c=1/2 165

Figure 5.5 Résultats de simulation pour un R-CSO^{2}C-SS, J=5, r_c=1/2, ... 168

Figure 5.6 Résultat de simulation montrant l'effet d'une rétroaction de la décision à chaque étape de décodage du processus itératif de décodage d'un code J=5, R-CSO^{2}C-SS, r_c = 1/2 170

Figure 6.1 Comparaisons des performances d'erreur de quelques codes CSO^{2}C ... 189

Figure 6.2 Comparaisons de la latence de quelques codes CSO^{2}C-WS et CSO^{2}C-SS 192

Figure 6.3 Comparaison du code CPCCSR, m=4, N=65536 avec le code R-CSO^{2}C-SS, J=5 194

Figure VIII.1 Réseaux bayesiens a) codes LDPC, b) codes Turbo 255

Figure VIII.2 Section d'un réseau bayesien illustrant le principe de parent/enfant d'un noeud X 256

Figure VIII.3 Résumé des échanges de message d'un noeud X vers les noeuds parents et les noeuds enfants 257

LISTE DES SIGLES ET ABRÉVIATIONS

ARE Algorithme de Rétropropagation de l'Erreur

APAP Approximation de la Probabilité *A Posteriori*

BPSK Binary Phase Shift Keying

BSC Binary Symetric Channel

CSOC Convolutional Self Orthogonal Code

CSO^2C Convolutional Self-Doubly Orthogonal Code

CSO^2C-WS Convolutional Self-Doubly Orthogonal Code in the Wide Sense

CSO^2C-SS Convolutional Self-Doubly Orthogonal Code in the Strict Sense

EQM Erreur Quadratique Moyenne

FDP Fonction de Densité de Probabilité

LRV Logarithme du Rapport de Vraisemblance

MAP Maximum *A Posteriori*

PAP Probabilité *A Posteriori*

R-CSOC Recursive Convolutional Self Orthogonal Code

R-CSO^2C Recursive Convolutional Self-Doubly Orthogonal Code

R-CSO^2C-WS Recursive Convolutional Self-Doubly Orthogonal Code in the Wide Sense

R-CSO^2C-SS Recursive Convolutional Self-Doubly Orthogonal Code in the Strict Sense

TEB Taux d'Erreur par Bit

CHAPITRE 1: INTRODUCTION

La remarquable théorie de Shannon précise que si les symboles d'information sont transmis à travers un canal avec un taux de transmission inférieur à une valeur limite appelée capacité du canal, en utilisant un code correcteur d'erreurs approprié, il est possible d'atteindre une probabilité d'erreurs arbitrairement petite [1]. Cependant, la théorie de Shannon ne donnent aucune indication sur la manière de construire ces codes. Depuis la publication des résultats de Shannon, un nombre considérable de travaux ont porté sur la recherche de codes et sur le développement de techniques de décodage pratiques. Les résultats de ces travaux ont permis d'améliorer de façon efficace la fiabilité des communications à haute vitesse. Sur un canal discret sans mémoire où le bruit est gaussien blanc et additif, plusieurs techniques de codage et de décodage permettent de communiquer de manière fiable avec des rapports signal à bruit E_b/N_o correspondant au taux de coupure R_o. Cependant, pour la plupart de ces techniques, lorsque le taux de codage r_c, est supérieur au taux de coupure, c'est-à-dire lorsque $R_o < r_c < C$, où C est la capacité du canal exprimé en bit par symbole codé, les performances d'erreur se dégradent rapidement ou encore, la complexité et la latence engendrées par ces techniques deviennent très grandes.

Plusieurs techniques de codage et de décodage ont été développées afin de surmonter ces difficultés. Par exemple, une approche souvent utilisée est basée sur la concaténation sérielle d'un code convolutionnel de taux de codage r_c faible avec un code en bloc ayant

un taux de codage plus élevé, les deux codeurs étant séparés d'un entrelaceur de grande taille [2], [3].

Plus récemment, une technique de codage utilisant une *concaténation parallèle de codeurs convolutionnels systématiques et récursifs* (CPCCSR) a été proposée par Berrou et *al.* [4]. L'utilisation de ces codes permet d'opérer à des taux de codage, r_c, nettement supérieur au taux de coupure, R_o, s'approchant même de la limite théorique prédite par le théorème de la capacité de Shannon [1] et ce en utilisant un décodage de complexité raisonnable. Ces codes sont décodés en utilisant un processus itératif de décodage où à chaque itération, un algorithme de décodage symbole par symbole est utilisé.

Le codage CPCCSR est généralement constitué de deux codeurs convolutionnels systématiques et récursifs séparés par un entrelaceur. Comme la Figure 1.1 le montre, la séquence d'information est codée deux fois. Le premier codeur utilise la séquence d'information pour générer une première séquence de symboles de parité alors que le deuxième codeur utilise une version entrelacée de la séquence d'information pour produire une deuxième séquence de symboles de parité.

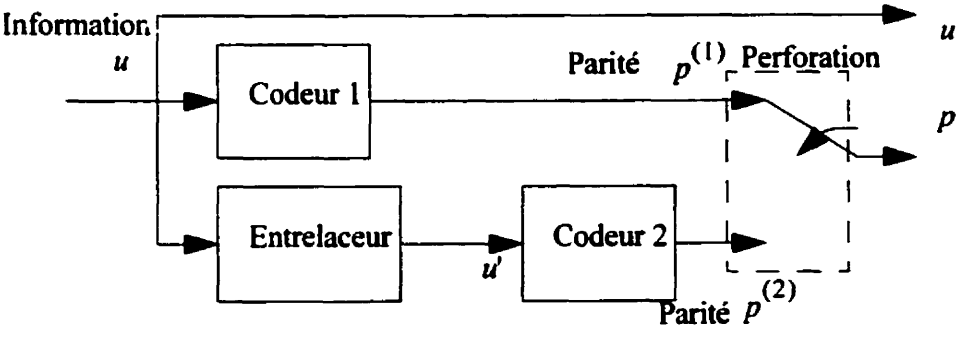


Figure 1.1 Schéma de principe d'un codeur CPCCSR

Le décodage des codes CPCCSR est essentiellement basé sur une procédure itérative de décodage dite «Turbo». La Figure 1.2 montre le schéma de principe du décodage «Turbo». L'idée principale est de décoder itérativement chaque code constituant en utilisant un algorithme de décodage symbole par symbole tout en s'assurant qu'à chaque itération, les observables sont indépendants. Cette condition d'indépendance entre les observables qui est essentielle au fonctionnement du processus itératif de décodage est satisfaite grâce à l'utilisation de l'entrelacement des observables à chaque itération. À la première itération, les observables sont statistiquement indépendants alors qu'après quelques itérations, puisque les mêmes observables sont réutilisés, la corrélation entre ceux-ci augmente n'entraînant que des améliorations minimes des performances d'erreur.

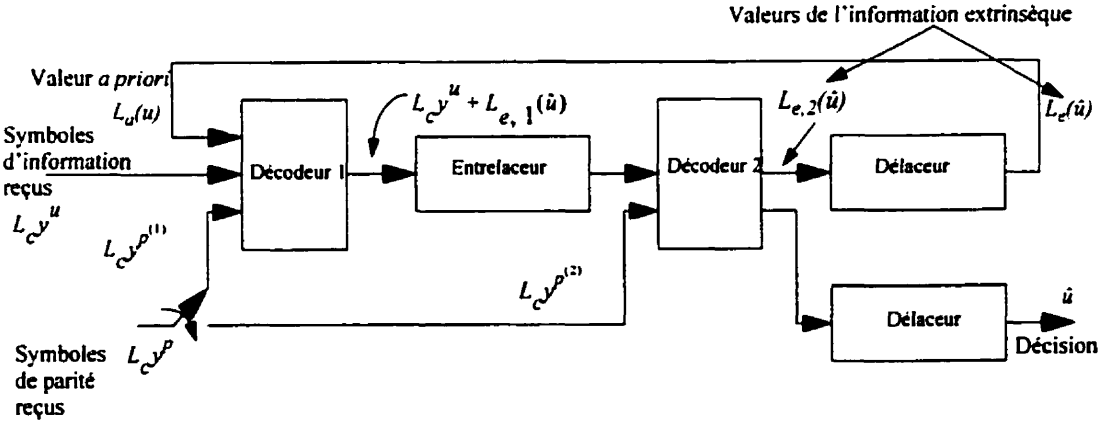


Figure 1.2 Schéma de principe du décodage «Turbo»

L'algorithme de décodage de Bahl (BCJR) [5] modifié par Berrou *et al.* [4] permet un décodage itératif des codes CPCCSR. L'algorithme BCJR consiste à décoder symbole par symbole l'information codée en déterminant le *maximum a posteriori* (MAP) de chaque

symbole d'information. La complexité de l'algorithme BCJR, évaluée à environ quatre fois celle du décodeur de Viterbi [5] rend difficile la réalisation matérielle du décodeur. Certains efforts ont été déployés notamment par Pietrobon [6], [7], [8], [9] afin de réduire cette complexité et permettre une réalisation matérielle raisonnable. Cependant, cette réduction de la complexité engendre aussi une réduction des performances d'erreur.

D'autres algorithmes de décodage symbole par symbole peuvent être utilisés dans le décodage Turbo. Par exemple, l'algorithme SOVA (Soft Output Viterbi Algorithm) [10] donne des performances d'erreur semblables à celles obtenues avec l'algorithme BCJR mais, avec une complexité moindre [11]. Un autre exemple de décodage symbole par symbole est le décodage à seuil. En effet, Riedel et Svirid [12], [13] proposent une méthode permettant d'utiliser le décodage à seuil dans le décodage Turbo. Les codes utilisés par les auteurs de [12] et [13] sont obtenus à l'aide d'une concaténation parallèle de codeurs *non-récursifs*. De plus, les codes constituants doivent être orthogonaux et systématiques afin que le décodage à seuil puisse s'appliquer [14]. Lorsqu'une concaténation parallèle de codeurs non-récursifs est utilisée, les développements théoriques de Benedetto [16], [17] montrent que l'entrelacement contribue peu à l'amélioration des performances d'erreur. Toutefois, l'entrelacement continue d'agir de manière à fournir à chaque itération des observables faiblement corrélés. D'autre part, lorsque les codes convolutionnels sont récursifs, l'auteur de [16] montre que l'amélioration des performances d'erreur est inversement proportionnelle à la taille de l'entrelacement.

La taille des entrelaceurs utilisés au codage et au décodage Turbo produit une augmentation de la latence. De plus, cette latence dépend aussi de l'algorithme de décodage utilisé. Par exemple avec l'algorithme BCJR, il est nécessaire d'attendre la réception complète de la séquence de symboles codés avant que le décodeur puisse générer sa première décision. Par conséquent, une augmentation de la taille des entrelaceurs et du nombre d'itérations requis permettant d'atteindre une probabilité d'erreur donnée engendre nécessairement une augmentation importante de la latence.

1.1 Définition du projet

Dans cette section, on définit ce projet de recherche. Les motivations et les problématiques de recherche sont tout d'abord exposées. Par la suite, les objectifs principaux sont expliqués en présentant les approches de solutions.

1.1.1 Motivations et problématiques de la recherche

Malgré la grande capacité de correction des erreurs caractérisant les codes CPCCSR, la réalisation pratique du décodage Turbo présente un certain nombre de problèmes. Parmi ceux-ci, la complexité et la latence du décodage Turbo sont souvent les problèmes qui dominent. Par exemple, dans des systèmes de communication où le délai de transmission est un paramètre critique, la grande latence engendrée par la présence des entrelaceurs dans le décodage Turbo devient un inconvénient majeur. Pour des systèmes de communication à haut débit d'information, assurer une communication fiable tout en considérant les contraintes de complexité et de délai de transmission est présentement un des problèmes principaux de conception. Réduire la complexité et la latence tout en

maintenant des performances d'erreur acceptables est la principale raison qui a motivé la poursuite de cette recherche.

1.1.2 Objectifs principaux

Ce projet de thèse se consacre tout particulièrement aux problèmes de réduction de la latence et de la complexité des processus itératifs de décodage. La grande complexité du processus de décodage Turbo qui dépend principalement de l'algorithme de décodage utilisé a pour effet de limiter le taux de transmission binaire. De plus, la latence engendrée par le décodage Turbo limite aussi l'application de cette technique de décodage itérative à des systèmes de communication à très haut débit.

Cette thèse a pour objectif d'explorer une procédure de codage et de décodage itératif qui réduit la complexité et la latence des techniques traditionnelles de codage et décodage Turbo tout en maintenant des performances d'erreur acceptables sans avoir recours à l'entrelacement.

Le décodage Turbo nécessite, à chaque itération, l'utilisation d'un algorithme de décodage symbole par symbole. Le décodage à seuil étant aussi un décodage symbole par symbole, il constitue une alternative intéressante aux autres algorithmes de décodage habituellement utilisés dans les techniques de décodage Turbo. En effet, la faible complexité mesurée en nombre d'opérations à effectuer par le décodeur à seuil permet une réalisation matérielle simple [18], [19]. Pour résoudre le problème de complexité associé au décodage Turbo, l'approche proposée est de considérer un décodage itératif basé sur le

décodage à seuil.

La présence d'entrelaceurs au codage et au décodage Turbo engendre une latence souvent importante. L'entrelacement permet de maintenir l'indépendance des observables dans le processus de décodage itératif Turbo. Dans cette thèse, on y propose une approche permettant de réduire cette latence. Elle consiste à éliminer les entrelaceurs au codage ainsi que dans tout le processus itératif de décodage. Évidemment, cette approche doit tenir compte de la contrainte qui consiste à maintenir l'indépendance entre les observables à chaque itération mais, sans avoir recours à des entrelaceurs dans tout le processus itératif de codage et de décodage.

Pour résoudre ce problème, on doit définir un sous-ensemble de codes convolutionnels qui satisfont les contraintes pour qu'un décodage itératif à seuil sans entrelacement soit applicable. Les codes convolutionnels pouvant être décodés par l'algorithme de décodage à seuil, sont des codes convolutionnels orthogonaux CSOC (Convolutional Self Orthogonal Codes) [2], [14], [15]. Pour maintenir l'orthogonalité et donc l'indépendance sur deux itérations sans avoir recours à un entrelaceur, il est nécessaire d'ajouter des propriétés aux codes convolutionnels orthogonaux. Ces propriétés consistent à augmenter le niveau d'orthogonalité des codes. Les codes obtenus sont appelés «codes convolutionnels doublement orthogonaux» (Convolutional Self Doubly Orthogonal Codes, CSO^2C)†.

---

†. La terminologie francophone n'étant pas encore établie pour cette nouvelle technique de codage, nous utiliserons dans tout le reste de cette thèse la terminologie anglophone.

Dans ce projet, on définit les conditions pour qu'un code convolutionnel systématique soit doublement orthogonal. Deux définitions de la double orthogonalité d'un code convolutionnel sont données. Une définition au sens large est tout d'abord obtenue. Les codes résultant de cette définition se nomment codes convolutionnels doublement orthogonaux définis au sens large (Convolutional Self Doubly Orthogonal Codes in Wide Sense, CSO^{2}C-WS). Ces codes étant définis au sens large, à la deuxième itération de décodage, la contrainte d'indépendance des observables n'est pas complètement respectée. D'autre part, une définition des codes convolutionnels doublement orthogonaux au sens strict (Convolutional Self Doubly Orthogonal Codes in Strict Sense, CSO^{2}C-SS) est obtenue. Avec cette deuxième définition, la contrainte d'indépendance des observables de la deuxième itération est satisfaite. En se basant sur la structure des codes à faible densité de parité [20] (Low Density Parity Check Codes, LDPC), des codes convolutionnels doublement orthogonaux récursifs définis au sens large et au sens strict (Recursive Convolutional Self Doubly Orthogonal Codes in Wide Sense et Strict Sens, R-CSO^{2}C-WS, R-CSO^{2}C-SS) sont obtenus. De plus, un nouvel algorithme de décodage approprié aux codes CSO^{2}C-WS et R-CSO^{2}C-SS est élaboré.

Les problèmes inhérents au décodage des CSO^{2}C et l'analyse de ce processus itératif de décodage sont alors considérés. Des solutions pratiques sont élaborées pour résoudre ces problèmes. L'évaluation des performances d'erreur des CSO^{2}C est faite tout d'abord au moyen de simulation. Par la suite, une analyse théorique des performances d'erreur du processus de décodage itératif à seuil des codes CSO^{2}C est proposée.

1.2 Contributions

Les travaux de recherche réalisés dans le cadre de cette thèse ont permis d'apporter les contributions suivantes.

1. Élaboration d'une nouvelle méthode de codage et de décodage itératif sans entrelacement de faible complexité et de faible latence.
2. Définitions au sens large et au sens strict des codes convolutionnels doublement orthogonaux.
3. Amélioration du processus de décodage itératif à seuil sans entrelacement des codes convolutionnels doublement orthogonaux définis au sens large.
4. Analyse du décodage itératif à seuil sans entrelacement et conclusions originales concernant l'utilisation des codes convolutionnels doublement orthogonaux définis au sens strict.
5. Définitions au sens large et au sens strict de nouveaux codes convolutionnels récursifs doublement orthogonaux.
6. Analyse et développement d'un nouvel algorithme de décodage des codes convolutionnels récursifs doublement orthogonaux définis au sens large et au sens strict.
7. Conclusions originales qui résultent des comparaisons entre les techniques de décodage itératif des codes convolutionnels doublement orthogonaux et la technique conventionnelle de décodage Turbo.

1.3 Organisation de la thèse

Cette thèse contient sept chapitres. Après ce chapitre d'introduction, le chapitre 2 décrit les éléments fondamentaux du décodage à seuil des codes convolutionnels orthogonaux (CSOC). Le décodage à seuil en quantification ferme et le décodage à seuil à sortie pondérée sont alors abordés. Une définition de l'orthogonalité simple des codes CSOC est donnée. On y présente aussi la notation qui sera utilisée tout au long de cette thèse.

Le processus itératif de décodage à seuil à sortie pondérée sans entrelacement est décrit au chapitre 3. Les définitions des codes convolutionnels doublement orthogonaux (CSO^2C) au sens large et au sens strict sont déduites de la structure de décodage itératif à seuil sans entrelacement. Par la suite, trois structures possibles de décodage itératif des CSO^2C-WS sont décrites. Chacune de ces structures impose des conditions particulières sur l'indépendance des observables utilisés dans le processus itératif de décodage à seuil. Une amélioration possible de l'algorithme de décodage itératif à seuil pour des codes CSO^2C-WS est alors présentée. Cette amélioration consiste à pondérer les équations de contrôle de parité de chaque itération par des coefficients. Une méthode expérimentale et une approche basée sur les réseaux de neurones sont élaborées afin de déterminer les coefficients de pondération appropriés qui minimisent la probabilité d'erreur à la dernière itération. Par la suite, le décodage itératif à seuil des codes CSO^2C-SS est abordé. Finalement, dans la dernière section de ce chapitre, des comparaisons sont faites entre les

codes CSO^2C-WS et CSO^2C-SS en fonction de la latence et des performances d'erreur que leur utilisation procure.

Au chapitre 4, on présente une analyse des performances d'erreur du décodage itératif à seuil pour des codes convolutionnels multiplement orthogonaux définis au sens strict (CSO^MC-SS) sans entrelacement. Une nouvelle approche de calcul de la probabilité d'erreur par bit du décodage à seuil à sortie pondérée des codes CSOC est tout d'abord présentée. Cette méthode d'analyse est basée sur le calcul de la fonction de densité de probabilité d'un opérateur «add-min». Une extension de la méthode d'analyse élaborée pour le décodage à seuil à sortie pondérée permet d'obtenir une analyse de la probabilité d'erreur pour M itérations de décodage à seuil à sortie pondérée. Cette analyse est essentiellement basée sur l'hypothèse que le code utilisé est multiplement orthogonal sur au moins M itérations consécutives. Quelques comparaisons des résultats de cette analyse avec ceux obtenus par simulation des codes CSO^2C-SS permettent d'apporter des conclusions sur l'utilisation des codes convolutionnels multiplement orthogonaux définis au sens strict.

Dans le chapitre 5, une définition des codes convolutionnels doublement orthogonaux récursifs (Recursive CSO^2C, R-CSO^2C) est donnée. Cette définition résulte de l'équivalence entre les codes en blocs doublement orthogonaux [21], [22] et les codes à faible densité de parité introduits par Gallager [20]. Un algorithme de décodage itératif semblable à celui utilisé pour les codes à faible densité de parité est alors élaboré. Des

résultats de simulation des codes R-CSO^{2}C définis au sens large et au sens strict sont présentés.

Une analyse de la complexité et de la latence du décodage itératif à seuil sans entrelacement est présentée au chapitre 6. Les résultats de cette analyse sont comparés à la procédure de décodage Turbo conventionnel. Certaines conclusions sont alors obtenues à partir de ces comparaisons.

Finalement, le chapitre 7 conclut cette thèse en présentant un résumé des principaux résultats obtenus au cours de ce travail de recherche. Des recommandations relatives à des recherches futures sont présentées à la fin de ce chapitre.

CHAPITRE 2: PRÉLIMINAIRES

2.1 Décodage à seuil en quantification ferme pour les codes convolutionnels

Dans cette section, on décrit tout d'abord, le modèle simplifié du système de communication numérique en quantification ferme. Suivra ensuite, une description du processus de codage et décodage en quantification ferme des codes convolutionnels systématiques orthogonaux (*CSOC, Convolutional Self-Orthogonal Codes*).

Sans perte de généralité, nous limitons notre discussion à des systèmes de transmission codés dont les codes sont systématiques de taux de codage égal à $r_c = 1/2$. De simples modifications apportées à leur structure mathématique permettront une généralisation à des codes systématiques de taux de codage, $r_c = b/v$, $b<v$, $v>1$. Le modèle simplifié du système de communication en quantification ferme est illustré à la Figure 2.1. Ce système est constitué d'un codeur de canal, d'un canal discret sans mémoire et d'un décodeur. Le canal discret et sans mémoire comporte un modulateur BPSK, un canal physique et un récepteur optimal dont la sortie est quantifiée sur 1 bit. Dans ce cas, le canal discret résultant se nomme canal binaire symétrique (BSC, Binary Symmetric Channel).

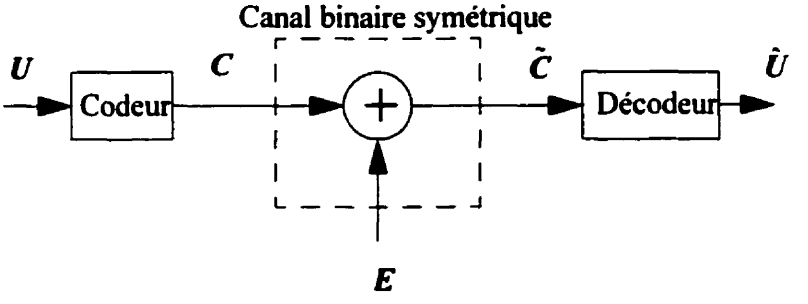


Figure 2.1 Modèle simplifié du système de communication numérique avec canal BSC

Soit $\boldsymbol{U} = (u_0, u_1, \ldots)$ la séquence de bits d'information à l'entrée du codeur où u_i est un bit d'information, $u_i \in \{0, 1\}$ et où l'indice i, $i = 0, 1, 2, \ldots$, représente l'instant de transmission. Cette séquence d'information est fournie à l'entrée du codeur de taux de codage r_c=1/2. Celui-ci génère à sa sortie, la séquence codée $\boldsymbol{C} = (\boldsymbol{c}_0, \boldsymbol{c}_1, \ldots)$. Les éléments $\boldsymbol{c}_i$ de la séquence $\boldsymbol{C}$ sont constitués du bit d'information u_i suivi du symbole de parité p_i ce qui revient à $\boldsymbol{c}_i = (u_i, p_i)$, $i = 0, 1, 2, \ldots$. La séquence de parité $\boldsymbol{P} = (p_0, p_1, \ldots)$ est générée par le codeur en effectuant des opérations modulo-2 sur un ensemble fini de bits d'information.

Les vecteurs de symboles codés, $\boldsymbol{c}_i = (u_i, p_i)$, $i = 0, 1, 2, \ldots$, sont transmis dans un canal binaire symétrique. À la sortie du canal, on observe donc la séquence de symboles codés contaminée par une séquence d'erreurs binaires $\boldsymbol{E} = (\boldsymbol{e}_0, \boldsymbol{e}_1, \ldots)$. Le vecteur d'erreur $\boldsymbol{e}_i = (e_{i,1}, e_{i,2})$ est formé des deux éléments $e_{i,1} = e_i^u$ et $e_{i,2} = e_i^p$,

$i = 0, 1, 2, \ldots$ qui affectent le bit d'information u_i et le symbole de parité p_i respectivement. Le canal étant quantifié à 2 sorties (quantification ferme), le premier élément binaire, e_i^u , est égal à:

$$e_i^u = \begin{cases} 1 \text{ avec une probabilité } (\gamma_0) \\ 0 \text{ avec une probabilité } (1-\gamma_0) \end{cases}, \tag{2.1}$$

et représente l'erreur sur le bit d'information u_i. De la même manière, l'erreur e_i^p, affectant le symbole de parité p_i est donnée par:

$$e_i^p = \begin{cases} 1 \text{ avec une probabilité } (\gamma_0) \\ 0 \text{ avec une probabilité } (1-\gamma_0) \end{cases} \tag{2.2}$$

Lorsqu'une modulation BPSK est utilisée pour transmettre l'information dans un canal réel dont le bruit additif est blanc, gaussien, de moyenne nulle et de spectre de densité de puissance bilatéral égal à $N_0/2$, γ_0 représente la probabilité de transition du canal binaire symétrique exprimée par [23]:

$$\gamma_0 = Q(\sqrt{2r_c E_b/N_o}) \tag{2.3}$$

où $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty \exp\left(-\frac{\alpha^2}{2}\right) d\alpha$ et où E_b représente l'énergie par bit d'information. Ici, le taux de codage est $r_c = 1/2$ de sorte que (2.3) devient $\gamma_0 = Q(\sqrt{E_b/N_o})$. Par conséquent, la sortie du canal est representée par la séquence $\tilde{C} = (\tilde{c}_0, \tilde{c}_1, \ldots)$ où

$\tilde{c}_i = c_i \oplus e_i$ et où $\oplus$ représente l'opération d'addition modulo-2. Les vecteurs de symboles codés reçus du canal et présents à l'entrée du décodeur sont donc dénotés par $\tilde{c}_i = (\tilde{u}_i, \tilde{p}_i)$ où:

$$\tilde{u}_i = u_i \oplus e_i^u \tag{2.4}$$

et

$$\tilde{p}_i = p_i \oplus e_i^p . \tag{2.5}$$

Observant la séquence reçue $\tilde{C}$, le décodeur produit à sa sortie une estimation de chacun des bits d'information, $\hat{u}_i$, $i = 0, 1, 2, \ldots$, de la séquence décodée $\hat{U}$.

Dans cette section, le décodage à seuil pour des codes convolutionnels systématiques orthogonaux, CSOC, (Convolutional Self-Orthogonal Codes) en quantification ferme est considéré. Nous limitons ici notre étude aux codes convolutionnels orthogonaux systématiques de taux de codage $r_c = 1/2$ et de mémoire m [14]. Le décodeur à seuil comportant une réplique exacte du codeur, il convient de décrire en premier lieu le processus de codage convolutionnel des codes CSOC [2], [14], [15].

2.1.1 Codage

Pour un code systématique de taux de codage $r_c = 1/2$, le processus de codage consiste à produire à chaque instant i le symbole de parité p_i correspondant à chaque bit d'information u_i. En général, un code systématique de taux de codage $r_c = 1/2$ est

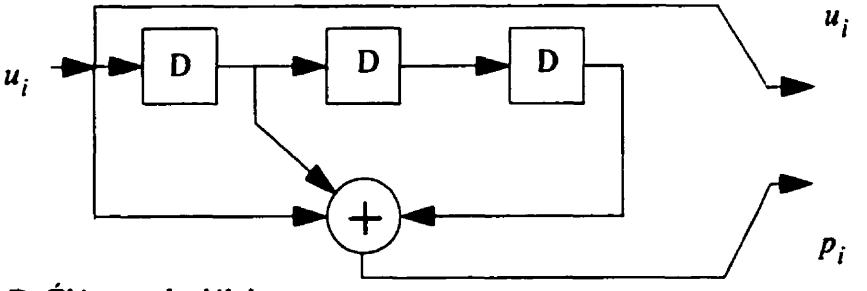
complètement spécifié par un vecteur de connexions $\boldsymbol{g} = (g_0, g_1, \ldots, g_m)$, où g_k est un élément binaire pouvant prendre la valeur 0 ou 1. Tel que montré à la Figure 2.2, une composante $g_k = 1$ signifie que la $k^{ième}$ cellule du codeur est connectée à l'additionneur modulo-2 tandis que $g_k = 0$ signifie qu'il n'y a pas de connexion entre la $k^{ième}$ cellule du codeur et l'additionneur modulo-2. Considérons maintenant les J composantes non-nulles, $J \leq m$, $g_{\alpha_1}, g_{\alpha_2}, \ldots, g_{\alpha_J}$ du vecteur de connexions $\boldsymbol{g}$ où $\alpha_1 < \alpha_2 < \ldots < \alpha_J$. Un code convolutionnel systématique peut être aussi spécifié par l'ensemble des valeurs entières α_j, $j = 1, 2, \ldots, J$. Tel qu'illustré à la Figure 2.2, $\alpha_j = k$ signifie que la $j^{ième}$ composante non-nulle du vecteur de connexions, $\boldsymbol{g}$, représente une connexion entre la $k^{ième}$ cellule du codeur et l'additionneur modulo-2. Le symbole de parité, p_i, est alors construit en utilisant le bit d'information courant, u_i, et une sélection particulière de $(J-1)$ bits d'information choisis parmi les $(m-1)$ autres bits d'information qui ont précédé u_i et qui résident dans le codeur. Ceci s'exprime comme suit :

$$p_i = \sum_{j=1}^{J} \oplus u_{i-\alpha_j}, \quad i = 0, 1, \ldots \tag{2.6}$$

où le symbole $\oplus$ représente la somme modulo-2 sur les deux valeurs binaires 0 et 1. Pour qu'un code soit complètement orthogonal [2], il faut que les $J(J-1)/2$ différences positives, $(\alpha_j - \alpha_l)$, $j = 1, 2, \ldots, J$, $l = 1, 2, \ldots J$, $j > l$, appartenant à l'ensemble des

différences des valeurs entières, $\{(\alpha_j - \alpha_l)\}$, soient distinctes entre eux. La valeur maximale des $\alpha_j, j = 1, 2, \ldots, J$, dénotée par α_J est égale à la mémoire m du codeur.

Le bit d'information u_i et le symbole de parité p_i forment le vecteur de symboles codés c_i qui sera transmis sur un canal binaire symétrique. La Figure 2.2 montre deux exemples de codeurs convolutionnels systématiques orthogonaux de taux de codage $r_c = 1/2$ avec m=3 et $J = 3$ et m=6 et $J = 4$ respectivement. À la Figure 2.2 (a), les différences sont $(\alpha_3 - \alpha_2) = 2$; $(\alpha_3 - \alpha_1) = 3$; $(\alpha_2 - \alpha_1) = 1$ alors que pour le codeur de la Figure 2.2 (b), l'ensemble des différences est $(\alpha_4 - \alpha_3) = 2$; $(\alpha_4 - \alpha_2) = 5$; $(\alpha_4 - \alpha_1) = 6$; $(\alpha_3 - \alpha_2) = 3$; $(\alpha_3 - \alpha_1) = 4$; $(\alpha_2 - \alpha_1) = 1$. Il est clair que la condition d'orthogonalité de ces codes est satisfaite car toutes les différences $(\alpha_j - \alpha_l)$ sont distinctes.



D: Élément de délai

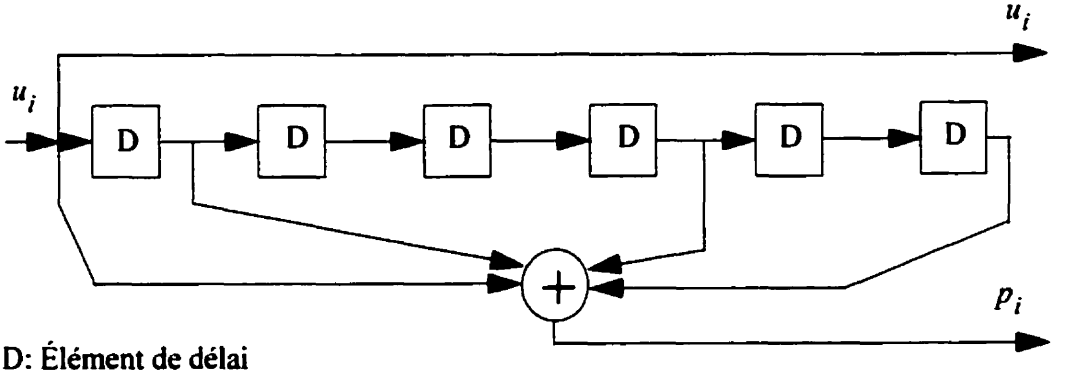
$g = (1101)$

$c_i = (u_i, p_i)$

$\alpha_1 = 0, \alpha_2 = 1, \alpha_3 = 3$

$m = \alpha_3 = 3$

(a)



D: Élément de délai

$g = (1100101)$

$c_i = (u_i, p_i)$

$\alpha_1 = 0, \alpha_2 = 1, \alpha_3 = 4, \alpha_4 = 6$

$m = \alpha_4 = 6$

(b)

Figure 2.2 Exemples de codeurs CSOC, r_c=1/2. a) m=3, J=3, b) m=6, J=4

2.1.2 Décodage

L'algorithme de décodage à seuil pour les codes convolutionnels est une procédure dite algébrique. Cette technique de décodage se distingue de celles de Viterbi et séquentielle pour les raisons suivantes. Les techniques de décodage de Viterbi ou séquentielle sont basées sur des considérations probabilistes et non pas algébriques. De plus, le décodage à seuil produit une décision finale sur le bit d'information courant en ne considérant que les symboles codés reçus sur une longueur $(m+1)$ plutôt que sur la séquence entière de symboles codés reçue comme c'est le cas pour les techniques de Viterbi et séquentielle.

L'algorithme de décodage à seuil est basé principalement sur le calcul des symboles de syndrome. Le décodeur reconstruit le symbole de parité, p'_i, à partir du bit d'information reçu $\tilde{u}_i$ et le compare au symbole de parité $\tilde{p}_i$ correspondant reçu du canal. Les résultats de ces comparaisons forment les symboles de syndrome, s_i, lesquels sont stockés dans un registre de longueur α_J. Les symboles de syndrome sont combinés entre eux pour déterminer, selon une simple règle de décision majoritaire, une estimation, $\hat{e}_i^u$, du symbole d'erreur ayant perturbé le bit d'information courant u_i. L'estimation, $\hat{u}_i$, du bit d'information courant est donc obtenue en effectuant l'opération $\hat{u}_i = \tilde{u}_i \oplus \hat{e}_i^u$. La décision est donc $\hat{u}_i$ et naturellement, il y aura une erreur si $\hat{u}_i \neq u_i$.

Soit un code convolutionnel systématique orthogonal, CSOC, ayant un taux de codage

$r_c = 1/2$. Au récepteur, on forme les symboles de syndrome s_l en calculant la somme modulo-2 $\tilde{p}_l \oplus p'_l$. La Figure 2.3 illustre le principe du calcul des symboles de syndrome au récepteur.

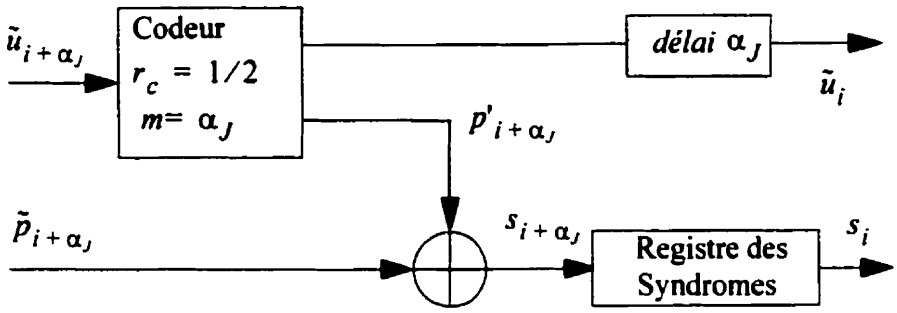


Figure 2.3 Schéma de principe du calcul des symboles de syndrome

Chaque symbole de syndrome correspond alors à une équation faisant intervenir le bit d'information, $\tilde{u}_i$, le symbole de parité, $\tilde{p}_i$, ainsi que $(J-1)$ autres bits d'information reçus du canal. Le symbole de syndrome, s_{i+k}, se trouvant dans la $k^{ième}$ cellule du registre des syndromes à l'instant $i = 0, 1, 2, \ldots$ s'écrit comme suit :

$$s_{i+k} = \tilde{p}_{i+k} \oplus \sum_{j=1}^{J} \tilde{u}_{i+k-\alpha_j}, \quad k = 0, 1, \ldots, m \tag{2.7}$$

En combinant les équations (2.4) à (2.7), on obtient pour chacun des $(m+1)$ symboles de syndrome la relation suivante :

$$s_{i+k} = e^{p}_{i+k} \oplus \sum_{j=1}^{J} \oplus e^{u}_{i+k-\alpha_j}, \quad k = 0, 1, \ldots, m \tag{2.8}$$

Un ensemble de J équations de parité $\{A_{i,1}, A_{i,2}, \ldots, A_{i,J}\}$ est orthogonal au symbole d'erreur e^{u}_{i}, si chaque équation de parité inclut le symbole e^{u}_{i} et que les autres symboles d'erreur n'apparaissent qu'une seule fois dans l'ensemble des J équations de parité. Pour obtenir l'estimation $\hat{e}^{u}_{i}$, on utilise une règle de décision basée sur une logique majoritaire :

Choisir $\hat{e}^{u}_{i} = 1$ *si et seulement si plus de* $\lfloor J/2 \rfloor^{\dagger}$ *équations de parité* $A_{k,i}$, $k = 1, 2, \ldots, J$, *orthogonales à* e^{u}_{i} *sont égales à la valeur* 1; *Choisir* $\hat{e}^{u}_{i} = 0$ *autrement.*

Pour un code convolutionnel complètement orthogonal, les J équations de parité orthogonales à e^{u}_{i} s'écrivent, en se référant à (2.8), de la façon suivante:

$$A_{k,i} \triangleq s_{i+\alpha_k} = e^{p}_{i+\alpha_k} \oplus e^{u}_{i} \oplus \sum_{j=1}^{k-1} \oplus e^{u}_{i+\alpha_k-\alpha_j} \oplus \sum_{j=k+1}^{J} \oplus e^{u}_{i+\alpha_k-\alpha_j},$$

$$k = 1, 2, \ldots, J, \; i = 0, 1, 2, \ldots \tag{2.9}$$

Le code étant complètement orthogonal, il est clair que chaque équation de parité $A_{k,i}$ est égale à $s_{i+\alpha_k}$. Les J équations données par (2.9) constituent les équations de contrôle

†. $\lfloor x \rfloor$ représente la partie entière du réel x, i.e. le plus grand entier $\leq x$

de parité d'un décodage défini [29] car, elles font intervenir les symboles d'erreur, $e^{u}_{i+\alpha_k-\alpha_j}$ dont les différences $(\alpha_k-\alpha_j)$ sont négatives. Cependant, la valeur de l'estimation $\hat{e}^{u}_{i+\alpha_k-\alpha_j}$, $(\alpha_k-\alpha_j)<0$ étant connue à l'instant courant i, elle peut être introduite dans les équations $A_{k,i}$ afin de réduire l'effet des symboles d'erreur $e^{u}_{i+\alpha_k-\alpha_j}$, $(\alpha_k-\alpha_j)<0$ qui interviennent dans (2.9) et par conséquent, d'améliorer la probabilité d'erreur [29]. Ce décodage avec rétroaction de la décision (« feedback decoding »), [2], [14], [29] s'effectue alors sur une longueur de contrainte (m+1). Les équations de contrôle de parité $A_{k,i}$ utilisées dans ce décodage avec rétroaction de la décision s'écrivent alors comme suit :

$$A_{k,i} = e^{p}_{i+\alpha_k} \oplus e^{u}_{i} \oplus \sum_{j=1}^{k-1}{\oplus}\, e^{u}_{i+\alpha_k-\alpha_j} \oplus \sum_{j=k+1}^{J}{\oplus}\,(e^{u}_{i+\alpha_k-\alpha_j} \oplus \hat{e}^{u}_{i+\alpha_k-\alpha_j}) \qquad (2.10)$$

où la dernière somme de (2.10) :

$$\sum_{j=k+1}^{J}{\oplus}\,(e^{u}_{i+\alpha_k-\alpha_j} \oplus \hat{e}^{u}_{i+\alpha_k-\alpha_j})$$

correspond alors à une rétroaction de la décision. Si celle-ci est «idéale», c'est-à-dire $(e^{u}_{i+\alpha_k-\alpha_j} = \hat{e}^{u}_{i+\alpha_k-\alpha_j})$, alors la dernière somme de (2.10) est égale à zéro de sorte que l'on obtient:

$$A_{k,i} = e^{p}_{i+\alpha_k} \oplus e^{u}_{i} \oplus \sum_{j=1}^{k-1} \oplus e^{u}_{i+\alpha_k-\alpha_j}, \; k = 1, 2, \ldots, J, \; i = 0, 1, 2, \ldots \qquad (2.11)$$

Les équations de parité exprimées par (2.11) correspondent bien à la définition d'un ensemble de J équations de parité orthogonales à e^{u}_{i}. Par conséquent, pour un ensemble donné de J équations de parité $A_{k,i}$, l'algorithme de décodage à seuil utilise la règle de décision suivante:

Choisir $\hat{e}^{u}_{i} = 1$ *si et seulement si*

$$\sum_{k=1}^{J} A_{k,i} > T, \qquad (2.12)$$

choisir $\hat{e}^{u}_{i} = 0$ *autrement*

où T est le seuil de décision égal à $\lfloor J/2 \rfloor$ pour une décision optimale. Le nombre total de symboles d'erreur contrôlés par l'ensemble des J équations $A_{k,i}$, $k = 1, \ldots, J$, données par (2.11) représente la longueur de contrainte effective, n_E, [2]. Puisque n_E symboles d'erreur sont contrôlés par les J équations $A_{k,i}$, avec cette règle de décision, le symbole e^{u}_{i} sera correctement déterminé si $T=\lfloor J/2 \rfloor$ symboles d'erreur ou moins parmi les n_E symboles d'erreur prennent la valeur binaire 1. La quantité $\lfloor J/2 \rfloor$ représente donc le pouvoir de correction des erreurs du canal. Le décodage à seuil détermine la valeur binaire du symbole d'erreur e^{u}_{i} en ne considérant que la première longueur de contrainte $(\alpha_J + 1)$

de la séquence codée reçue. Par conséquent, pour un décodage à seuil, le pouvoir de correction t sera dicté par la distance minimale, d_m, du code convolutionnel de la manière suivante [2]:

$$t = \left\lfloor \frac{d_m - 1}{2} \right\rfloor \tag{2.13}$$

Sachant aussi que pour un décodage à seuil ce pouvoir de correction est $\lfloor J/2 \rfloor$, il s'en suit que la distance minimale d'un CSOC est :

$$d_m = J + 1 \tag{2.14}$$

Notons qu'en général, si le code n'est pas complètement orthogonal, mais orthogonalisable, on peut encore former un ensemble de J équations de parité orthogonales, en construisant des combinaisons linéaires des symboles de syndrome [14]. Cependant, dans toute la suite de cette thèse, seuls les codes complètement orthogonaux ou CSOC sont considérés. Pour un CSOC, le schéma de principe d'un décodeur à seuil avec rétroaction en quantification ferme pour un code $r_c = 1/2$ est illustré à la Figure 2.4.

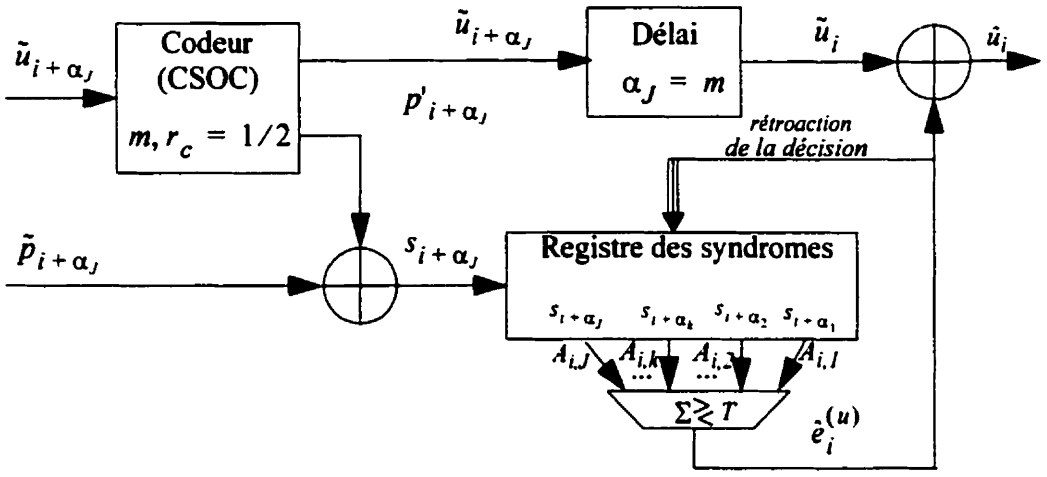


Figure 2.4 Schéma de principe d'un décodeur à seuil en quantification ferme

2.2 Décodeur à seuil à sortie pondérée (ou à sortie non-quantifiée)

Dans cette section, nous décrivons le processus de décodage à seuil en quantification pondérée pour les codes complètement orthogonaux. Pour simplifier l'analyse, on suppose que la sortie du canal est non-quantifiée, c'est-à-dire que les entrées du décodeur correspondent aux sorties non-quantifiées du canal.

L'algorithme de décodage à seuil, initialement proposé par Massey [14] et amélioré par la suite par Wu [24], [25] permet d'obtenir une estimation de la *probabilité a posteriori* (PAP) des bits décodés. L'estimation de la PAP donne une mesure de la probabilité que la décision sur le bit d'information courant est correcte. De plus, l'architecture d'un décodeur à seuil en quantification pondérée proposée par Lavoie *et al.* [18] et simplifiée par Gagnon *et al.* [19], utilise une *approximation de la probabilité a posteriori* (APAP) [26]. L'ensemble du développement de l'algorithme de décodage

itératif exposé dans cette thèse est basé sur ces derniers résultats.

2.2.1 Modèle du canal de transmission

Avant d'aborder le processus de décodage à seuil non-quantifié, il est nécessaire d'apporter quelques précisions concernant le canal de transmission. Dans la section précédente, le canal de transmission était du type binaire symétrique. Les entrées et les sorties du canal étaient alors binaires (0 ou 1). Dans ce qui suit, nous considérons le modèle d'un système de communication utilisant un canal non-quantifié. Ce modèle est illustré à la Figure 2.5.

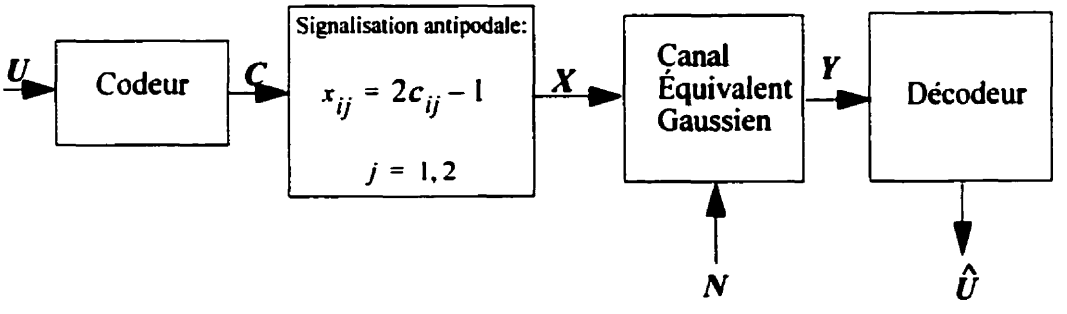


Figure 2.5 Modèle d'un système de communication avec un canal non-quantifié

La séquence de symboles codés $\boldsymbol{C}$ est transformée en une séquence de symboles antipodaux représentée par $\boldsymbol{X} = (x_0, x_1, \ldots)$. En suivant la même notation que dans la section précédente, les éléments x_i, $i = 0, 1, \ldots$, sont formés des symboles antipodaux $x_{i,1} = x_i^u$ et $x_{i,2} = x_i^p$. La transformation antipodale est obtenue selon la relation:

$$x_{i,j} = 2c_{i,j} - 1, j = 1,2 \text{ et } i = 0,1... \tag{2.15}$$

Avec cette notation, chaque élément x_i de la séquence X est exprimé par:

$$x_i = (x_i^u, x_i^p) = ((2u_i - 1), (2p_i - 1)) \tag{2.16}$$

où tout comme à la section précédente, u_i et p_i représentent, à l'instant i, le bit d'information et le symbole de parité respectivement. La séquence X se présente à l'entrée du canal gaussien équivalent. Tel qu'illustré à la Figure 2.6, ce canal équivalent gaussien est constitué d'un modulateur BPSK, d'un canal physique et d'un filtre adapté.

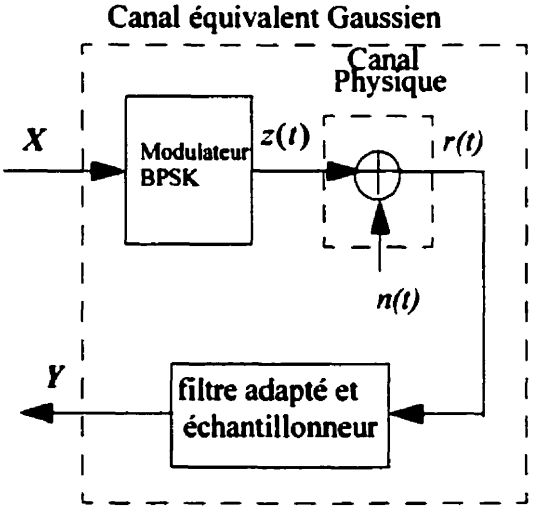


Figure 2.6 Modèle du canal équivalent Gaussien

Ainsi, la séquence X est modulée BPSK pour former un signal $z(t)$ qui est transmis dans un canal physique. Dans ce canal physique, le bruit, $n(t)$, est additif, blanc et gaussien de moyenne nulle et de spectre de densité de puissance bilatéral $(N_0/2)$ W/Hz.

Le signal reçu $r(t)$ s'écrit donc $r(t) = z(t) + n(t)$. Le signal $r(t)$ est démodulé en utilisant un filtre adapté qui produit à sa sortie la séquence $Y = (y_0, y_1 \ldots)$ où $y_0, y_1 \ldots$ sont des variables aléatoires gaussiennes. Cette séquence représente alors la sortie du canal équivalent gaussien.

Chaque élément y_i, $i = 0, 1 \ldots$, est composé des deux valeurs $y_{i,1} = y_i^u$ et $y_{i,2} = y_i^p$, où y_i^u et y_i^p sont donnés par les deux relations suivantes:

$$y_i^u = x_i^u + n_i^u \tag{2.17}$$

$$y_i^p = x_i^p + n_i^p . \tag{2.18}$$

de sorte que $Y = (y_0, y_1 \ldots) = ((y_0^u, y_0^p), (y_1^u, y_1^p), \ldots)$.

La séquence X de symboles codés transmis est donc modifiée par une séquence de bruit $N = (n_0, n_1, \ldots)$ où $n_i = (n_i^u, n_i^p)$ et où n_i^u et n_i^p sont les éléments de bruit ajoutés à x_i^u et à x_i^p respectivement. Les éléments de bruits n_i^u et n_i^p sont des variables aléatoires réelles et gaussiennes de moyenne nulle et de variance $N_0/2$.

La séquence Y est fournie au décodeur qui génère à sa sortie une séquence de bits d'information décodés $\hat{U}$.

2.2.2 Décodage à seuil et probabilité *a posteriori* (PAP)

Pour simplifier la description de la structure du décodeur non-quantifié supposons un code orthogonal de taux de codage $r_c = 1/2$ et ayant J équations de parité. La description de la structure du décodeur à seuil non-quantifié est basée sur une variante de la structure du décodeur à seuil en quantification ferme présentée à la section précédente.

Dans [14], Massey définit deux types de structures de décodeur à seuil: Type I et Type II. À la section 2.1.2, nous avons considéré une structure de décodage que Massey [14] appelle Type I. Le décodage de Type I utilise les J équations de parité $A_{j,i}$, $j = 1, 2, \ldots, J$, pour obtenir une décision sur le symbole d'erreur courant e_i^u. Les développements présentés dans cette section concernant la structure du décodeur à seuil avec *PAP* utilisent une variante de l'algorithme de Massey que l'on nomme Type II. Cette variante permet d'obtenir directement une estimation, $\hat{u}_i$, du bit décodé. Dans ce cas, la règle de décision fait appel à un ensemble de $(J+1)$ équations $B_{j,i}$, $j = 0, 1, \ldots, J$. Les équations $B_{j,i}$, $j = 1, 2, \ldots, J$, sont obtenues à partir des équations $A_{j,i}$ du décodeur de Type I en éliminant le symbole $\tilde{u}_i$ des équations $A_{j,i}$ alors que $B_{0,i} = \tilde{u}_i$. En se référant à l'équation (2.7) de la section précédente, les équations de parité $A_{j,i}$, $j = 1, 2, \ldots, J$, $i = 0, 1, 2, \ldots$, avec rétroaction s'écrivent:

$$A_{j,i} = \tilde{p}_{i+\alpha_j} \oplus \tilde{u}_i \oplus \sum_{k=1}^{j-1}{}^{\oplus}\, \tilde{u}_{i+\alpha_j-\alpha_k} \oplus \sum_{k=j+1}^{J}{}^{\oplus}\, \hat{u}_{i+\alpha_j-\alpha_k}, \quad (2.19)$$

où tout comme à la section précédente, $\tilde{u}_l$ et $\tilde{p}_l$ représentent respectivement les symboles d'information et de parité en quantification ferme reçus du canal. On définit les équations $B_{j,i}$ comme la somme modulo-2 des symboles reçus du canal en quantification ferme appartenant à l'équation $A_{j,i}$, $j = 1, 2, \ldots, J$, $i = 0, 1, 2, \ldots$, mais en excluant le symbole $\tilde{u}_i$:

$$\begin{aligned} B_{j,i} &= A_{j,i} \oplus \tilde{u}_i \\ &= \tilde{p}_{i+\alpha_j} \oplus \sum_{k=1}^{j-1}{}^{\oplus}\, \tilde{u}_{i+\alpha_j-\alpha_k} \oplus \sum_{k=j+1}^{J}{}^{\oplus}\, \hat{u}_{i+\alpha_j-\alpha_k}, \end{aligned} \quad (2.20)$$

Pour le cas particulier $j = 0$, on définit l'équation $B_{0,i} = \tilde{u}_i = u_i \oplus e_i^u$. On montre qu'à partir de l'équation (2.10) et du fait que $\tilde{u}_i = u_i \oplus e_i^u$, les équations $B_{j,i}$, $j = 1, 2, \ldots, J$, s'écrivent aussi comme suit [14]:

$$B_{j,i} = u_i \oplus \sum_{k=1}^{j-1}{}^{\oplus}\, e^u_{i+\alpha_j-\alpha_k} \oplus e^p_{i+\alpha_j} \oplus \sum_{k=j+1}^{J}{}^{\oplus}\, (e^u_{i+\alpha_j-\alpha_k} \oplus \hat{e}^u_{i+\alpha_j-\alpha_k}), \; j = 1, 2, \ldots, J,$$

$$i = 0, 1, 2, \ldots \quad (2.21)$$

Notons que le bit d'information à l'instant i, u_i, est présent dans chacune des $(J+1)$ équations $B_{j,i}$, $j = 0, 1, \ldots, J$, et tout comme les équations $A_{j,i}$, chaque symbole

d'erreur n'apparaît qu'une seule fois dans l'ensemble des $(J+1)$ équations $B_{j,i}$. Les équations $B_{j,i}$ forment donc un ensemble d'équations de syndrome modifiées orthogonales au bit u_i. Il importe de signaler qu'ici les équations $B_{j,i}$ ne sont pas des équations de contrôle de parité proprement dites mais, plutôt d'« inversion de parité ».

Dans ce qui suit, on décrit l'algorithme de décodage à seuil non-quantifié en utilisant les équations $B_{j,i}$ plutôt que les équations $A_{j,i}$. Dans le but de minimiser la probabilité d'erreur moyenne à la sortie du décodeur, l'algorithme de décodage détermine la valeur du bit d'information à décoder u_i en lui assignant la valeur binaire ξ de sorte que la probabilité conditionnelle:

$$Pr\{u_i = \xi | \{B_{j,i}\}\}, i = 0, 1, 2, \ldots \tag{2.22}$$

est maximisée.

Ainsi, il est clair que la règle de décision devient: *Choisir* $\hat{u}_i = 1$ *si et seulement si*

$$Pr\{u_i = 1 | \{B_{j,i}\}\} \geq Pr\{u_i = 0 | \{B_{j,i}\}\} \tag{2.23}$$

$\hat{u}_i = 0$ *autrement.*

En prenant le logarithme de part et d'autre de la relation (2.23) et en réarrangeant les termes, la règle de décision s'écrit aussi : *Choisir* $\hat{u}_i = 1$ *si et seulement si*

$$L(u_i | \{B_{j,i}\}) = \ln\left(\frac{Pr\{u_i = 1 | \{B_{j,i}\}\}}{Pr\{u_i = 0 | \{B_{j,i}\}\}}\right) \geq 0 \tag{2.24}$$

où $L(u_i|\{B_{j,i}\})$ est le *logarithme du rapport de vraisemblance* (LRV) de u_i conditionné sur l'ensemble $\{B_{j,i}\}$. Signalons que la valeur absolue de cette dernière, $|L(u_i|\{B_{j,i}\})|$, représente la fiabilité de la décision. Puisqu'il y a orthogonalité intrinsèque des équations d'inversion de parité, les symboles d'erreur qui interviennent dans les équations $B_{j,i}$ sont indépendants. Ainsi, en utilisant le théorème de Bayes, le LRV donné par (2.24) s'écrit:

$$L(u_i|\{B_{j,i}\}) = \sum_{j=1}^{J} \ln\left(\frac{P(B_{j,i}|u_i=1)}{P(B_{j,i}|u_i=0)}\right) + \ln\left(\frac{P(B_{0,i}|u_i=1)}{P(B_{0,i}|u_i=0)}\right) + \ln\left(\frac{P(u_i=1)}{P(u_i=0)}\right) \quad (2.25)$$

La valeur de $\ln\left(\frac{P(u_i=1)}{P(u_i=0)}\right)$ que l'on dénote aussi $L(u_i)$ représente l'information *a priori*. En général les probabilités de transmettre $u_i=1$ et $u_i=0$ sont égales à 1/2 de sorte que $L(u_i)=0$. Des simplifications peuvent être apportées à (2.25) en considérant les observations suivantes. Selon l'équation (2.21), on remarque que $B_{j,i}$ prend la valeur binaire 0 lorsque u_i est égal à 0 et que le nombre de symboles d'erreur de valeur binaire «1» dans l'équation $B_{j,i}$ est pair ou lorsque $u_i=1$ et que le nombre de symboles d'erreur de valeur binaire «1» dans $B_{j,i}$ est impair. Il est évident que l'inverse est aussi vrai, c'est-à-dire que $B_{j,i}=1$ si $u_i=0$ et que le nombre de symboles d'erreur de valeur binaire «1» dans $B_{j,i}$ est impair ou lorsque $u_i=1$ et qu'il y a un nombre pair de symboles d'erreur de valeur binaire «1» dans $B_{j,i}$. Définissons maintenant

$\gamma_{j,i} \triangleq (1-\rho_{j,i})$, $j = 0, \ldots, J$, $i = 0, 1, \ldots$ comme étant la probabilité d'avoir un nombre impair de symboles d'erreur de valeur binaire «1» dans $B_{j,i}$ excluant le bit d'information u_i. Il s'en suit alors que:

$$P(B_{j,i} = 0 | u_i = 1) = P(B_{j,i} = 1 | u_i = 0) = \gamma_{j,i} \tag{2.26}$$

$$P(B_{j,i} = 0 | u_i = 0) = P(B_{j,i} = 1 | u_i = 1) = \rho_{j,i} \tag{2.27}$$

À partir de (2.26) et (2.27), l'équation (2.25) devient [14]

$$L(u_i | \{B_{j,i}\}) = \sum_{j=1}^{J} (2B_{j,i} - 1) w_{j,i} + (2B_{0,i} - 1) w_{0,i} + L(u_i) \tag{2.28}$$

où les facteurs de pondération ou de poids $w_{j,i}$ sont donnés par $w_{j,i} = \ln\left(\frac{\rho_{j,i}}{\gamma_{j,i}}\right)$, $j = 0, \ldots, J$. Notons que la valeur de $\gamma_{0,i}$ représente la probabilité de transition du canal pour le symbole u_i telle que définie par (2.3). La valeur du poids $w_{j,i}$ se définit comme étant le LRV de l'équation $B_{j,i}$ excluant le bit d'information u_i et représente la fiabilité de l'équation $B_{j,i}$.

2.2.3 Calcul des poids

Massey [14] définit les poids $w_{j,i}$ en supposant que la rétroaction de la décision est idéale, c'est-à-dire $(e^{u}_{i+\alpha_t-\alpha_j} = \hat{e}^{u}_{i+\alpha_t-\alpha_j})$. Selon la technique utilisée dans [14], les

poids $w_{j,i}$, $j = 1, 2, \ldots, J$ sont donnés par l'équation suivante:

$$w_{j,i} = \ln\left(\frac{\rho_{j,i}}{\gamma_{j,i}}\right) = -\ln\left(\frac{P\{(B_{j,i} \oplus u_i) = 1\}}{P\{(B_{j,i} \oplus u_i) = 0\}}\right) = -L(B_{j,i} \oplus u_i)$$

$$= -L\left(\sum_{k=1}^{j-1} \oplus e^u_{i+\alpha_j-\alpha_k} \oplus e^p_{i+\alpha_j} \middle| y^u_{i+\alpha_j-\alpha_1}, \ldots, y^u_{i+\alpha_j-\alpha_{(j-1)}}, y^p_{i+\alpha_j} \right) \quad (2.29)$$

où $j = 1, 2, \ldots, J$, tandis que pour $w_{0,i}$, on obtient:

$$w_{0,i} = \ln\left(\frac{\rho_{0,i}}{\gamma_{0,i}}\right) = -L(B_{0,i} \oplus u_i) = -L(e^u_i \middle| y^u_i) \quad (2.30)$$

Le lemme suivant permet d'obtenir une expression des poids $w_{j,i}$ en fonction du LRV de chaque symbole d'erreur qui intervient dans les équations d'inversion de parité $B_{j,i}$ excluant le symbole d'information u_i.

Lemme. Soit e_{l1}, e_{l2}, ..., e_{lj}, un ensemble de variables aléatoires binaires statistiquement indépendantes telles que $P(e_{lk} = 1|y_{lk}) = 1 - P(e_{lk} = 0|y_{lk})$. Alors la probabilité γ_{jl} d'avoir un nombre impair de variables aléatoires $e_{lk} = 1$ est donnée par :

$$\gamma_{jl} = \frac{1}{2}\left[1 - \prod_{k=1}^{j}(1 - 2P(e_{lk} = 1|y_{lk}))\right] \quad (2.31)$$

Le lecteur trouvera la preuve de ce lemme dans [14]. Sachant que le LRV du $k^{ième}$

symbole d'erreur e_{lk} à l'instant l est :

$$L(e_{lk}|y_{lk}) = \ln\left(\frac{P(e_{lk}=1|y_{lk})}{P(e_{lk}=0|y_{lk})}\right) \tag{2.32}$$

la probabilité $P(e_{lk}=1|y_{lk})$ s'écrit comme suit :

$$P(e_{lk}=1|y_{lk}) = \frac{\exp\{L(e_{lk}|y_{lk})\}}{1+\exp\{L(e_{lk}|y_{lk})\}} \tag{2.33}$$

En considérant la probabilité donnée par (2.33), l'application du lemme précédent au calcul des poids $w_{j,i}$ donnée par (2.29) permet d'écrire :

$$w_{j,i} = 2\,\text{atanh}\left(\tanh\left(\frac{-L(e^p_{i+\alpha_j}|y^p_{i+\alpha_j})}{2}\right)\prod_{k=1}^{j-1}\tanh\left(\frac{-L(e^u_{i+\alpha_j-\alpha_k}|y^u_{i+\alpha_j-\alpha_k})}{2}\right)\right) \tag{2.34}$$

où $L(e^u_l|y^u_l)$ et $L(e^p_l|y^p_l)$ sont les LRV de e^u_l et e^p_l respectivement. Les valeurs des LRV $L(e^u_l|y^u_l)$ et $L(e^p_l|y^p_l)$ sont obtenues en considérant le canal équivalent gaussien présenté à la section 2.2.1. Pour une modulation BPSK, le LRV du symbole d'erreur $e_{l,k}$ s'obtient de la façon suivante. Lorsque la valeur du symbole codé x_{lk}, k=1 et 2, transmis à l'instant l=0, 1, 2, ..., prend la valeur -1, le symbole d'erreur binaire correspondant e_{lk} est égal à 1 si la valeur obtenue à la sortie du filtre adapté échantillonné à l'instant l, y_{lk}, est supérieure à 0. De la même manière le symbole d'erreur e_{lk} est aussi égal à 1, lorsque x_{lk}=1 et que y_{lk}<0. Par conséquent, on peut écrire la relation suivante :

$$P(e_{lk} = 1|y_{lk}) = \frac{\exp\left(-\frac{E_c(|y_{l,k}| + 1)^2}{N_0}\right)}{\exp\left(-\frac{E_c(|y_{l,k}| + 1)^2}{N_0}\right) + \exp\left(-\frac{E_c(|y_{l,k}| - 1)^2}{N_0}\right)} \tag{2.35}$$

En suivant le même raisonnement, on détermine la probabilité conditionnelle $P(e_{lk} = 0|y_{lk}) = 1 - P(e_{lk} = 1|y_{lk})$ comme suit :

$$P(e_{lk} = 0|y_{lk}) = \frac{\exp\left(-\frac{E_c(|y_{l,k}| - 1)^2}{N_0}\right)}{\exp\left(-\frac{E_c(|y_{l,k}| + 1)^2}{N_o}\right) + \exp\left(-\frac{E_c(|y_{l,k}| - 1)^2}{N_o}\right)} \tag{2.36}$$

Le LRV du symbole d'erreur $e_{l,k}$, $k = 1, 2$, à l'instant $l = 0, 1, 2, \ldots$ s'obtient en combinant (2.35) et (2.36). Pour une modulation BPSK, le LRV du symbole d'erreur $e_{l,k}$, est donné par [14]:

$$L(e_{l,k}|y_{lk}) = \ln\left(\frac{P(e_{l,k} = 1|y_{l,k})}{P(e_{l,k} = 0|y_{l,k})}\right) = \ln\left(\frac{\exp\left(-\frac{E_c(|y_{l,k}| + 1)^2}{N_0}\right)}{\exp\left(-\frac{E_c(|y_{l,k}| - 1)^2}{N_0}\right)}\right) \tag{2.37}$$

$$= -4\frac{E_c}{N_0}|y_{l,k}|$$

où $y_{l,k}$ est la sortie du filtre adapté correspondant au symbole $x_{l,k}$ et où l'énergie par symbole codé $E_c = r_c E_b$ dans lequel E_b est l'énergie du bit d'information et r_c est le

taux de codage. Ici, $r_c = 1/2$ de sorte que $E_c = (1/2)E_b$. En combinant (2.34) à (2.37), il est clair que les poids $w_{j,i}$ sont positifs.

Jusqu'à présent, nous avons considéré que les termes de la rétroaction n'intervenaient pas dans le calcul des poids $w_{j,i}$. Toutefois, puisque la valeur pondérée obtenue à la sortie du décodeur, $L(u_l|\{B_{j,l}\})$, $l < i$, représente une estimation améliorée du bit u_l, elle peut être utilisée en rétroaction dans le calcul des poids $w_{j,i}$. L'utilisation de cette rétroaction dans le calcul des poids $w_{j,i}$ devrait alors contribuer à l'amélioration de l'estimation des autres bits d'information [19]. Si l'on se réfère à l'équation (2.21), l'équation (2.34) devient alors:

$$w_{j,i} = 2\,\mathrm{atanh}\left(\tanh\left(\frac{-L(e^p_{i+\alpha_j}|y^p_{i+\alpha_j})}{2}\right)\prod_{k=1}^{j-1}\tanh\left(\frac{-L(e^u_{i+\alpha_j-\alpha_k}|y^u_{i+\alpha_j-\alpha_k})}{2}\right)\right. \qquad (2.38)$$
$$\left.\prod_{k=j+1}^{J}\tanh\left(\frac{-L(e^u_{i+\alpha_j-\alpha_k}\oplus\hat{e}^u_{i+\alpha_j-\alpha_k})}{2}\right)\right)$$

En reprenant le même raisonnement que précédemment, on obtient que le LRV $L(e^u_{i+\alpha_j-\alpha_k}\oplus\hat{e}^u_{i+\alpha_j-\alpha_k}) = -|L(u_{i+\alpha_j-\alpha_k}|\{B_{j,i+\alpha_j-\alpha_k}\})|$, $j < k$, ce qui représente la fiabilité de la décision sur le bit décodé à la sortie du décodeur (voir Annexe I).

2.2.4 Définition de l'opération «add-min»

Pour simplifier davantage la complexité du décodeur à seuil non-quantifié, une approximation du LRV du bit d'information à décoder est introduite en utilisant

l'opération «add-min» [26]. Cette approximation a pour but de simplifier le calcul des poids $w_{j,i}$. À l'annexe II, on montre qu'une approximation du LRV de la somme modulo-2 de deux variables aléatoires binaires indépendantes ξ_1 et ξ_2 s'obtient par :

$$\begin{aligned} -L(\xi_1 \oplus \xi_2) &= 2\,\mathrm{atanh}\left(\tanh\left(\frac{-L(\xi_1)}{2}\right)\tanh\left(\frac{-L(\xi_2)}{2}\right)\right) \\ &\approx sign\{L(\xi_1)\}sign\{L(\xi_2)\}min\{|L(\xi_1)|, |L(\xi_2)|\} \end{aligned} \tag{2.39}$$

En utilisant l'approximation obtenue en (2.39), nous définissons maintenant l'opération «add-min» représentée par l'opérateur $\Diamond$ comme étant:

$$L(\xi_1) \Diamond L(\xi_2) = -sign(L(\xi_1))sign(L(\xi_2))min(|L(\xi_1)|, |L(\xi_2)|) \tag{2.40}$$

Le cas où il y a $N > 2$ variables aléatoires binaires indépendantes, $\xi_1, \xi_2, \ldots \xi_N$, (2.40) se généralise facilement de la manière suivante :

$$\sum_{i=1}^{N}{}^{\Diamond} L(\xi_i) = (-1)^{N+1}\prod_{i=1}^{N} sign(L(\xi_i)) \min_{1 \le i \le N}\{|L(\xi_i)|\} \tag{2.41}$$

Une approximation des poids $w_{j,i}$ est alors obtenue par l'application de l'opération add-min. Dans ce cas, puisque les poids $w_{j,i}$ sont positifs, il est alors clair que l'approximation par l'opération add-min revient simplement à prendre le minimum des LRV des symboles reçus du canal qui composent l'équation $B_{j,i}$ [26]:

$$w_{j,i} \approx 4\frac{E_c}{N_o}\left|y^p_{i+\alpha_j}\right| \lozenge \sum_{k=1}^{j-1}\lozenge 4\frac{E_c}{N_o}\left|y^u_{i+\alpha_j-\alpha_k}\right| \lozenge \sum_{k=j+1}^{J}\lozenge \left|L(u_{i+\alpha_j-\alpha_k}|\{B_{j,i+\alpha_j-\alpha_k}\})\right|$$

$$= \min_{\substack{k=1,\ldots,J \\ k\neq j}} J\left(4\frac{E_c}{N_o}\left|y^p_{i+\alpha_j}\right|, 4\frac{E_c}{N_o}\left|y^u_{i+\alpha_j-\alpha_k}\right|, \left|L(u_{i+\alpha_j-\alpha_k}|\{B_{j,i+\alpha_j-\alpha_k}\})\right|\right) \tag{2.42}$$

En tenant compte du fait que les poids $w_{j,i}$ sont strictement positifs et que les termes $(2B_{j,i}-1)$ de l'équation (2.28) ne représentent que les signes des LRV $L(B_{j,i}|u_i)$, on peut écrire:

$$L(u_i|\{B_{j,i}\}) \approx \lambda_i = y^u_i + \sum_{j=1}^{J}\left(y^p_{i+\alpha_j} \lozenge \sum_{k=1}^{j-1}\lozenge y^u_{i+\alpha_j-\alpha_k} \lozenge \sum_{k=j+1}^{J}\lozenge \lambda_{i+\alpha_j-\alpha_k}\right)$$

$$= y^u_i + \sum_{j=1}^{J}\psi_{j,i} \tag{2.43}$$

où, ici, on a supposé que le terme *a priori* de u_i, $L(u_i)$, est égal à zéro. Le terme $\psi_{j,i}$ représente la valeur approximative de chaque équation $B_{j,i}$. Les résultats de simulation obtenus dans [18] et [19] démontrent la validité de cette approximation obtenue par l'opération add-min. La Figure 2.7 illustre, pour le code J=4 de la Figure 2.2 (b), le décodeur à seuil à sortie pondérée avec rétroaction utilisant l'opération add-min.

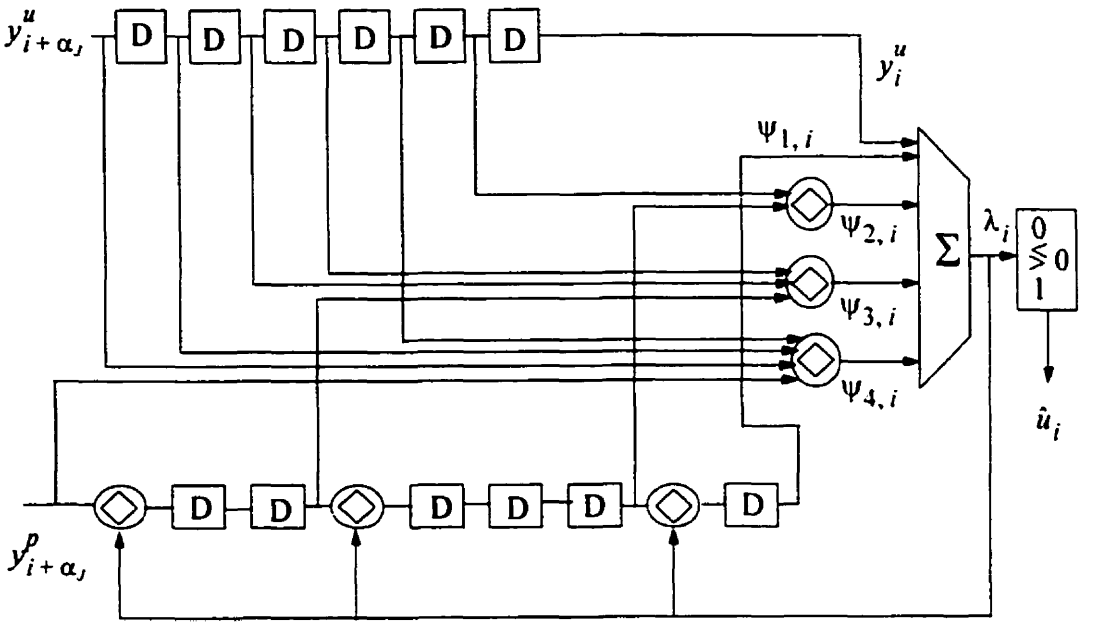


Figure 2.7 Décodeur à seuil non quantifié utilisant l'opération add-min

2.2.5 Décodeur à seuil et valeur extrinsèque

Le décodeur à seuil à sortie pondérée génère une valeur réelle servant à la décision finale sur le bit d'information u_i. Cette valeur réelle correspond au LRV du bit d'information u_i conditionné sur un ensemble fini de valeurs réelles observées à la sortie du canal équivalent gaussien. Le décodeur à seuil à sortie non-quantifiée est donc une version sous-optimale de l'algorithme de décodage MAP (Maximum a posteriori) [14], [28], [29]. Un décodeur de symbole utilisant l'algorithme MAP détermine une valeur réelle (non-quantifiée) correspondant au LRV, $L(\hat{u}_i) = L(u_i|Y)$. Pour un code systématique, cette valeur se subdivise en trois termes [28]:

$$L(\hat{u}_i) = L_e(\hat{u}_i) + L_c y_i^u + L(u_i) \tag{2.44}$$

Ainsi, ce décodeur utilise, lorsque disponible, une valeur a priori $L(u_i)$, une valeur $L_c y_i^u$ provenant du canal et une valeur extrinsèque $L_e(\hat{u}_i)$ [28], [4]. Pour le décodage à seuil, la sortie non-quantifiée devient une approximation de $L(\hat{u}_i)$. Par conséquent la sortie non-quantifiée du décodeur à seuil se subdivise aussi en trois termes. En utilisant la même notation que dans [28], l'équation (2.28) devient:

$$L(\hat{u}_i) \approx L(u_i | \{B_j\}) = \tilde{L}_e(\hat{u}_i) + L_c y_i^u + L(u_i) \tag{2.45}$$

où la valeur extrinsèque est donnée par $\tilde{L}_e(\hat{u}_i) = \sum_{j=1}^{J} (2B_{j,i} - 1) w_{j,i}$ et où $L_c = 4\frac{E_c}{N_0}$.

La valeur extrinsèque $\tilde{L}_e(\hat{u}_i)$ ne dépend pas de l'information reçue $\tilde{u}_i$ et représente l'effet des symboles d'erreur orthogonaux à e_i^u sur la valeur de la décision finale $\hat{u}_i$.

On montre alors que le décodeur à seuil à sortie non-quantifiée ou pondérée offre une structure permettant son application dans un processus itératif de décodage [30]. En effet, la valeur de l'information extrinsèque pourrait être utilisée dans une itération subséquente comme mise à jour de l'information a priori que l'on a supposée jusqu'ici nulle. Ainsi, la première itération utilise l'information du canal en supposant que l'information *a priori* est nulle. La deuxième itération utilise la valeur de l'information extrinsèque comme nouvelle information *a priori* en plus d'utiliser l'information du canal. Le processus itératif continue de la même manière jusqu'à ce qu'aucune amélioration des performances

d'erreur ne soit possible pour toute itération supplémentaire. Dans les chapitres suivants, nous développons davantage le rôle de la valeur de l'information extrinsèque dans le processus de décodage itératif à seuil à sortie non-quantifiée utilisant des codes doublement orthogonaux.

CHAPITRE 3: PROCESSUS ITÉRATIF DE DÉCODAGE À SEUIL À SORTIE PONDÉRÉE SANS ENTRELACEMENT

La technique de décodage « Turbo » conventionnelle présentée dans [4] et [28] est généralement de grande complexité. Dans le but de réduire la complexité du processus de décodage « Turbo » conventionnel, les auteurs de [6] et [32] ont apporté des simplifications aux algorithmes MAP et SOVA. En plus de la grande complexité engendrée par l'utilisation de ces deux algorithmes de décodage, la technique « Turbo » conventionnelle a aussi l'inconvénient d'introduire une latence importante dans tout le processus de décodage. Cette latence est due à la présence des entrelaceurs au niveau du codage et du décodage, au nombre requis d'itérations pour atteindre une probabilité d'erreur spécifique ainsi qu'au délai engendré par le décodage proprement dit. Évidemment, cette latence correspond à un délai temporel pouvant limiter le type d'application. Dans des applications telles que les communications sans fil, le délai de transmission est un critère important de performance à minimiser. Cette latence implicite à la technique « Turbo » conventionnelle peut diminuer les possibilités d'applications de cette méthode de correction des erreurs du canal.

Dans cette section, on présente une autre technique de codage et de decodage itératif de faible complexité et ayant une latence réduite. Cette nouvelle technique consiste à utiliser un codage convolutionnel orthogonal pour lequel, le décodage à seuil est facilement applicable. Le choix du décodage à seuil est dicté par sa faible complexité et par le fait que tout comme les algorithmes MAP et SOVA, c'est une technique de

décodage symbole par symbole. Cette technique de décodage itératif se distingue des autres méthodes conventionnelles de décodage «Turbo» [4], [13] car, elle ne nécessite aucun entrelacement ni au codage et ni au décodage. Le codage de l'information est effectué par un seul codeur et chaque itération du processus de décodage ne nécessite qu'un seul décodeur sans aucun entrelaceur. Par conséquent, on peut espérer une réduction de la complexité et de la latence de ce processus itératif de décodage.

À chaque itération du processus de décodage « Turbo » conventionnel, le logarithme du rapport de vraisemblance de chaque symbole d'information décodé est déterminé en utilisant un ensemble d'observables indépendants. Cette condition d'indépendance des observables dans ce processus de décodage est obtenue par l'entrelacement (délacement) des symboles d'information. Une augmentation de la taille des entrelaceurs et une procédure d'entrelacement adéquate entraînent une diminution de la probabilité d'erreur [31], [32]. Cette amélioration des performances d'erreur est quasi proportionnelle à la taille de l'entrelacement [16]. Évidemment, le processus de décodage à seuil itératif sans entrelacement nécessite aussi cette condition d'indépendance entre les observables.

Pour obtenir cette condition d'indépendance sans introduire un entrelacement au codage, des propriétés supplémentaires sur le code sont nécessaires. Les codes *convolutionnels doublement orthogonaux* (en anglais, Convolutional Self Doubly Orthogonal Code, CSO^2C) respectent ces conditions d'indépendance des observables sur au moins les deux premières itérations du processus de décodage [33-36]. Grâce aux conditions d'orthogonalité implicites à ces codes, ceux-ci peuvent être facilement décodés par l'algorithme de décodage à seuil. La propriété de *double orthogonalité* des CSO^2C

contribue à la réduction de la complexité et de la latence du décodeur, tout en conservant un caractère indépendant entre les observables utilisés dans tout le processus de décodage [33]. Évidemment, cette nouvelle notion de double orthogonalité des codes convolutionnels pourrait se généraliser aussi au concept de multiple orthogonalité des codes convolutionnels.

3.1 Décodage à seuil itératif à sortie non-quantifiée

L'application itérative de l'algorithme de décodage à seuil peut se faire sans avoir recours à des modifications majeures de l'algorithme de décodage à seuil. Le processus de décodage itératif utilisant le décodage à seuil à sortie non-quantifiée peut être décrit comme suit. La valeur extrinsèque $L_e^{(\mu)}(\hat{u}_i) = \sum_{j=1}^{J} \psi_{j,i}^{(\mu)}$ de l'itération (μ) est formée d'une combinaison de la sortie du décodeur à l'itération (μ-1), à l'instant i dénotée $\lambda_i^{(\mu-1)}$, avec des symboles de parité reçus du canal. Par la suite à l'itération (μ), le décodeur combine la valeur extrinsèque $L_e^{(\mu)}(\hat{u}_i)$ avec le symbole d'information reçu, y_i^u pour produire à la sortie la valeur non-quantifiée $\lambda_i^{(\mu)}$. La Figure 3.1 illustre le principe du décodage itératif à seuil à sortie non-quantifiée pour M itérations. En se référant à (2.43), la sortie non-quantifiée, $\lambda_i^{(\mu)}$, de l'itération μ et à l'instant i s'écrit comme suit:

$$\lambda_i^{(\mu)} = y_i^u + \sum_{j=1}^{J} \left(y_{i+\alpha_j}^p \Diamond \sum_{k=1}^{j-1} \Diamond \lambda_{i+\alpha_j-\alpha_k}^{(\mu-1)} \Diamond \sum_{k=j+1}^{J} \Diamond \lambda_{i+\alpha_j-\alpha_k}^{(\mu)} \right) \tag{3.1}$$

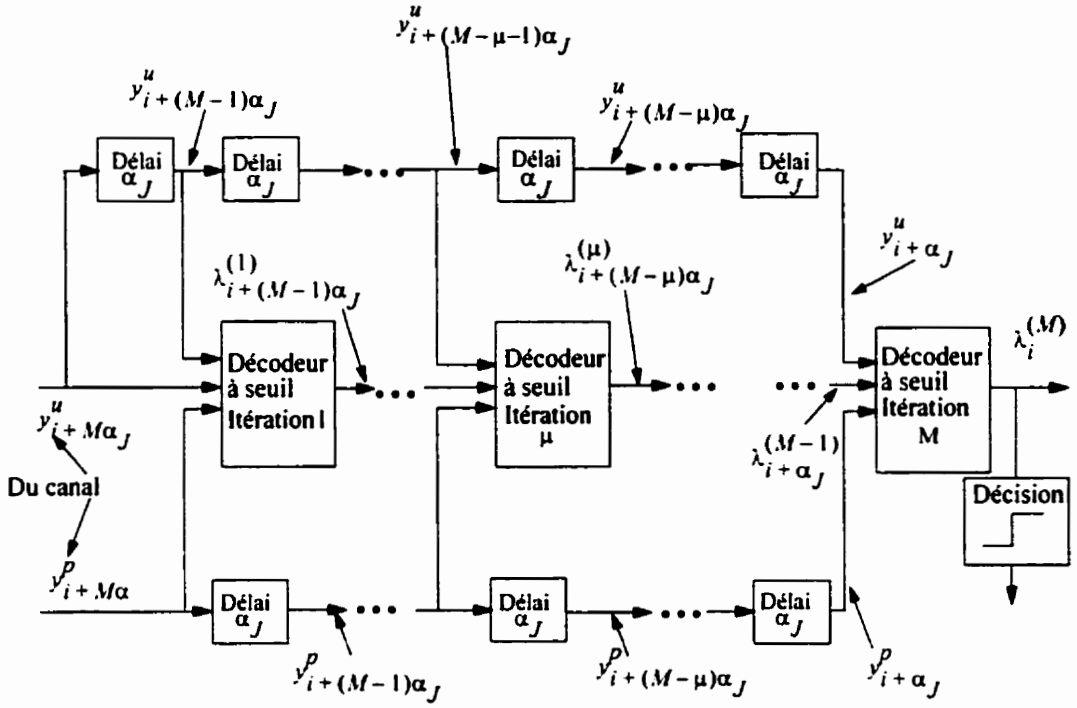


Figure 3.1 Schéma de principe du décodage à seuil itératif à sortie non-quantifiée sans entrelacement pour M itérations

où les termes entre parenthèses correspondent aux J équations de contrôle de parité modifiées, $\psi_{j,i}^{(\mu)}$, de l'itération μ lesquelles s'écrivent comme suit :

$$\psi_{j,i}^{(\mu)} = y_{i+\alpha_j}^{p} \Diamond \sum_{k=1}^{j-1}\Diamond \lambda_{i+\alpha_j-\alpha_k}^{(\mu-1)} \Diamond \sum_{k=j+1}^{J}\Diamond \lambda_{i+\alpha_j-\alpha_k}^{(\mu)} \tag{3.2}$$

de sorte que,

$$\lambda_i^{(\mu)} = y_i^u + \sum_{j=1}^{J} \psi_{j,i}^{(\mu)} = y_i^u + L_e^{(\mu)}(\hat{u}_i) \tag{3.3}$$

3.2 Définition des Codes Convolutionnels Doublement Orthogonaux

L'application récursive de (3.1) pour les deux premières itérations, c'est-à-dire pour $\mu=1$ et $\mu=2$ donne pour l'étape de décodage $\mu=2$, le résultat suivant :

$$\begin{aligned}\lambda_i^{(2)} &= y_i^u + \sum_{j=1}^{J} \psi_{j,i}^{(2)} \quad = y_i^u + L_e^{(2)}(\hat{u}_i) \\ &= y_i^u + \sum_{j=1}^{J} \left(y_{i+\alpha_j}^p \Diamond \sum_{k=j+1}^{J} \Diamond \lambda_{i+\alpha_j-\alpha_k}^{(2)} \quad \Diamond \right. \\ &\qquad \left. \sum_{k=1}^{j-1} \Diamond \left(y_{i+\alpha_j-\alpha_k}^u + \sum_{l=1, l\neq k}^{J} \left(y_{i+\alpha_j-\alpha_k+\alpha_l}^p \quad \Diamond \right. \right. \right. \\ &\qquad \left. \left. \left. \sum_{q=1, q\neq j}^{l-1} \Diamond y_{i+(\alpha_j-\alpha_k)-(\alpha_q-\alpha_l)}^u \quad \Diamond \sum_{q=l+1}^{J} \Diamond \lambda_{i+(\alpha_j-\alpha_k)-(\alpha_q-\alpha_l)}^{(1)} \right) \right) \right)\end{aligned} \tag{3.4}$$

La condition d'indépendance entre les variables observées sur les deux premières itérations doit être respectée. Cela revient aussi à maintenir l'orthogonalité sur les deux premières itérations du processus de décodage. Ainsi, pour s'assurer que l'équation (3.4) soit bien une fonction de variables indépendantes, des conditions supplémentaires sur l'ensemble des valeurs entières $\{\alpha_j\}$ décrivant le code convolutionnel sont nécessaires. Ces conditions supplémentaires à satisfaire sont décrites par la Définition 1.

Définition 1. Un code convolutionnel de taux de codage $r_c = 1/2$ est doublement orthogonal, si les trois conditions suivantes sont satisfaites :

1) *les différences* $(\alpha_j - \alpha_k)$ *sont distinctes;*

2) *les différences de différences* $(\alpha_j - \alpha_k) - (\alpha_q - \alpha_l)$ *sont distinctes des différences*;

3) *les différences de différences* $(\alpha_j - \alpha_k) - (\alpha_q - \alpha_l)$ *sont distinctes*;

pour les toutes les combinaisons des indices (j, k, l, q), $j \neq k, l \neq q, l \neq k, q \neq j$.

On peut montrer que la condition 3 de la Définition 1, implique les deux premières [21]. Les conditions $l \neq k$ et $q \neq j$ appliquées à la deuxième somme de valeurs réelles et à la troisième somme *addmin* de (3.4) permettent d'éliminer toutes répétitions des symboles de parité $y^{p}_{i+\alpha_j}$ et des symboles d'information $y^{u}_{i+(\alpha_j-\alpha_k)-(\alpha_q-\alpha_l)}$.

3.2.1 Définition des codes convolutionnels doublement orthogonaux au sens large

Avec le codeur de structure simple considéré jusqu'à présent, il est impossible d'obtenir des éléments différences de différences, $(\alpha_j - \alpha_k) - (\alpha_q - \alpha_l)$, qui soient complètement distincts. En effet, toutes les permutations inévitables des indices k et q produisent les mêmes éléments différences de différence : $(\alpha_j - \alpha_k) - (\alpha_q - \alpha_l) = (\alpha_j - \alpha_q) - (\alpha_k - \alpha_l)$. Ces répétitions d'observables ne peuvent être éliminées du processus itératif sans l'apparition de problèmes de propagation des erreurs dans le reste du processus de décodage. Pour éliminer ces répétitions du processus itératif, plusieurs équations $\psi^{(\mu)}_{j,i}$ d'une itération μ donnée ne peuvent plus être considérées dans l'évaluation de la valeur de l'information extrinsèque $L^{(\mu)}_e(\hat{u}_i)$ du symbole décodé à cette

itération μ. D'autre part, on notera qu'avec un décodage à seuil en quantification ferme (décodage à logique majoritaire), au moins $(J/2+1)$ symboles de syndrome doivent être observés pour éviter tout problème de propagation des erreurs ou de convergence. De la même manière, on en déduit qu'il faut aussi observer à chaque itération au moins $(J/2+1)$ équations $\psi_{j,i}^{(\mu)}$ pour que le processus de décodage itératif utilisant le décodage à seuil à sortie non-quantifiée puisse converger. Si l'on retient cet argument, il serait impossible de déterminer des conditions favorables sur les indices (j, k, l, q) pour que le processus de décodage itératif converge pour un code CSO^2C de taux de codage $r_c = 1/2$. Nous dirons que le processus de décodage itératif converge lorsqu'une amélioration des performances d'erreur est possible par l'ajout d'une itération supplémentaire.

En présence de ces répétitions inévitables et indésirables d'observables à la deuxième itération du processus de décodage, les codes convolutionnels alors obtenus sont appelés *codes convolutionnels doublement orthogonaux au sens large* (CSO^2C-WS, «Convolutional Self-Doubly Orthogonal Codes in the Wide Sense») et sont décrits par la Définition 2.

Définition 2. Un code convolutionnel de taux de codage $r_c = 1/2$ est doublement orthogonal au *sens large*, si et seulement si les indices de position $\{\alpha_j\}$ sont tels que :

$(\alpha_j - \alpha_k) - (\alpha_q - \alpha_l)$ *sont distincts pour tout* (j, k, l, q), $j \neq k, l \neq q, l \neq k, q \neq j$ *sauf pour les permutations inévitables.*

3.2.2 Construction des codes *CSO²C-WS*

En utilisant la Définition 2, une recherche de codes CSO²C-WS a été effectuée par les auteurs de [37], [22]. Puisque la longueur de contrainte, α_J, du code représente la latence d'une itération du processus de décodage, les codes CSO²C-WS doivent être construits de sorte que la valeur α_J soit minimale. Pour ce faire, les méthodes proposées dans [37], [22] passent toutes par deux étapes : la première est la génération d'un ensemble valide de valeurs entières $\{\alpha_1, \alpha_2, ...,\alpha_J\}$ respectant la Définition 2 et la seconde est la réduction de la longueur de contrainte (span), α_J. Pour des codes CSO²C-WS, la procédure de réduction adoptée dans [21] et [22] utilise les propriétés suivantes : toutes opérations d'addition et de multiplication appliquées sur $\{\alpha_1, \alpha_2, ...,\alpha_J\}$ conservent la propriété d'orthogonalité d'ordre 2. La procédure de réduction consiste alors à effectuer des opérations élémentaires modulo n où n, un entier, décroît graduellement jusqu'à ce que la plus grande réduction de la longueur de contrainte α_J soit obtenue.

La géométrie projective [2], [39] généralement utilisée pour générer des codes CSOC peut être aussi étendue à la recherche des codes CSO²C [37], [38]. Il a été démontré dans [38] que les J entiers α_i, $i = 1, ..., J$, qui décrivent un code CSO²C-WS s'obtiennent en déterminant les $J=p^s+1$ exposants possibles des points d'une ligne $\eta_1 + \eta_2\alpha$ définie dans une géométrie projective $PG(4, p^s)$ où p est un nombre premier et où s est un nombre entier qui satisfait la relation $J=p^s+1$. Cependant, même en appliquant une méthode de

réduction de la longueur de contrainte, les résultats obtenus avec cette méthode algébrique montrent que les codes ainsi générés n'ont pas toujours une longueur de contrainte, α_J, (span) minimale.

Tableau 3.1: Résultats de la construction de codes CSO^2-WS [21]

J	Codes	J	Codes
5	0 1 24 37 41 0 4 17 40 41	8	0 43 139 322 422 430 441 459 0 18 29 37 137 320 416 459 0 13 102 146 393 454 459 461 0 2 7 68 315 359 448 461 0 22 73 87 209 427 473 474 0 1 47 265 387 401 452 474 0 9 44 82 110 342 451 475 0 24 133 365 393 431 466 475
6	0 1 17 70 95 100 0 5 30 83 99 100 0 2 13 72 97 102 0 5 30 89 100 102	9	0 9 21 395 584 767 871 899 912 0 13 41 145 328 517 891 903 912 0 2 140 267 304 822 858 916 943 0 27 85 121 639 676 803 941 943 0 34 48 60 386 743 836 935 979 0 44 143 236 593 919 931 945 979 0 4 155 191 216 330 857 969 999 0 30 142 669 783 808 844 995 999 0 101 118 269 459 944 1021 1025 1044 0 19 23 100 585 775 926 943 1044 0 7 23 241 419 740 980 1034 1046 0 12 66 306 627 805 1023 1039 1046
7	0 1 9 71 177 215 228 0 13 51 157 219 227 228 0 1 13 83 196 201 229 0 28 33 146 216 228 229 0 1 14 74 178 227 231 0 4 53 157 217 230 231 0 1 5 141 193 216 232 0 16 39 91 227 231 232 0 12 54 71 200 223 232 0 9 32 161 178 220 232	10	0 27 93 503 600 1247 1646 1714 1825 1835 0 10 121 189 588 1235 1332 1742 1808 1835 0 52 144 186 564 667 1120 1748 1814 1863 0 49 115 743 1196 1299 1677 1719 1811 1863 0 92 104 176 339 1311 1325 1738 1845 1876 0 31 138 551 565 1537 1700 1772 1784 1876 0 176 200 338 577 1375 1401 1844 1871 1876 0 5 32 475 501 1299 1538 1676 1700 1876

En effet, l'application d'une méthode basée sur l'utilisation d'un paramètre aléatoire

aussi présentée dans [21] et [22] permet une réduction substantielle de α_j en plus d'une augmentation de la vitesse d'exécution de l'algorithme de recherche comparativement à la méthode algébrique. Le Tableau 3.1 donne quelques-uns des meilleurs résultats présentés dans [21] obtenus par la méthode pseudo-aléatoire.

3.3 Décodage itératif des codes CSO^{2}C-WS

L'algorithme de décodage itératif à seuil donné par (3.4) ne peut pas être appliqué directement sans avoir recours à certaines modifications. En effet, la condition $q \neq j$ ne peut être respectée sans introduire une propagation des erreurs. Cela s'explique en faisant appel au même raisonnement utilisé à la section 3.2 pour démontrer que les répétitions des observables ne peuvent être complètement éliminées du processus sans propagation des erreurs dans le reste du processus de décodage itératif. Cette condition $q \neq j$ doit donc être éliminée du processus de décodage.

Dans ce qui suit, on décrit deux structures de décodage, lesquelles dépendent de la condition $l \neq k$. La première structure de décodage ne fait pas intervenir cette condition $l \neq k$ alors que la deuxième approche consiste à appliquer directement l'algorithme de décodage tel que décrit par (3.4) mais sans la condition $q \neq j$.

3.3.1 Première structure de décodage (Structure 1) : sans la condition $l \neq k$

En se référant à (3.4), on remarque que les J symboles de parité, $y^p_{i+\alpha_j}$, sont utilisés

au plus J fois lorsque $l = k$. De plus, pour chaque permutation possible des indices k avec q et j avec l, les symboles d'information provenant du canal, $y^u_{i+\alpha_j-\alpha_k+\alpha_l-\alpha_q}$, sont utilisés au moins deux fois. Le processus de décodage itératif à seuil est décrit par l'équation suivante où contrairement à (3.4), les deux conditions $l \neq k$ et $q \neq j$ ne sont pas appliquées, c'est-à-dire :

$$\begin{aligned}
\lambda_i^{(2)} &= y_i^u + \sum_{j=1}^{J} \psi_{j,i}^{(2)} \quad = y_i^u + L_e^{(2)}(\hat{u}_i) \\
&= y_i^u + \sum_{j=1}^{J} \Bigg(y^p_{i+\alpha_j} \Diamond \sum_{k=j+1}^{J} \Diamond \lambda^{(2)}_{i+\alpha_j-\alpha_k} \quad \Diamond \\
&\qquad \sum_{k=1}^{j-1} \Diamond \Bigg(y^u_{i+\alpha_j-\alpha_k} + \sum_{l=1}^{J} \Bigg(y^p_{i+\alpha_j-\alpha_k+\alpha_l} \quad \Diamond \\
&\qquad\qquad \sum_{q=1}^{l-1} \Diamond y^u_{i+(\alpha_j-\alpha_k)-(\alpha_q-\alpha_l)} \Diamond \sum_{q=l+1}^{J} \Diamond \lambda^{(1)}_{i+(\alpha_j-\alpha_k)-(\alpha_q-\alpha_l)} \Bigg) \Bigg) \Bigg)
\end{aligned} \tag{3.5}$$

En considérant un code simplement orthogonal (CSOC), tous les symboles d'information provenant du canal, $y^u_{i+\alpha_j-\alpha_k+\alpha_l-\alpha_q}$, seraient répétés au moins $J(J-1)/2$ fois. Puisqu'il y a $(J(J-1)/2)+1$ symboles d'information, $y^u_{i+\alpha_j-\alpha_k}$, distincts et J symboles de parité distincts, seulement $(J(J-1)/2+J+1)$ termes seraient distincts à la deuxième itération. Par conséquent, même avec la Structure 1 où moins de la moitié des observables présents à la deuxième itération du processus itératif sont distincts,

les codes doublement orthogonaux produisent de meilleures performances d'erreur que les codes simplement orthogonaux. Cela s'explique facilement par le fait qu'avec des codes CSO^2C, la corrélation entre les observables à la deuxième itération en est réduite par rapport à la situation où des codes CSOC sont décodés itérativement. La Figure 3.2 montre une comparaison entre des résultats de simulation pour des codes J=6, CSOC et CSO^2C utilisant la Structure 1 de décodage. Ces résultats de simulation démontrent clairement l'avantage des codes CSO^2C sur les codes CSOC et ce, tout particulièrement après la deuxième itération. La Figure 3.2 indique que pour une probabilité d'erreur par bit de 2×10^{-4} obtenue à la deuxième itération, le code CSO^2C donne un gain de codage supplémentaire supérieur à 0.3 dB par rapport au code CSOC.

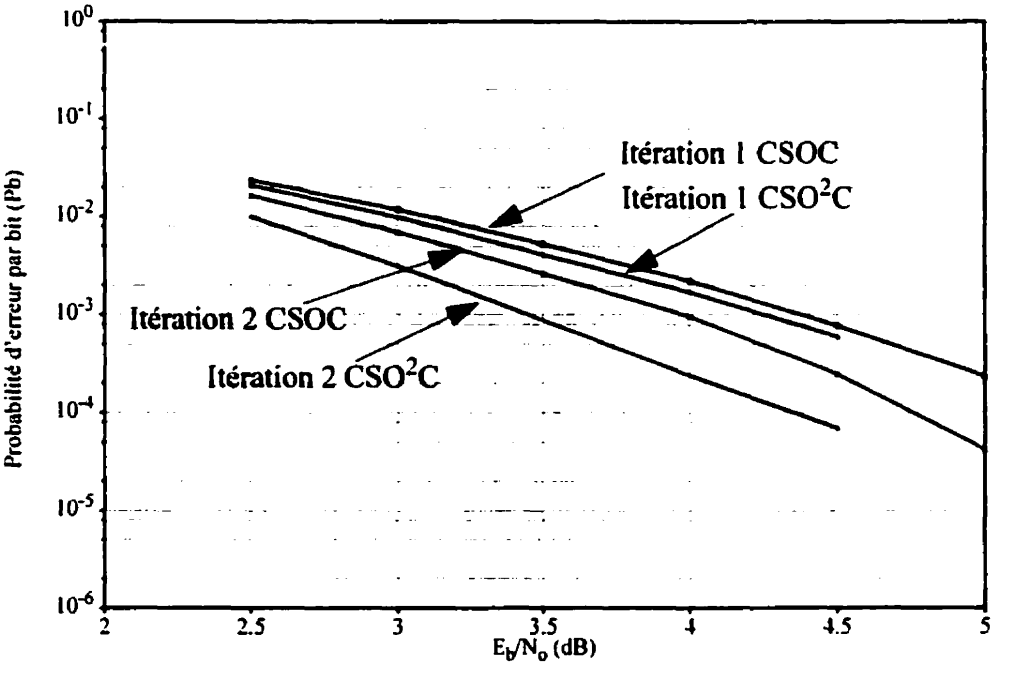


Figure 3.2 Comparaison des performances d'erreur entre des codes J=6, CSOC et CSO^2C pour les deux premières itérations

3.3.2 Deuxième structure de décodage (Structure 2) : avec la condition $l \neq k$

Avec cette deuxième structure de décodage, les répétitions indésirables des symboles de parité, $y^{p}_{i+\alpha_j}$, et d'information $y^{u}_{i+\alpha_j-\alpha_k}$ provenant du canal sont éliminées par la condition $l \neq k$ tandis que les symboles d'information provenant du canal, $y^{u}_{i+\alpha_j-\alpha_k+\alpha_l-\alpha_q}$, sont utilisés au moins deux fois. Cette seule condition implique qu'à chaque itération, J résultats intermédiaires doivent être produits et mémorisés avant d'être utilisés à l'itération suivante. Quoique le nombre nécessaire d'éléments mémoires dans le processus de décodage soit augmenté d'un facteur J, la latence demeure inchangée. Seule la complexité est affectée par un facteur J. Même si seulement $J-1$ équations $\psi^{(\mu)}_{j,i}$ sont utilisées à chaque itération, la convergence de l'algorithme de décodage est assurée car, plus de la moitié des équations $\psi^{(\mu)}_{j,i}$ sont alors utilisées à chaque itération. La Figure 3.3 illustre une comparaison des performances d'erreur obtenues par simulation en utilisant les Structures 1 et 2 de décodage pour un code J=6 CSO^{2}C. Quoique les résultats de simulation montrent une différence marginale entre les itérations de chaque structure, on remarque qu'à la deuxième itération, les performances d'erreur obtenues par la Structure 2 sont faiblement inférieures à celles obtenues en utilisant la Structure 1. Par contre, à la quatrième itération, la situation inverse se produit : les performances d'erreur obtenues par la Structure 1 sont faiblement inférieures à celles de la Structure 2. Cela signifie que la présence des répétitions indésirables provoque une convergence rapide mais, vers de

moins bonnes performances d'erreur que lorsque ces répétitions sont éliminées. Cependant, l'amélioration marginale des performances d'erreur apportée par la Structure 2 ne justifie pas l'accroissement de la complexité par un facteur de J. D'un point de vue pratique, une implantation matérielle devrait alors se faire à partir de la Structure 1. Pour cette raison, tous les résultats de simulation produits dans cette thèse sont obtenus en utilisant la Stucture 1.

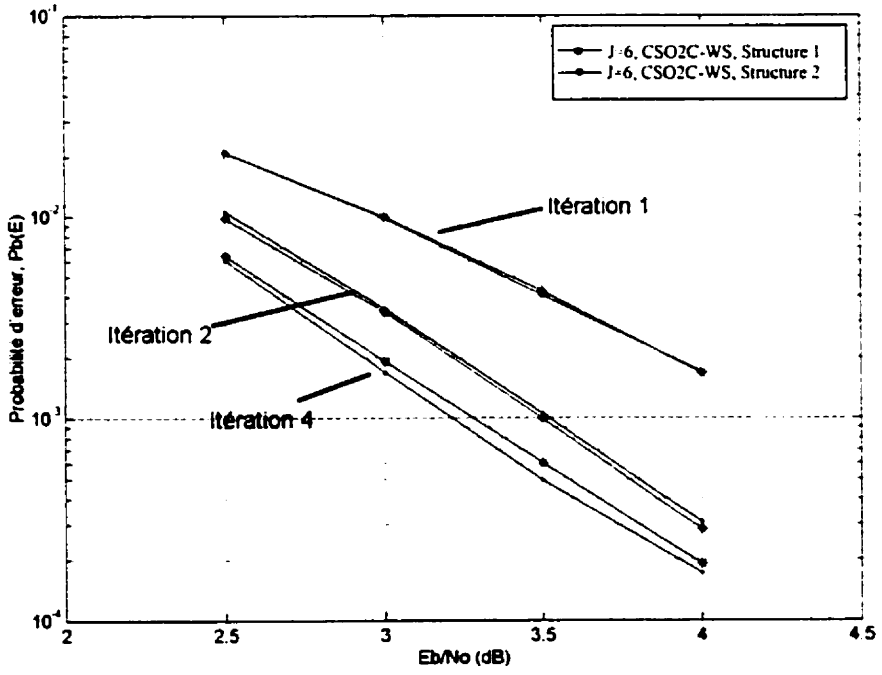


Figure 3.3 Performances d'erreur pour un code J=6 CSO^{2}C-WS utilisant les structures 1 et 2

3.4 Amélioration de l'algorithme de décodage à seuil itératif pour des CSO^{2}C-WS

Pour améliorer davantage la convergence du processus de décodage itératif vers de meilleures performances d'erreurs, une modification est apportée à l'algorithme de décodage itératif décrit par (3.1). La difficulté majeure de cet algorithme itératif de décodage est de maintenir les équations $\psi_{i,j}^{(\mu)}$ orthogonales dans tout le processus itératif. Évidemment, si les équations $\psi_{i,j}^{(\mu)}$ demeurent orthogonales dans tout le processus itératif de décodage, celui-ci devient une fonction de variables indépendantes et la corrélation entre les observables est nulle. Comme nous l'avons vu précédemment, avec l'utilisation des codes CSO^{2}C-WS, les relations (3.2), $\psi_{i,j}^{(\mu)}$, de la deuxième itération (μ=2) ne sont pas complètement orthogonales. Ceci empêche le processus de décodage de converger vers les meilleures performances d'erreur dictées par le pouvoir de correction du code. Une solution à ce problème consiste à pondérer par des coefficients $a_j^{(\mu)}$ les relations données par (3.2), $\psi_{i,j}^{(\mu)}$, lesquelles contiennent des termes corrélées. Il en résulte, à chaque itération, une amélioration de la fiabilité de la décision puisque l'importance d'une équation $\psi_{i,j}^{(\mu)}$ comportant des termes non-orthogonaux est réduite devant d'autres qui contiennent moins de termes non-orthogonaux. Cependant, comme nous le verrons plus loin, un plus grand nombre d'itérations est nécessaire pour atteindre la convergence du processus itératif. En introduisant les coefficients de pondération, $a_j^{(\mu)}, j = 0, 1, ..., J$, dans

l'algorithme décrit par (3.1), le LRV $\lambda_i^{(\mu)}$ obtenu à l'itération μ et à l'instant i devient :

$$\lambda_i^{(\mu)} = a_0^{(\mu)} y_i^u + \sum_{j=1}^{J} a_j^{(\mu)} \psi_{j,i}^{(\mu)} \tag{3.6}$$

Avec la première structure de décodage (Structure 1) décrite précédemment, sans l'ajout de ces coefficients, le nombre d'erreurs corrigées est généralement plus élevé à la première itération alors que les itérations suivantes apportent peu de corrections supplémentaires. Par exemple, avec un code CSO^{2}C-WS de taux de codage $r_c = 1/2$, $J = 6$, les résultats de simulation donnés à la Figure 3.4 montrent qu'après la troisième itération, il n'y a pas d'amélioration significative des performances d'erreur, c'est-à-dire que ce processus de décodage a convergé vers sa valeur maximale.

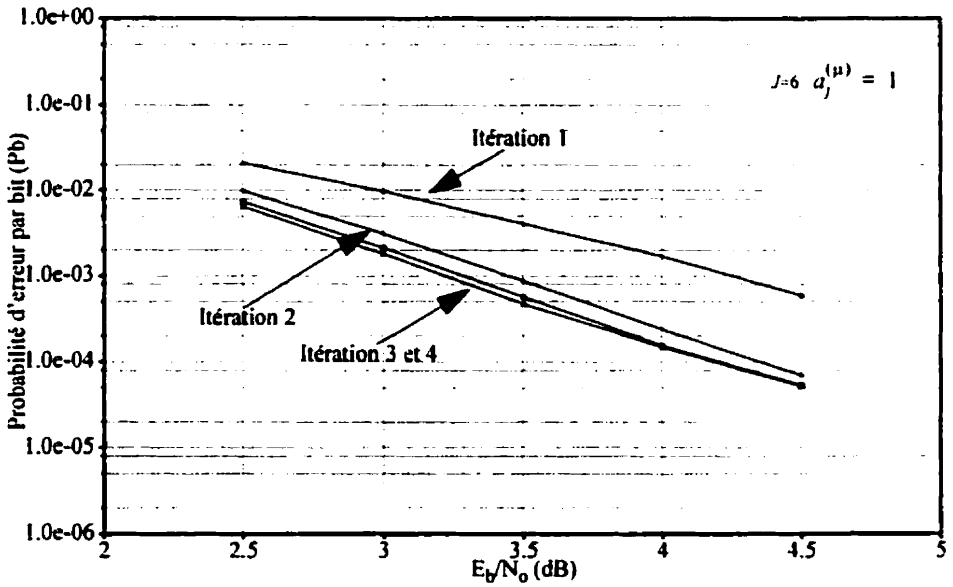


Figure 3.4 Résultats de simulation pour un code CSO^2-WS, J=6, $a_j^{(\mu)} = 1$, j=0,..., 6, μ=1,..., 4

En introduisant les coefficients $a_j^{(\mu)}$ à l'intérieur de chaque somme réelle de (3.5) et toujours en considérant la Structure 1 pour les deux premières itérations, l'algorithme de décodage s'écrit maintenant :

$$\begin{aligned}
\lambda_i^{(2)} &= a_0^{(2)} y_i^u + \sum_{j=1}^{J} a_j^{(2)} \psi_{i,j}^{(2)} \\
&= a_0^{(2)} y_i^u + \sum_{j=1}^{J} a_{(j)}^{(2)} \Bigg(y_{i+\alpha_j}^p \Diamond \sum_{k=j+1}^{J} \Diamond \lambda_{i+\alpha_j-\alpha_k}^{(2)} \quad \Diamond \\
&\qquad \sum_{k=1}^{j-1} \Diamond \Bigg(a_0^{(1)} y_{i+\alpha_j-\alpha_k}^u + \sum_{l=1}^{J} a_l^{(1)} \big(y_{i+\alpha_j-\alpha_k+\alpha_l}^p \quad \Diamond \\
&\qquad\qquad \sum_{q=1}^{l-1} \Diamond y_{i+\alpha_j-\alpha_k+\alpha_l-\alpha_q}^u \quad \Diamond \sum_{q=l+1}^{J} \Diamond \lambda_{i+\alpha_j-\alpha_k+\alpha_l-\alpha_q}^{(1)} \Bigg) \Bigg) \Bigg)
\end{aligned} \tag{3.7}$$

Le problème qui se pose alors consiste à déterminer la valeur de chaque coefficient, $a_j^{(\mu)}$, associé à chaque relation $\psi_{j,i}^{(\mu)}$ de sorte que la probabilité d'erreur à la dernière itération M soit minimale. Pour résoudre ce problème d'optimisation, il faut tout d'abord comprendre le rôle que jouent les coefficients dans le processus itératif de décodage à seuil. Pour des codes CSO^{2}C-WS, on note que la corrélation augmente rapidement après la deuxième itération. On pourrait alors conclure que les coefficients doivent être ajustés de manière à ne contrôler que le niveau de corrélation entre les variables observées à chaque itération. Cependant, il n'est pas suffisant de considérer la corrélation comme le facteur dominant dans l'amélioration de la fiabilité sur la décision finale car, minimiser la corrélation ne signifie pas pour autant que les relations $\psi_{j,i}^{(\mu)}$ ont pu être maintenues

orthogonales dans tout le processus de décodage. La fiabilité de chaque décision dépend aussi de la qualité du canal, c'est-à-dire du rapport signal à bruit E_b/N_o. Pour de très faibles valeurs de E_b/N_o, même en supposant un code parfaitement orthogonal sur M itérations, les relations $\psi_{j,i}^{(\mu)}$ peuvent être fortement perturbées par le grand nombre d'éléments bruités qui les composent de sorte que leur fiabilité en est diminuée.

3.4.1 Analyse en quantification ferme du processus de décodage itératif

Dans le but de fournir une explication adéquate de ce qui se produit sous les conditions exposées à la section précédente, il est préférable de faire appel à un modèle du processus de décodage en quantification ferme. En se référant à la relation (2.8) du chapitre 2 et en suivant le même principe du décodage itératif qui a conduit à (3.1), les J symboles de syndrome $s'^{(\mu)}_{i+\alpha_n}$, n=1, ..., J, de l'itération (μ) orthogonaux au symbole d'information u_i courant s'écrivent comme suit :

$$s'^{(\mu)}_{i+\alpha_n} = \tilde{u}_i \oplus \tilde{p}_{i+\alpha_n} \oplus \sum_{j=1}^{n-1} \oplus \hat{u}^{(\mu-1)}_{i+\alpha_n-\alpha_j}, \; n=1, ..., J \tag{3.8}$$

où dans (3.8) la rétroaction de la décision est considéré idéale, c'est-à-dire qu'aucune propagation d'erreur n'est possible par le biais de la rétroaction de la décision. D'autre part, nous supposerons dans tout le reste de cette analyse que le code est parfaitement orthogonal pour les M itérations du processus de décodage. Sous cette condition, le code est dit multiplement orthogonal au sens strict d'ordre M. À la section 3.5, nous

reviendrons sur cette notion de code au sens strict et au chapitre 4 le principe de multiplement orthogonal au sens strict d'ordre M sera nécessaire afin d'évaluer et d'analyser les performances d'erreur du décodage à seuil itératif à sortie non-quantifiée. L'équation (3.8) s'obtient facilement en modifiant les symboles de syndrome $s_{i+\alpha_n}^{(\mu)}$ normalement calculés à partir du symbole d'information $\hat{u}_i^{(\mu-1)}$ provenant de l'itération précédente :

$$s'^{(\mu)}_{i+\alpha_n} = s_{i+\alpha_n}^{(\mu)} \oplus \hat{u}_i^{(\mu-1)} \oplus \tilde{u}_i \tag{3.9}$$

La Figure 3.5 illustre la structure de décodage à seuil itératif en quantification ferme à l'itération (μ) correspondant à l'équation (3.8). Notons cependant que contrairement à l'équation (3.8), le schéma de la Figure 3.5 inclut la rétroaction de la décision. En utilisant la même procédure qu'au chapitre 2, (3.8) s'écrit aussi en fonction des symboles d'erreur :

$$s'^{(\mu)}_{i+\alpha_n} = e^{p}_{i+\alpha_n} \oplus e^{u}_{i} \oplus \bigoplus_{j=1}^{n-1} (e^{u}_{i+\alpha_n-\alpha_j} \oplus \hat{e}^{(\mu-1)}_{i+\alpha_n-\alpha_j}) \tag{3.10}$$

Un symbole de syndrome égal à la valeur binaire 1 indique que le symbole d'information reçu courant $\tilde{u}_i$ est possiblement erroné. Pour qu'il y ait, à chaque itération, amélioration des performances d'erreur, il faut que le nombre de symboles de syndrome de valeur binaire 1 puisse diminuer à chaque itération. Puisque chaque symbole de syndrome de l'itération (μ) dépend de la valeur des symboles de correction, $\hat{e}^{(\mu-1)}_{i+\alpha_n-\alpha_j}$, décidés à

l'itération précédente (μ-1), une erreur sur l'estimation du symbole $\hat{e}_{i+\alpha_n-\alpha_j}^{(\mu-1)}$ entraîne une propagation d'erreur aux itérations suivantes.

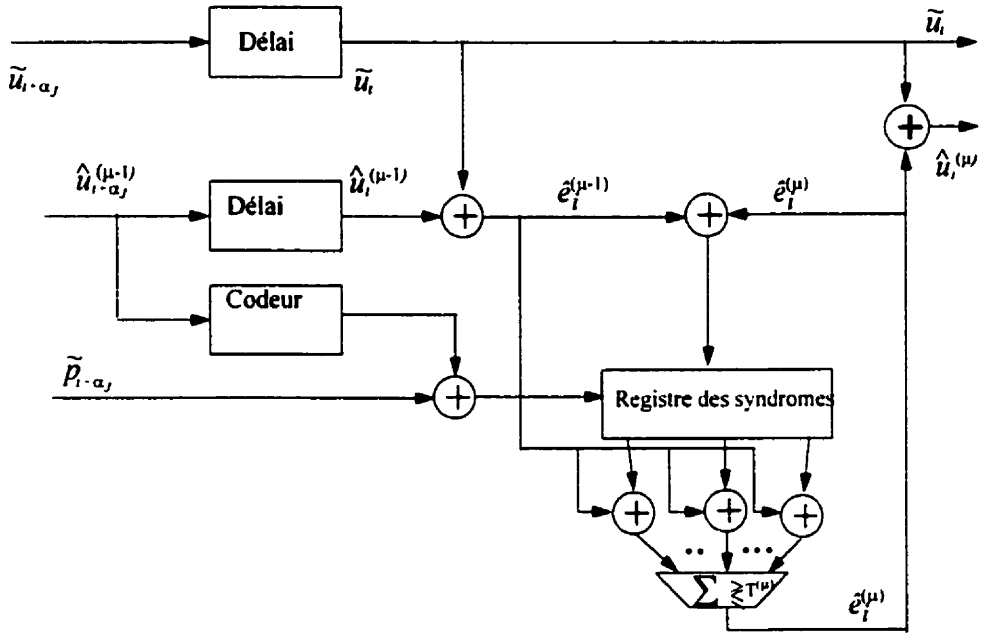


Figure 3.5 Structure en quantification ferme du processus de décodage à seuil itératif

Pour éviter ce problème de propagation, il faut s'assurer de n'introduire que des symboles décidés, $\hat{e}_{i+\alpha_n-\alpha_j}^{(\mu-1)} = 1$ ayant la plus grande fiabilité possible. Pour maintenir les décisions à leur plus grande fiabilité, il suffit de placer le seuil de décision à sa valeur la plus élevée. On montre dans [14], que la valeur optimale du seuil de décision qui minimise la probabilité d'erreur est $\lfloor J/2 \rfloor$. Cependant, cette valeur ne garantit pas que l'erreur est corrigée avec la plus grande fiabilité. Avec un seuil de décision égal à J-1 à la première itération, on impose que les J symboles de syndrome indiquent tous une erreur

sur le symbole d'information courant $\hat{u}_i$ pour décider que celui-ci est erroné. L'incertitude sur la décision est par conséquent réduite à sa valeur minimale et les valeurs des symboles d'erreur, $\hat{e}^{(\mu-1)}_{i+\alpha_n-\alpha_l}$, fournies à la seconde itération sont d'une grande fiabilité. À la deuxième itération, la valeur du seuil peut être réduite à (J-2) de manière à maintenir une faible incertitude sur les décisions mais, tout en corrigeant suffisamment d'erreurs pour permettre la convergence du processus de décodage. De la même manière, à chaque itération, le seuil de décision est réduit jusqu'à sa valeur optimale $\lfloor J/2 \rfloor$. La règle de décision utilisée à chaque itération s'écrit :

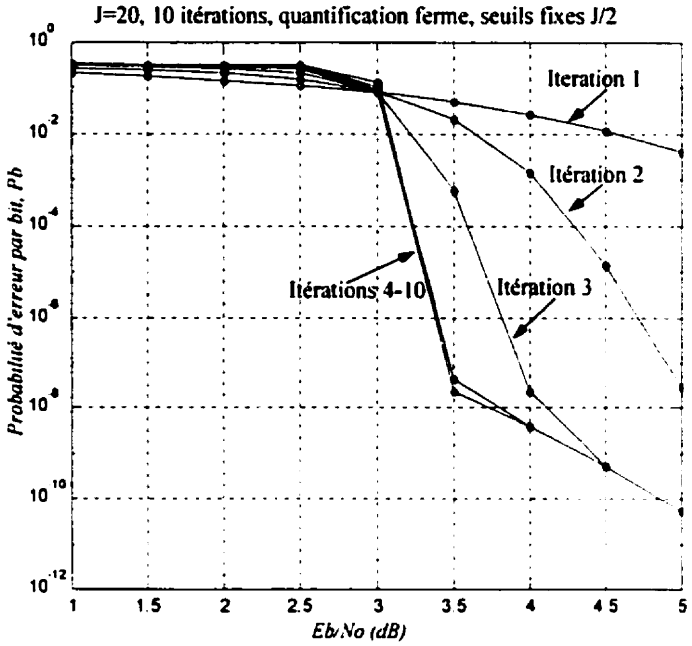
$$\textit{Choisir } \hat{e}_i^{(\mu)} = 1 \textit{ si et seulement si } \sum_{n=1}^{J} s'^{(\mu)}_{i+\alpha_n} > T^{(m)} \textit{ autrement } \hat{e}_i^{(\mu)} = 0 .$$

$$T^{(1)} = (J-1),\ T^{(2)} = (J-2),\ \ldots,\ T^{(k)} = \lfloor J/2 \rfloor,\ \ldots,\ T^{(M)} = \lfloor J/2 \rfloor .$$

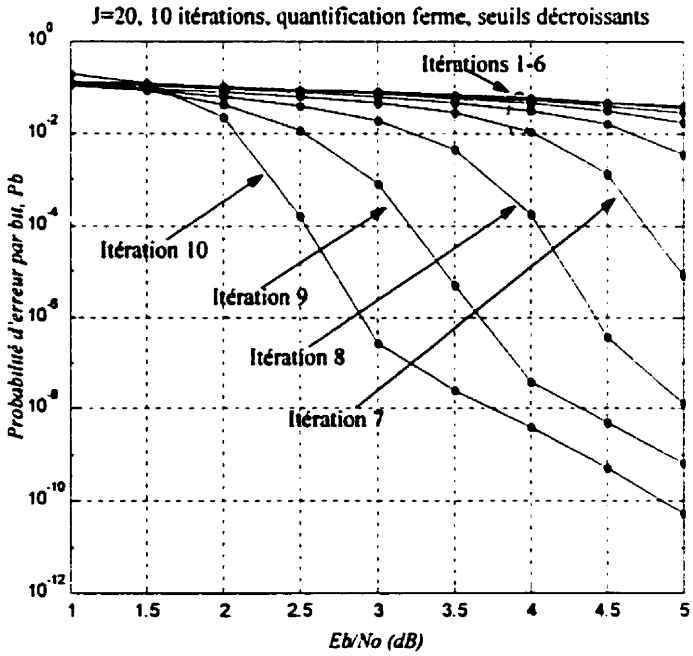
De cette façon, chaque symbole de syndrome devient un indicateur d'erreur fiable améliorant ainsi la fiabilité de la décision finale. Ce concept se vérifie théoriquement en calculant la probabilité d'erreur à chaque itération du processus de décodage en quantification ferme et en supposant un code multiplement orthogonal d'ordre M au sens strict. La probabilité d'erreur $P_b^{(\mu)}$ de l'itération (μ) s'obtient par :

$$P_b^{(\mu)} = P\left\{ \sum_{n=1}^{J} s'^{(\mu)}_{i+\alpha_n} > T^{(\mu)} \middle| e_i^u = 0 \right\}(1-\gamma_t) + P\left\{ \sum_{n=1}^{J} s'^{(\mu)}_{i+\alpha_n} \le T^{(\mu)} \middle| e_i^u = 1 \right\}\gamma_t \quad (3.11)$$

où l'on rappelle que la probabilité de transition du canal $\gamma_t = P\left\{e_i^u = 1\right\} = Q(\sqrt{E_b/N_o})$. Les détails des calculs permettant d'évaluer (3.11) sont fournis à l'Annexe VI. La Figure 3.6 montre, pour un code J=20 multiplement orthogonal d'ordre M=10 au sens strict, comment à chaque itération (μ) la probabilité d'erreur $P_b^{(\mu)}$ évolue en fonction de E_b/N_o lorsque le seuil de décision de chaque itération est égale à la valeur optimale $\lfloor J/2 \rfloor$ (Figure 3.6a) et lorsqu'un ensemble de seuils de décision décroissants est utilisé (Figure 3.6b).



(a)



(b)

Figure 3.6 (a) Seuils fixes $T^{(k)} = \lfloor J/2 \rfloor$, k=1, 2, ..., 10, (b) Seuils décroissants

La comparaison des dernières itérations de chaque système montre, pour de faibles valeurs de E_b/N_o, une amélioration importante des performances d'erreur lorsqu'un ensemble de seuils de décision décroissants est utilisé même si le code est parfaitement orthogonal sur les M=10 itérations de décodage. Il est évident que pour un code CSO^{2}C-WS, cette amélioration est limitée par la présence d'une corrélation entre les observables. Les coefficients de pondération dans le processus de décodage à sortie non-quantifiée servent à maintenir l'orthogonalité des relations $\psi_{j,i}^{(\mu)}$ dans tout le processus de décodage et tout comme le fait l'ensemble de seuils décroissants dans le cas d'une quantification ferme, ils servent aussi à conserver la fiabilité des observables à leur valeur maximale tout en assurant la convergence du processus de décodage itératif.

Toutefois, une solution analytique au problème de maximisation de la fiabilité des observables dans le processus de décodage à seuil itératif à sortie non-quantifiée semble impraticable puisqu'elle demanderait une étude de l'interdépendance des observables dans tout le processus de décodage itératif. Dans cette thèse, deux autres approches de solutions sont examinées. La première méthode est une technique dite expérimentale et la seconde fait appel à des concepts s'inspirant des réseaux de neurones.

3.4.2 Méthode expérimentale

Cette méthode consiste à varier les coefficients $a_j^{(\mu)}$ pour une valeur spécifique de E_b/N_o et ainsi déterminer, par simulation, les valeurs optimales $a_j^{(\mu)*}$ qui minimisent le

taux d'erreur binaire, TEB. Une solution simple est d'assigner à tous les coefficients, $a_j^{(\mu)}$, $j = 0, 1, ..., J$ et $\mu = 1, 2, ..., M$, une même valeur a, c'est-à-dire :

$$a_j^{(\mu)} = a,\ j = 0, 1, ..., J \text{ et } \mu = 1, 2, ..., M \tag{3.12}$$

Par conséquent, (3.6) s'écrit de la façon suivante :

$$\lambda_i^{(\mu)} = a_0^{(\mu)} y_i^u + \sum_{j=1}^{J} a_j^{(\mu)} \psi_{j,i}^{(\mu)} = a\left(y_i^u + \sum_{j=1}^{J} \psi_{j,i}^{(\mu)} \right) \tag{3.13}$$

Pour un rapport signal à bruit E_b/N_o fixe, des résultats de simulation sont obtenus en faisant varier la valeur du coefficient de pondération a. Cette expérimentation a été menée pour plusieurs codes CSO^{2}C-WS de taux de codage $r_c = 1/2$ indiqués au Tableau 3.1. Par exemple, pour le code CSO^{2}C-WS, J=8, $r_c = 1/2$ spécifié par l'ensemble des valeurs entières $\{\alpha_j\}$={0, 43, 139, 322, 422, 430, 441, 459}, les Figures 3.7 et 3.8 indiquent très clairement qu'il existe une valeur de a^* qui minimise le TEB pour une valeur spécifique de E_b/N_o où 8 itérations sont utilisées dans les deux cas. À partir des résultats montrés aux Figures 3.7 et 3.8, on observe qu'une valeur de $a = 0.3$ est nécessaire pour atteindre un TEB minimal à un rapport signal à bruit $E_b/N_o = 3$ dB mais, qu'une valeur $a = 0.2$ permet un TEB minimal pour un $E_b/N_o = 4$ dB.

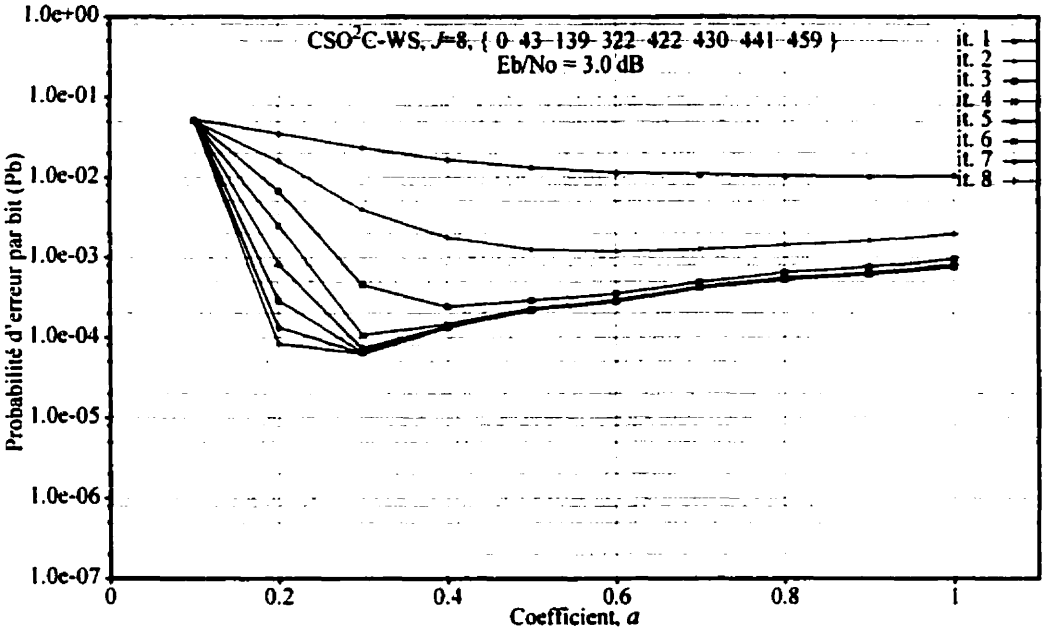


Figure 3.7 Variation du coefficient a pour un code CSO^{2}C-WS, $J = 8$ à $E_b/N_o = 3$ dB

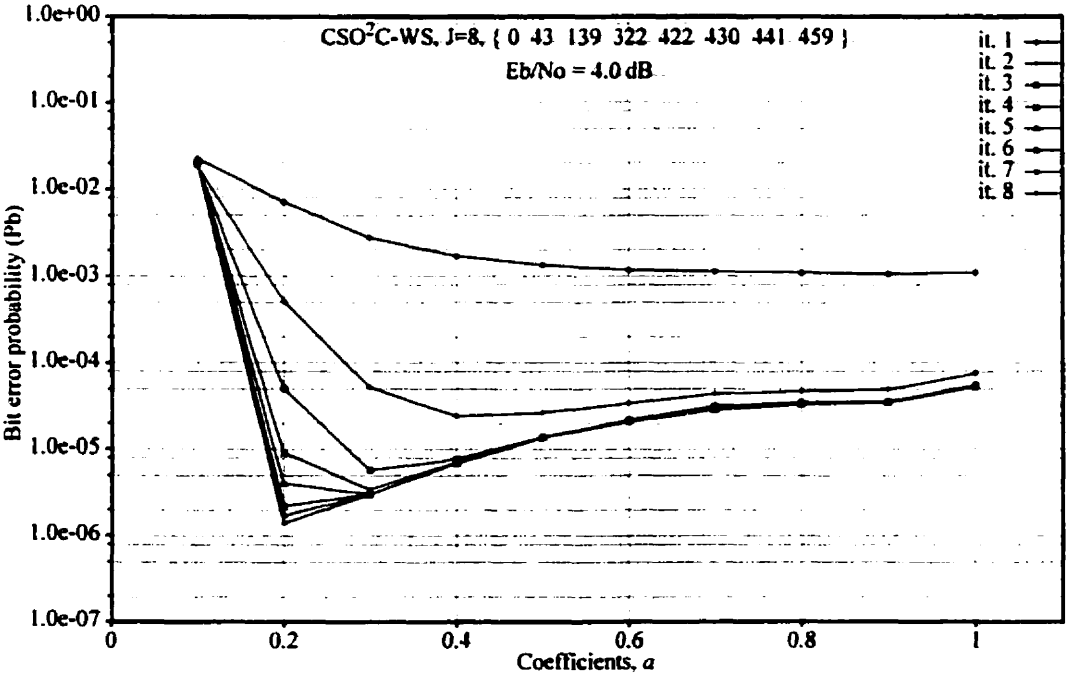


Figure 3.8 Variation du coefficient a pour un code CSO^{2}C-WS, $J = 8$ à $E_b/N_o = 4$ dB

Ces résultats de simulation indiquent aussi une sensibilité faible du TEB minimal en fonction de la variation du coefficient a sur une plage de variation de E_b/N_o relativement grande. Par conséquent, le coefficient de pondération a varie très peu en fonction du rapport signal à bruit E_b/N_o.

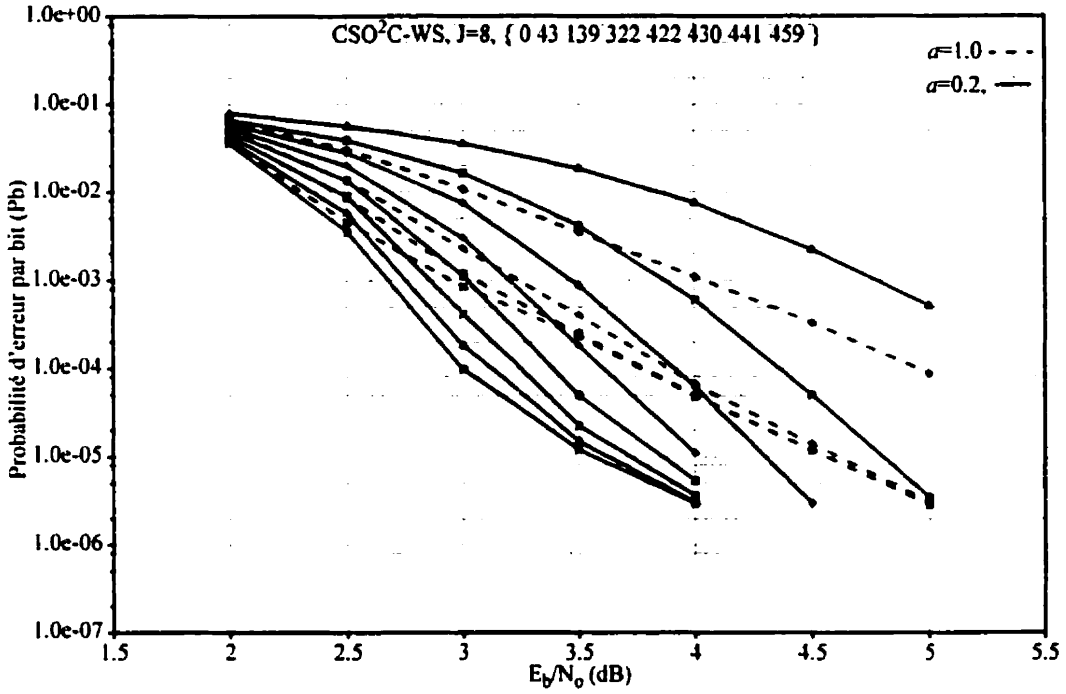


Figure 3.9 Comparaison entre le processus itératif sans l'utilisation des coefficients et celui utilisant le coefficient optimal $a^* = 0.2$

Toujours pour le même code CSO²C-WS, J=8, des résultats de simulation sont obtenus pour $a^* = 0.2$ et $a = 1.0$ (sans coefficient) où, encore une fois, 8 itérations sont utilisées. Ces résultats de simulation sont comparés entre eux et sont reportés à la Figure 3.9 où l'on observe que pour un TEB de 10^{-4}, le processus itératif utilisant un coefficient optimal $a^* = 0.2$ procure un gain d'environ 0.75 dB comparé à un processus itératif sans

coefficient (a =1.0) et ce pour un rapport E_b/N_o de 3.0 dB. De plus, avec un coefficient optimal a^* = 0.2, un plus grand nombre d'itérations est nécessaire pour atteindre la convergence comparativement à un processus sans ces coefficients. Cet accroissement du nombre d'itérations engendre aussi une augmentation de la latence totale du processus de décodage. Des coefficients de pondération de la plupart des codes CSO^{2}C-WS donnés au Tableau 3.1 ont été obtenus en utilisant cette méthode expérimentale et les résultats de simulation utilisant ces coefficients de pondération sont fournis à l'Annexe III.

3.4.3 Approche par réseaux de neurones

Dans la section précédente, le problème de déterminer les coefficients optimaux $a_j^{(\mu)*}$ pour le cas simple où tous étaient égaux à une valeur a^* spécifique a été considéré. Le problème qui nous concerne maintenant est de déterminer un ensemble de coefficients optimaux où ceux-ci peuvent être différents. Il est clair qu'utiliser la technique précédente devient très coûteuse en temps de calcul dû au très grand nombre de valeurs possibles que peut prendre simultanément chacun des coefficients $a_j^{(\mu)}$. Ainsi, dans le but de réduire le temps de calcul et d'exécution du processus d'optimisation, une méthode plus efficace est nécessaire. Toutefois, on observe que le processus itératif de décodage à seuil décrit par (3.6) est similaire à un réseau de neurones. Généralement, chaque neurone appartenant à un réseau de neurones est décrit par la relation suivante :

$$X_j^{(\mu)} = f\left(\sum_k a_{kj}^{(\mu)} X_k^{(\mu-1)}\right) \tag{3.14}$$

où $X_j^{(\mu)}$ est le $j^{ième}$ neurone de la couche μ et où $f(.)$ est une fonction non-linéaire d'une seule variable [40]. La relation (3.14) est une fonction non-linéaire d'une combinaison linéaire des variables $X_k^{(\mu-1)}$ alors que (3.6) est une combinaison linéaire des fonctions non-linéaires $\psi_{j,i}^{(\mu)}$. En examinant attentivement la relation (3.6), on constate rapidement que chaque fonction $\psi_{j,i}^{(\mu)}$ représente le $j^{ième}$ neurone d'une couche μ d'un réseau. Pour déterminer les meilleurs coefficients de pondération, il est alors possible d'utiliser une des nombreuses techniques d'apprentissage disponibles dans la littérature portant sur les réseaux de neurones [40]. Dans cette étude, l'algorithme de rétropropagation du gradient est choisi pour sa simplicité même si sa vitesse de convergence est assez lente et nécessite un petit pas d'adaptation pour éviter l'instabilité [41]. Notons que ces coefficients de pondérations peuvent être optimisés avant une implantation matérielle du décodeur itératif si le canal de transmission est invariant dans le temps ou être ajustés dynamiquement pour un signal reçu en présence d'évanouissements.

Comme nous l'avons déjà mentionné, le critère d'optimisation est le TEB minimal pour un rapport signal à bruit E_b/N_o donné. Cependant, puisque le TEB désiré est typiquement petit, de l'ordre 10^{-6} ou même 10^{-7}, l'ajustement des coefficients pourrait se faire très rarement. Si l'on accepte l'hypothèse que la sortie du décodeur, $\lambda_i^{(M)}$, à la dernière itération M, suit approximativement une fonction de distribution gaussienne,

réduire le TEB ou déterminer le maximum de vraisemblance est équivalent à minimiser l'erreur quadratique moyenne, EQM, de la sortie. Sachant que les symboles transmis sont dénotés par x_i^u, l'EQM, dénoté par $E^{(M)}$ obtenu à la sortie du décodeur de la dernière itération M est :

$$E^{(M)} = \overline{\left\{\lambda_i^{(M)} - x_i^u\right\}^2} \tag{3.15}$$

où $\overline{\{.\}}$ signifie une moyenne temporelle. L'algorithme de rétropropagation de l'erreur souvent utilisé dans les réseaux de neurones est adapté à ce problème d'optimisation sans contrainte. Quoique cette procédure d'optimisation ne garantit pas que les résultats obtenus représentent les vraies valeurs optimales des coefficients de pondération $a_j^{(\mu)*}$, elle fournit néanmoins une solution quasi-optimale.

3.4.4 Algorithme de rétropropagation de l'erreur adapté au décodage itératif à seuil

Dans cette section, on présente l'algorithme d'optimisation utilisé pour minimiser sans contrainte l'erreur quadratique moyenne, EQM, à la sortie du décodeur à seuil à la dernière itération M. Notons que dans cette procédure d'optimisation, les termes de la rétroaction de la décision présents dans (3.2) ne sont pas considérés, c'est-à-dire que l'on suppose qu'ils n'introduisent aucune erreur dans le processus de décodage. Évidemment, cette supposition n'est pas toujours réaliste, tout particulièrement à de faibles rapports signal à bruit, E_b/N_o, où les erreurs de transmission sont les plus fréquentes.

Cette procédure d'optimisation est basée sur l'algorithme de rétropropagation de l'erreur souvent utilisé dans les réseaux de neurones [40]. Les détails du développement menant à cet algorithme sont donnés à l'Annexe VII. Pour une séquence d'information de longueur L, l'EQM donné par (3.15) devient :

$$E^{(M)} = \overline{\left\{\lambda_i^{(M)} - x_i^u\right\}^2} \approx \frac{1}{L}\sum_{i=0}^{L-1}(\lambda_i^{(M)} - x_i^u)^2 \tag{3.16}$$

On définit aussi l'erreur instantanée $\delta_i^{(M)}$ à l'itération M et à l'instant i de transmission comme suit :

$$\delta_i^{(M)} = \lambda_i^{(M)} - x_i^u \tag{3.17}$$

En utilisant l'algorithme de rétropropagation de l'erreur, on détermine les valeurs optimales des coefficients de pondération $a_j^{(\mu)*}$ telles que :

$$\frac{\partial E^{(M)}}{\partial a_j^{(\mu)}} \to 0, \forall \mu \text{ et } \forall j \tag{3.18}$$

Puisque l'algorithme de rétropropagation de l'erreur est une procédure itérative, le résultat désiré est obtenu après plusieurs itérations ou lorsque la convergence de l'algorithme est atteint. À l'Annexe VII, on montre que, $\partial E^{(M)}/\partial a_j^{(\mu)}$, pour $j = 1, 2, ..., J$, s'obtiennent par les relations suivantes :

$$\frac{\partial E^{(M)}}{\partial a_j^{(\mu)}} = \frac{2}{L}\sum_{i=0}^{L-1}\sum_{l=0}^{(M-\mu)\alpha_J} \delta_{i+l}^{(\mu)}\psi_{j,i+l}^{(\mu)}, \mu = 1, 2, ..., M \tag{3.19}$$

alors que pour $j = 0$, $\partial E^{(M)} / \partial a_0^{(\mu)}$, on trouve les relations suivantes :

$$\frac{\partial E^{(M)}}{\partial a_0^{(\mu)}} = \frac{2}{L} \sum_{i=0}^{L-1} \sum_{l=0}^{(M-\mu)\alpha_J} \delta_{i+l}^{(\mu)} y_{i+l}^{u} \quad , \mu = 1, 2, ..., M \tag{3.20}$$

où les termes $\psi_{j,i}^{(\mu)}$ dans (3.19), sont donnés par (3.2). Les valeurs de $\delta_{i+l}^{(\mu)}$, $l = 0, 1, 2, ...$ $(M-\mu)\alpha_J$, $\mu = 1, 2, ..., M$-1, sont les erreurs instantanées propagées à l'itération μ et elle sont obtenues par :

$$\delta_{i+l}^{(\mu)} = \sum_{k=0}^{(M-\mu-1)\alpha_J} \delta_{i+k}^{(\mu+1)} \frac{\partial \lambda_{i+k}^{(\mu+1)}}{\partial \lambda_{i+l}^{(\mu)}} \quad , \tag{3.21}$$

$$l = 0, 1, 2, ..., (M-\mu)\alpha_J, \mu = 1, 2, ..., M\text{-}1$$

où la valeur de $\delta_i^{(M)}$ est donnée par (3.17). Le résultat de la dérivée $\frac{\partial \lambda_{i+k}^{(\mu+1)}}{\partial \lambda_{i+l}^{(\mu)}}$ est aussi donné à l'Annexe VII. La mise à jour des coefficients de pondération $a_j^{(\mu)}$, $j = 0, 1, 2, ...$ J et $\mu = 1, 2, ..., M$, se fait à chaque itération de l'algorithme de rétropropagation de l'erreur selon la règle delta [40], [41]:

$$a_j^{(\mu)} \leftarrow a_j^{(\mu)} + \Delta a_j^{(\mu)} \tag{3.22}$$

où $\Delta a_j^{(\mu)}$, $\mu = 1, 2, ..., M$, représente l'incrément nécessaire pour la mise à jour des $a_j^{(\mu)}$ et s'obtient de la façon suivante :

$$\Delta a_j^{(\mu)} = \begin{cases} -\eta \frac{2}{L} \sum_{i=0}^{L-1} \sum_{l=0}^{(M-\mu)\alpha_J} \delta_{i+l}^{(\mu)} \psi_{j,i+l}^{(\mu)}, & j = 1, 2, \ldots, J \\ -\eta \frac{2}{L} \sum_{i=0}^{L-1} \sum_{l=0}^{(M-\mu)\alpha_J} \delta_{i+l}^{(\mu)} y_{i+l}^{u}, & j = 0 \end{cases} \tag{3.23}$$

Dans (3.23), $0 < \eta \leq 1$ est le pas d'adaptation de l'algorithme de rétropropagation de l'erreur. Les relations (3.22) et (3.23) sont appliquées de manière itérative jusqu'à ce que l'algorithme atteigne la convergence pour une séquence codée transmise auquelle s'ajoute une séquence de bruit. Dans cette étude, le choix du critère de convergence est le suivant : $\left| E_{t_r}^{(M)} - E_{t_r-1}^{(M)} \right| \leq \varepsilon$ ou lorsque le nombre maximal d'itérations est atteint. Dans ce critère de convergence, $E_{t_r}^{(M)}$, représente le EQM de l'étape de décodage M obtenu à l'itération t_r de l'algorithme de rétropropagation de l'erreur. La valeur ε doit être fixée la plus petite possible pour s'assurer de la convergence de l'algorithme d'optimisation et a été évaluée expérimentalement à environ 10^{-6}. Toutefois, le nombre maximal d'itérations de l'algorithme de rétropropagation de l'erreur doit être suffisamment grand pour assurer un résultat le plus fiable possible. Pour chaque cas traité dans cette thèse, ce nombre maximal d'itérations a été déterminé de manière expérimentale. Les meilleurs résultats obtenus ont montré qu'un nombre maximal de 5 itérations était suffisant. Pour améliorer davantage les performances, l'algorithme de rétropropagation de l'erreur est répété en utilisant à chaque fois, les coefficients de pondération obtenus à l'étape précédente et une nouvelle séquence codée transmise auquelle s'ajoute le bruit du canal. Toujours de manière expérimentale, le

nombre de répétitions de l'algorithme de rétropropagation de l'erreur a été déterminé pour donner les meilleurs résultats compte tenu du temps d'exécution du processus d'optimisation. En moyenne ce nombre de répétitions est d'environ 20.

Puisque cet algorithme converge lentement, la valeur du pas d'adaptation, η, est choisie comme étant la plus petite possible mais, tout en tenant compte d'un temps de calcul acceptable. Ce pas d'adaptation η a été aussi déterminé expérimentalement pour chaque cas traité et de façon générale, sa valeur se situe dans l'ordre de grandeur de 0.1. D'autre part, la longueur L de la séquence d'information peut aussi influencer les résultats de l'optimisation. Pour réduire l'influence de L sur les résultats, il suffit de choisir L le plus grand possible. Pour un temps de calcul acceptable et des résultats fiables, la longueur L de la séquence d'information a été fixée à $L = 10^6$ bits d'information. Avec cette longueur de séquence, on peut espérer atteindre des TEB d'environ 10^{-4} à 10^{-5} avec suffisamment de précision.

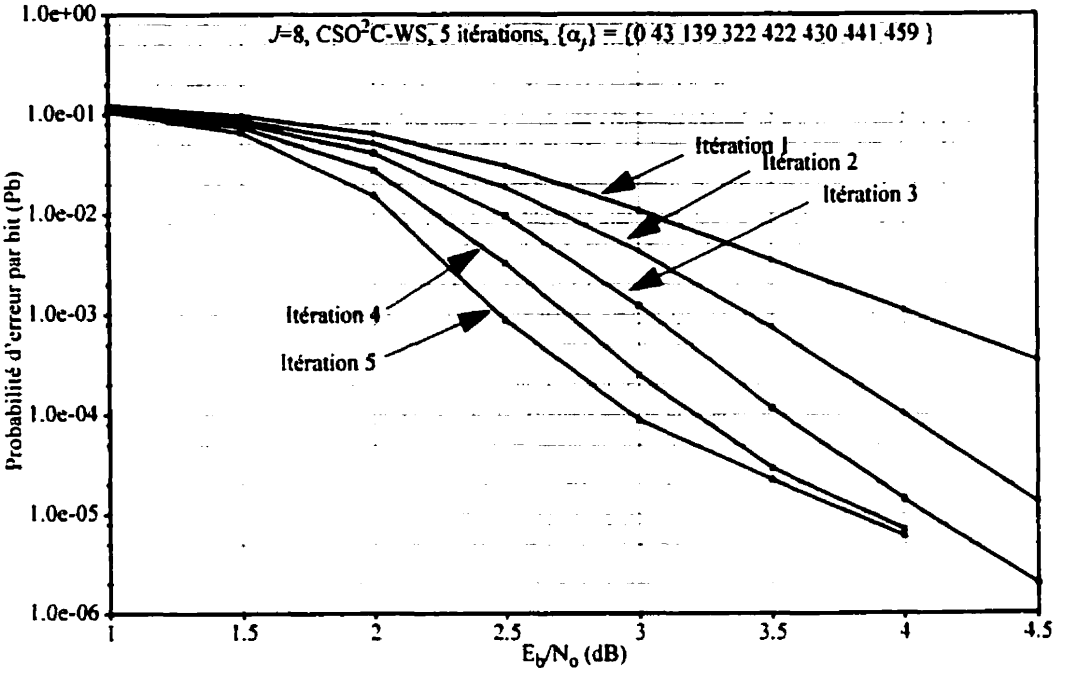


Figure 3.10 Résultats de simulation pour un code CSO^2-WS, $J = 8$ utilisant des coefficients de pondération obtenus par l'algorithme de rétropropagation de l'erreur

La Figure 3.10 montre des résultats de simulation d'un code J=8, CSO^{2}C-WS utilisant l'algorithme de rétropropagation de l'erreur pour déterminer les coefficients de pondération sous-optimaux pour un point d'opération de E_b/N_o = 3.0 dB. Pour ce code, la procédure d'optimisation a été appliquée pour chaque itération montrée à la Figure 3.10. Les valeurs des coefficients de pondération obtenues par cette procédure sont données dans le Tableau 3.2 pour 5 itérations. Les coefficients des deux dernières itérations peuvent être fixés à la valeur 1.0 sans introduire une réduction des performances d'erreur puisque, par hypothèse, à ces deux dernières itérations, la double orthogonalité est pratiquement atteinte.

Tableau 3.2 : Coefficients de pondération pour le CSO^2-WS, $J = 8$ de la Figure 3.10

	Équations								
itérations	0	1	2	3	4	5	6	7	8
1	0.398789	0.277387	0.247278	0.293361	0.055445	0.214385	0.216243	0.217340	0.218382
2	0.140025	0.274598	0.244374	0.228919	0.266262	0.276888	0.277108	0.278195	0.0.279358
3	0.122809	0.259570	0.226189	0.208575	0.245274	0.256188	0.256905	0.258776	0.260965
4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

3.4.5 Sensibilité des performances d'erreur aux imprécisions des coefficients de pondération

Avec la méthode expérimentale, il a été déterminé que la sensibilité des performances d'erreur aux variations du coefficient de pondération a^* était faible sur un grande plage de variation du rapport signal à bruit E_b/N_o. Il peut être aussi intéressant de vérifier cette sensibilité lorsque les coefficients sont ajustés en utilisant l'algorithme de rétropropagation de l'erreur. La Figure 3.11 montre un résultat de simulation pour un code CSO^{2}C-WS, J=6 dont les coefficients de pondération sont ajustés selon l'algorithme de rétropropagation de l'erreur sur quatre itérations de décodage. Sur cette même figure, sont superposés d'autres résultats de simulation du même code où chaque coefficient de pondération est affecté par une perturbation aléatoire de distribution gaussienne de moyenne nulle et de variance $s^2 = 0.01$. On définit un coefficient de variation dénoté par CV comme étant le rapport s/u de l'écart-type s de la perturbation avec la moyenne u des coefficients de pondération dans tout le processus de décodage itératif. Par exemple, pour le code de la Figure 3.11, CV = s/u = 0.1/0.35456 = 28.2 %.

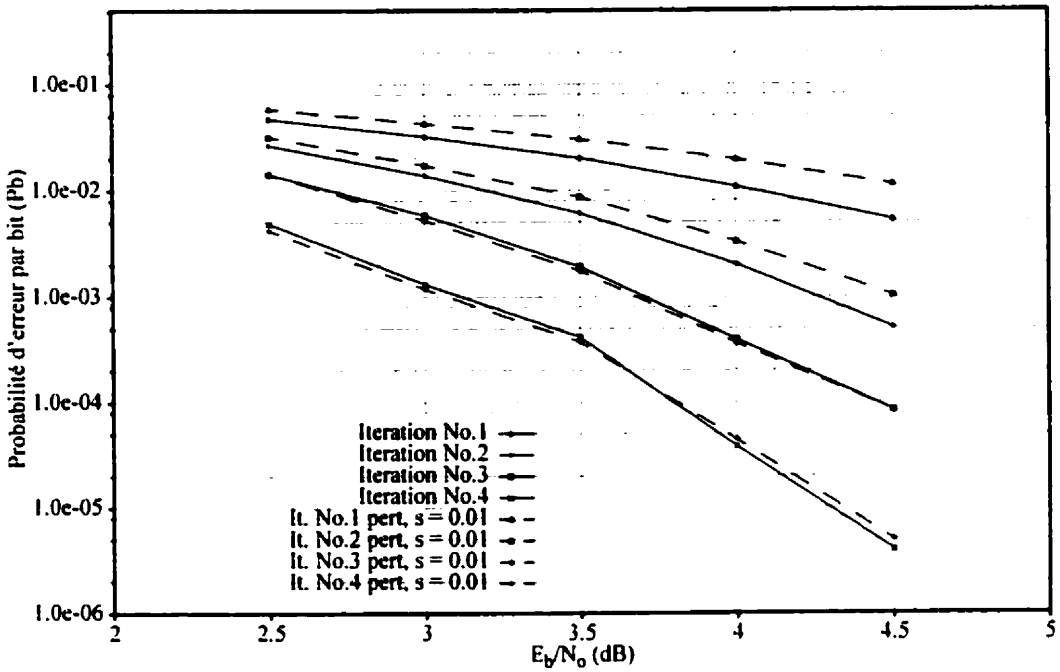


Figure 3.11 Sensibilité des performances d'erreur aux variations des coefficients de pondération pour un CSO^{2}C-WS, J=6, CV=28 %

On note qu'une perturbation légère des coefficients de pondération entraîne une dégradation des performances plus importante pour les premières itérations. Par exemple, la première itération de la Figure 3.11 subit une dégradation des performances d'erreur supérieure à 0.5 dB. Cette dégradation devient moins importante pour les itérations ultérieures. On remarque qu'à la dernière itération, seulement une très faible dégradation des performances d'erreur d'environ 0.1 dB se produit. Évidemment, cette dégradation est plus importante lorsque la dispersion de la perturbation autour de la moyenne u augmente. Toujours pour le même code CSO^{2}C-WS, J=6, les Figures 3.12 et 3.13 illustrent bien ce qui se passe lorsque le coefficient de variation CV augmente. Les plus fortes dégradations

des performances d'erreur surviennent avec un CV=70.5%. Cependant, même avec un CV de près de 40 %, une dégradation de seulement de 0.3 dB se produit. Cela démontre que ce système de décodage itératif est peu sensible aux imprécisions des coefficients de pondération.

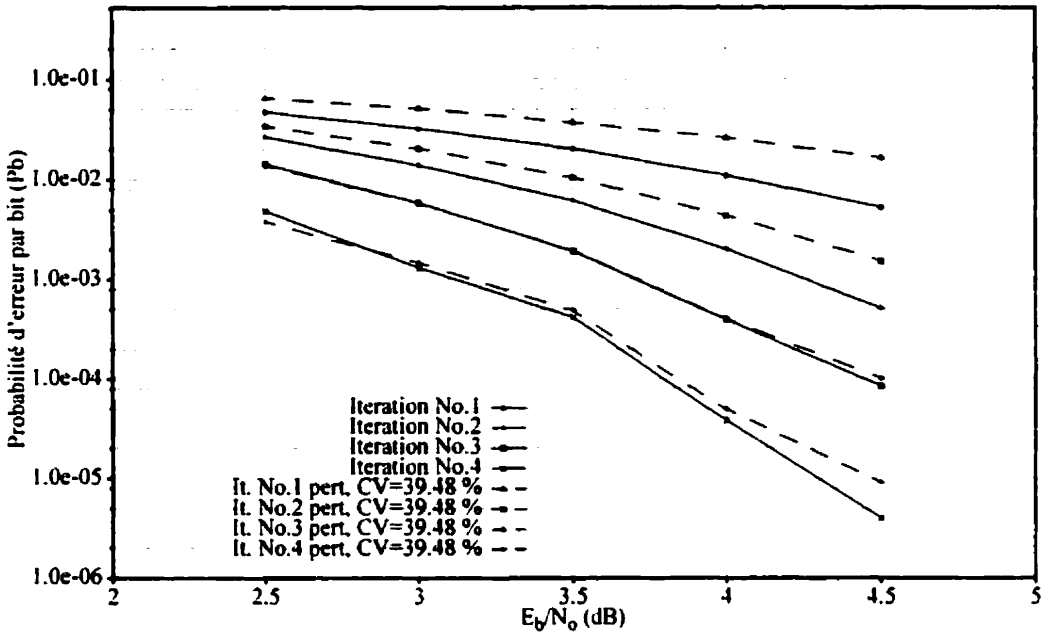


Figure 3.12 Sensibilité des performances d'erreur aux variations des coefficients de pondération pour un CSO^{2}C-WS, J=6, CV=39.48 %

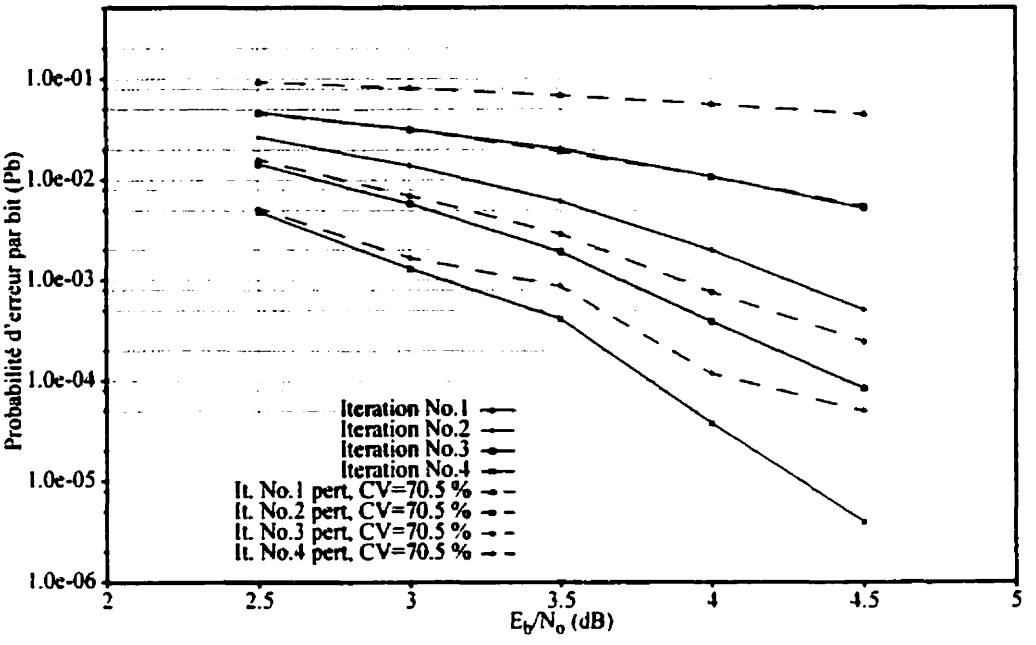


Figure 3.13 Sensibilité des performances d'erreur aux variations des coefficients de pondération pour un CSO^{2}C-WS. J=6, CV=70.5 %

3.4.6 Performance de l'algorithme de rétropropagation de l'erreur

Une comparaison des résultats de simulation montrés à la Figure 3.10 avec ceux de la Figure 3.9 obtenus pour le même code mais, utilisant des coefficients de pondération égaux, montre que pour un TEB d'environ 10^{-4}, seulement 5 itérations sont nécessaires lorsque l'algorithme de rétropropagation de l'erreur est appliqué alors qu'il en faut 8 pour atteindre le même TEB lorsque $a_j^{(\mu)} = a^* = 0.2$, j=0, 1, 2, ...,J, μ= 1, 2, ..., M. D'autre part, on remarque que pour de plus forts rapports signal à bruit et avec un décodage itératif où les coefficients sont tous égaux, les performances d'erreur sont meilleures d'environ 0.2 à 0.3 dB comparativement à un décodage itératif utilisant des coefficients ajustés selon

la méthode basée sur les réseaux de neurones. Ce comportement s'explique par le fait que les coefficients de pondération obtenus par la méthode basée sur les réseaux de neurones sont, dans ce cas-ci, optimaux pour la seule valeur E_b/N_o = 3.0 dB alors que dans l'autre cas, la valeur a^*=0.2 est optimale pour au moins $3.0 < E_b/N_o \leq 4.0$ dB où 8 itérations sont utilisées.

D'autres résultats de simulation utilisant des coefficients de pondération ajustés par l'algorithme de rétropropagation de l'erreur sont présentés à l'Annexe IV. Malheureusement, il fut très difficile d'obtenir des résultats pour des codes avec $J \geq 9$. En effet, le temps d'exécution et la complexité de mise en oeuvre de cette méthode augmentent rapidement avec J. À titre d'exemple, le Tableau 3.3 montre les temps d'exécution pour des codes CSO^{2}C-WS, J=6 à 8.

Tableau 3.3 : Durée d'exécutions de l'algorithme d'optimisation

	J=6	J=7	J=8
temps (heure)	8.97	64.6	236

Une comparaison des performances d'erreur du processus de décodage itératif utilisant des coefficients de pondération ajustés selon l'algorithme de rétropropagation de l'erreur avec ceux obtenus avec la méthode expérimentale indique très peu de différence. Le temps d'exécution et la complexité de mise en oeuvre de la méthode utilisant la rétropropagation de l'erreur sont relativement très importants par rapport à la méthode expérimentale. Compte tenu de ces dernières observations, il apparaît très clairement que la méthode expérimentale demeure la plus efficace.

3.5 Définition au sens strict des codes convolutionnels doublement orthogonaux

Les codes convolutionnels doublement orthogonaux au sens large présentés à la section 3.2.1 ont pour inconvénient d'introduire dès la deuxième itération du processus de décodage des répétitions indésirables d'observables. L'effet immédiat de ces répétitions est de limiter la convergence du processus itératif vers de faibles performances d'erreur. Des coefficients de pondération sont nécessaires pour diminuer les effets néfastes engendrés par ces répétitions. Évidemment, l'introduction de ces coefficients ne se fait pas sans une augmentation non négligeable de la complexité du processus de décodage. Dans le but de réduire ces répétitions indésirables, une structure parallèle de codage est définie dans laquelle chaque symbole d'information est connecté à un seul symbole de parité d'une séquence de parité donnée. De cette façon, les équations de parité formées au décodage ne font plus intervenir les mêmes observables puisque toutes permutations des indices sont maintenant éliminées. Sous cette nouvelle condition, le code convolutionnel doublement orthogonal est alors défini au *sens strict* (CSO^2C-SS). Le code est alors spécifié par une matrice de valeurs entières de dimension $J \times J$ et le taux de codage $r_c = 1/2$ devient $J/2J$. Chaque entrée de cette matrice, $\alpha_{j,n}$ représente la position du $j^{ième}$ symbole d'information dans le régistre à décalage du codeur connecté au $n^{ième}$ symbole de parité. Ce processus d'encodage consiste à générer à la sortie du codeur les symboles de parité de la façon suivante :

$$p_{n,i} = \sum_{k=1}^{J} \oplus u_{k,i-\alpha_{k,n}} \quad , n = 1, 2, ..., J, i = 0, 1, 2, ... \tag{3.24}$$

La Figure 3.14 montre un exemple d'un codeur convolutionnel doublement orthogonal au sens strict de taux de codage $r_c = 3/6$, J=3, spécifié par la matrice $[\alpha_{j,n}]$ suivante :

$$[\alpha_{j,n}] = \begin{bmatrix} 0 & 0 & 5 \\ 1 & 3 & 0 \\ 5 & 2 & 0 \end{bmatrix} \tag{3.25}$$

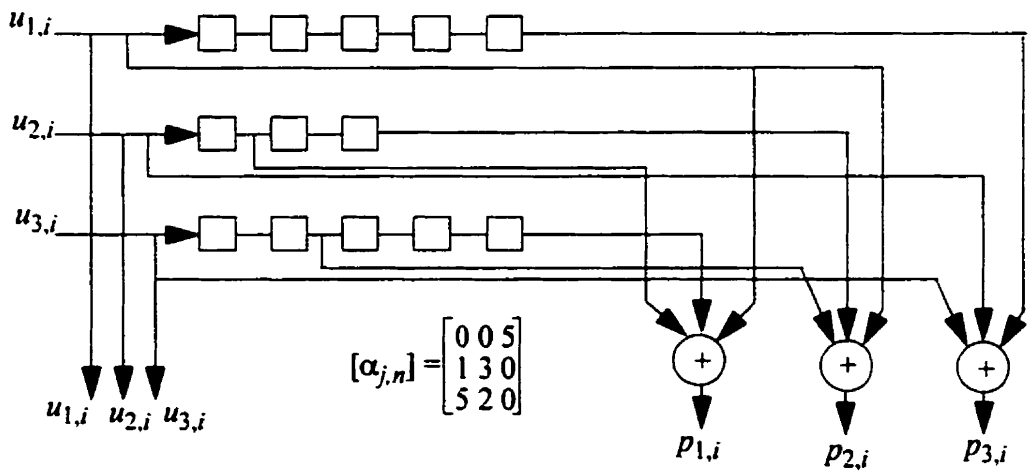


Figure 3.14 Exemple d'un codeur convolutionnel doublement orthogonal au sens strict, J=3, $r_c = 3/6$

En suivant un développement semblable à celui qui a conduit au résultat donné par (3.4), on obtient à la deuxième itération :

$$\lambda_{j,i}^{(2)} = y_{j,i}^{u} + \sum_{n=1}^{J} \Bigg(y_{n,i+\alpha_{j,n}}^{p} \Diamond \sum_{\substack{k=1,k\neq j \\ (\alpha_{j,n}-\alpha_{k,n})<0}}^{J} \Diamond \lambda_{k,(i+(\alpha_{j,n}-\alpha_{k,n}))}^{(2)} \Diamond \sum_{\substack{k=1,k\neq j \\ (\alpha_{j,n}-\alpha_{k,n})\geq 0}}^{J} \Diamond \Bigg(y_{k,(i+(\alpha_{j,n}-\alpha_{k,n}))}^{u} +$$

$$\sum_{\substack{s=1 \\ s\neq n}}^{J} \Bigg(y_{s,(i+\alpha_{j,n}-(\alpha_{k,n}-\alpha_{k,s}))}^{p} \Diamond \sum_{\substack{l=1,l\neq k \\ (\alpha_{l,s}-\alpha_{k,s})\geq 0}}^{J} \Diamond y_{l,(i+(\alpha_{j,n}-\alpha_{k,n})-(\alpha_{l,s}-\alpha_{k,s}))}^{u} \Diamond \tag{3.26}$$

$$\sum_{\substack{l=1,l\neq k \\ (\alpha_{l,s}-\alpha_{k,s})<0}}^{J} \Diamond \lambda_{l,(i+(\alpha_{j,n}-\alpha_{k,n})-(\alpha_{l,s}-\alpha_{k,s}))}^{(1)} \Bigg) \Bigg) \Bigg), j = 1, 2, \ldots, J,\ i = 1, 2, \ldots$$

À partir de (3.26), la définition des codes doublement orthogonaux au sens strict (CSO^{2}C-SS) est la suivante.

Définition 3. *Un code convolutionnel systématique de taux $r_c = J/2J = 1/2$ est doublement orthogonal au sens strict si, les éléments de la matrice $(\alpha_{j,n})$, satisfont aux conditions suivantes :*

1) les différences, $(\alpha_{j,n}-\alpha_{k,n})$, sont distinctes,

2) les différences de différences, $(\alpha_{j,n}-\alpha_{k,n})-(\alpha_{l,s}-\alpha_{k,s})$, sont distinctes et,

3) les différences de différences sont distinctes des différences.

pour toutes les combinaisons des indices $j, n, s \in \{1, 2, \ldots, J\}$ avec $s \neq n$, $k \neq j$ et $k \neq l$.

Grâce à la Définition 3, aucune répétition n'est possible à la deuxième itération et le décodage s'effectue sur un ensemble d'observables indépendants. Ainsi, comme nous le

verrons à la section 3.6, aucune modification de l'algorithme de décodage n'est nécessaire. Il faut aussi noter qu'avec les codes CSO^{2}C-SS, la latence pour une itération de décodage est proportionnelle à la valeur maximale $\max_{1 \le k, n \le J} \{\alpha_{k,n}\}$ qui représente aussi la longueur maximale des registres à décalage du codeur. Puisqu'il y a J registres à décalage, la latence d'une itération est donnée par $J \times \max_{1 \le k, n \le J} \{\alpha_{k,n}\}$. Minimiser cette dernière valeur est alors le critère principal utilisé dans la recherche des meilleurs codes CSO^{2}C-SS [21], [22].

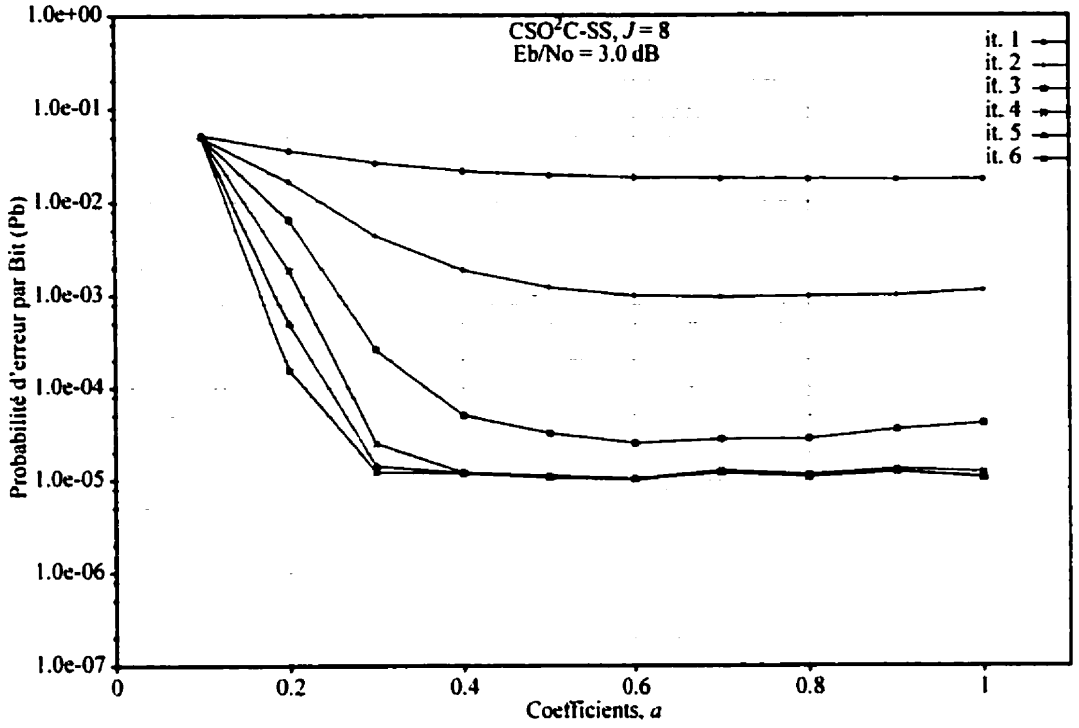
3.5.1 Construction des codes convolutionnels doublement orthogonaux au sens strict

Cette section résume très brièvement les méthodes et les résultats présentés dans [21] et [22] concernant la construction des CSO^{2}C-SS. Comme c'est aussi le cas pour les CSO^{2}C-WS, la construction des CSO^{2}C-SS requiert deux étapes : génération des codes et minimisation de la longueur de contrainte. Quoiqu'il soit toujours possible d'utiliser la géométrie projective [38], une méthode utilisant un paramètre aléatoire s'avère encore une fois plus efficace [21], [22]. La méthode de minimisation de la longueur de contrainte du code aussi proposée dans [21] est basée sur la technique développée par W. W. Wu [24]. Cette méthode consiste à effectuer des opérations arithmétiques simples sur les lignes et les colonnes de la matrice initiale $[\alpha_{j,n}]$. La procédure de minimisation est exécutée de manière itérative jusqu'à ce que les plus grands éléments de la matrice $[\alpha_{j,n}]$ résultante soient aussi petits que possibles. Les codes CSO^{2}C-SS étudiés dans cette thèse sont

fournis aux annexes de [21].

3.6 Décodage à seuil itératif des CSO^{2}C-SS

Le décodage itératif des codes CSO^{2}C-SS se fait simplement par l'application de l'algorithme itératif décrit par (3.26). En utilisant cette structure parallèle du codeur et en respectant les conditions données par la Définition 3, un décodage itératif efficace est possible puisque les observables sont indépendants sur au moins, les deux premières itérations. Cela implique naturellement que la corrélation entre les observables devient très faible. Par conséquent, l'usage de coefficients de pondération devrait être inutile. Cette affirmation se vérifie facilement par simulation en reprenant la méthode expérimentale exposée précédemment pour les codes CSO^{2}C-WS. Les résultats de simulation pour un code J=8 CSO^{2}C-SS présentés à la Figure 3.15 indiquent clairement que pour un rapport E_b/N_0 = 3 dB, aucune amélioration des performances d'erreur n'est possible pour des coefficients de pondération supérieurs à 0.3. Ce résultat se vérifie aussi pour de plus forts rapports signal à bruit. En d'autres mots, pour de forts rapports signal à bruit, les performances d'erreur que procurent les CSO^{2}C-SS sont insensibles aux changements des coefficients de pondération. Par conséquent, il n'est plus nécessaire d'avoir recours à une pondération des équations d'inversion de parité, $\overset{(\mu)}{}_{j,n,i}$, par des coefficients.



$$[\alpha_{j,n}] = \begin{bmatrix} 151 & 58 & 65 & 2263 & 0 & 377 & 386 & 320 \\ 0 & 87 & 14 & 553 & 650 & 2179 & 207 & 505 \\ 2212 & 340 & 62 & 0 & 2660 & 0 & 0 & 0 \\ 0 & 2542 & 787 & 45 & 0 & 210 & 208 & 165 \\ 1104 & 82 & 104 & 185 & 0 & 537 & 497 & 2311 \\ 196 & 59 & 158 & 894 & 0 & 661 & 367 & 1413 \\ 88 & 0 & 0 & 124 & 0 & 1478 & 2659 & 288 \\ 1958 & 68 & 62 & 304 & 0 & 744 & 1900 & 925 \end{bmatrix}$$

Figure 3.15 Variation des coefficients de pondération a pour un code J=8 CSO²C-SS, $E_b/N_o = 3$ dB

La Figure 3.16 montre les performances d'erreur en fonction de E_b/N_o pour le même code J=8 CSO²C-SS. On observe que pour des rapports signal à bruit E_b/N_o (i.e. $E_b/N_o \geq 4.0$ dB), après trois itérations, aucune amélioration des performances d'erreur n'est possible. D'autre part, est-il possible d'améliorer davantage les performances

d'erreur pour de plus faibles rapports signal à bruit E_b/N_0 ? Pour répondre à cette question, il suffit de reprendre la méthode expérimentale mais, pour de faibles valeurs de E_b/N_0.

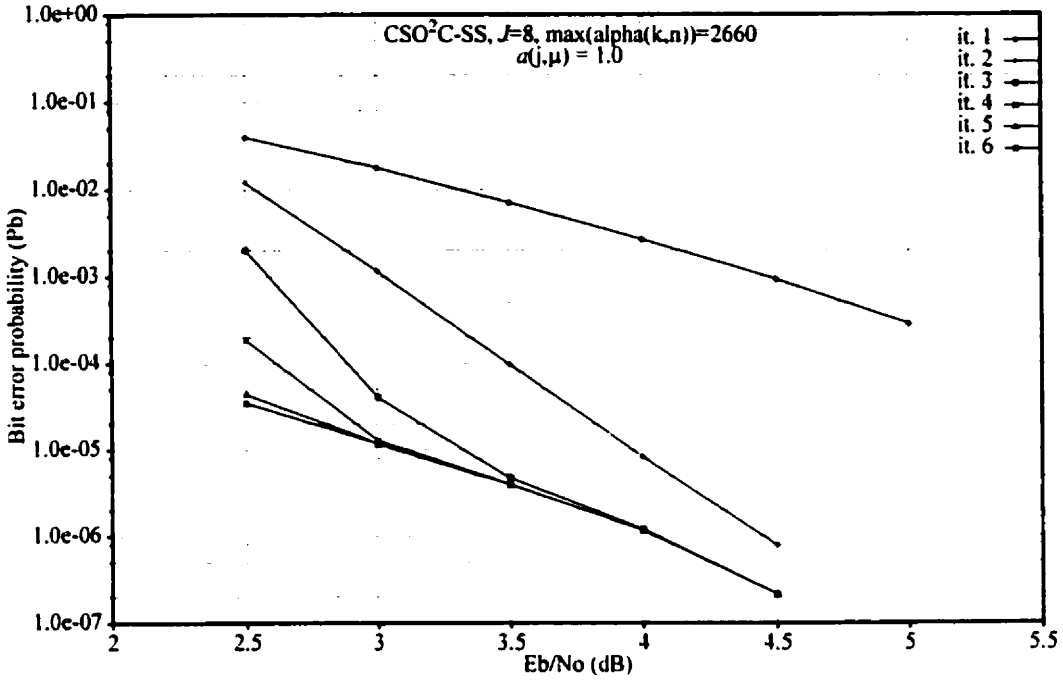


Figure 3.16 Résultats de simulation pour J=8, CSO^{2}C-SS, r_c = 8/16

La Figure 3.17 montre un exemple d'un résultat obtenu avec cette méthode expérimentale appliquée à un code J=8, CSO^{2}C-SS, r_c = 8/16 pour un rapport signal à bruit E_b/N_0=2.0 dB. On observe très clairement que pour un coefficient de pondération a = 1.0 la probabilité d'erreur après 6 itérations n'est que de 3×10^{-2} alors qu'avec un coefficient a^* de 0.4, la probabilité d'erreur obtenue après 6 itérations est d'environ 2×10^{-4} soit une amélioration d'environ deux ordres de grandeur.

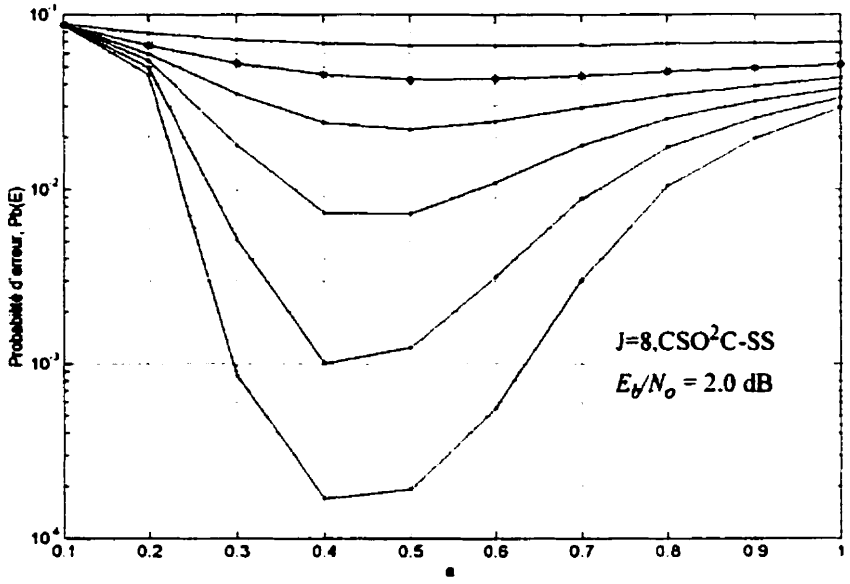


Figure 3.17 Variation des coefficients de pondération a pour un code J=8 CSO^{2}C-SS, E_b/N_0 = 2 dB

La nécessité de ces coefficients de pondération à de faibles valeurs du rapport signal à bruit E_b/N_0 se justifie par les mêmes arguments apportés à la section 3.4.1 et par le fait qu'en présence de bruit de grande puissance, les performances d'erreur sont fortement affectées par la non-orthogonalité des équations $\psi_{j,n,i}^{(\mu)}$ après la deuxième itération (μ>2). À l'Annexe V d'autres résultats de simulation des codes CSO^{2}C-SS sont fournis pour des valeurs de J allant de J=2 à 10 et où pour certains codes, des coefficients de pondération ont été obtenus par la procédure expérimentale.

3.7 Discussion et conclusion

Les codes convolutionnels doublement orthogonaux définis au sens strict ont une longueur de mémoire supérieure à celle des codes doublement orthogonaux définis au sens large. Pour de forts rapports signal à bruit, les CSO^{2}C-SS ne nécessitent que trois ou quatre itérations de décodage avant d'atteindre la convergence alors qu'il en faut plus de huit pour des CSO^{2}C-WS. Généralement, après avoir atteint la convergence, les performances d'erreur que procurent les codes CSO^{2}C-SS sont meilleures que celles obtenues avec des codes CSO^{2}C-WS. Le Tableau 3.4 montre, pour un rapport signal à bruit E_b/N_o = 3.5 dB, une comparaison des performances d'erreur que procurent les codes définis au sens large et au sens strict en fonction de leur latence respective évaluée après convergence. Comme l'indique le Tableau 3.4 et la Figure 3.18, pour une même valeur de J, les codes CSO^{2}C-SS procurent de meilleures performances d'erreur que les codes CSO^{2}C-WS. Cependant, cette amélioration des performances d'erreur se fait généralement au prix d'une augmentation importante de la latence.

Tableau 3.4 : Comparaison entre sens large (WS) et sens strict (SS) pour E_b/N_o = 3.5 dB

	CSO^{2}C-WS				CSO^{2}C-SS			
J	Lat./It. (bits)	Nb. It.	Lat. tot. (bits)	Pb	Lat/It (bits)	Nb. It.	Lat. tot. (bits)	Pb
6	180	8	1440	8×10^{-4}	363x6	4	8712	4×10^{-5}
7	228	8	1824	3.5×10^{-4}	1102x7	4	30856	1.1×10^{-5}
8	459	8	3672	1.2×10^{-5}	2660x8	4	85120	4×10^{-6}
9	912	8	7296	3×10^{-6}	5707x9	4	205452	10^{-6}
10	1835	8	14680	1.3×10^{-6}	12426x10	3 ~ 4	372780 ~ 497040	4×10^{-7}
11	2843	8	22744	10^{-6}	-----	-----	-----	-----
12	4951	7	34657	3×10^{-7}	-----	-----	-----	-----

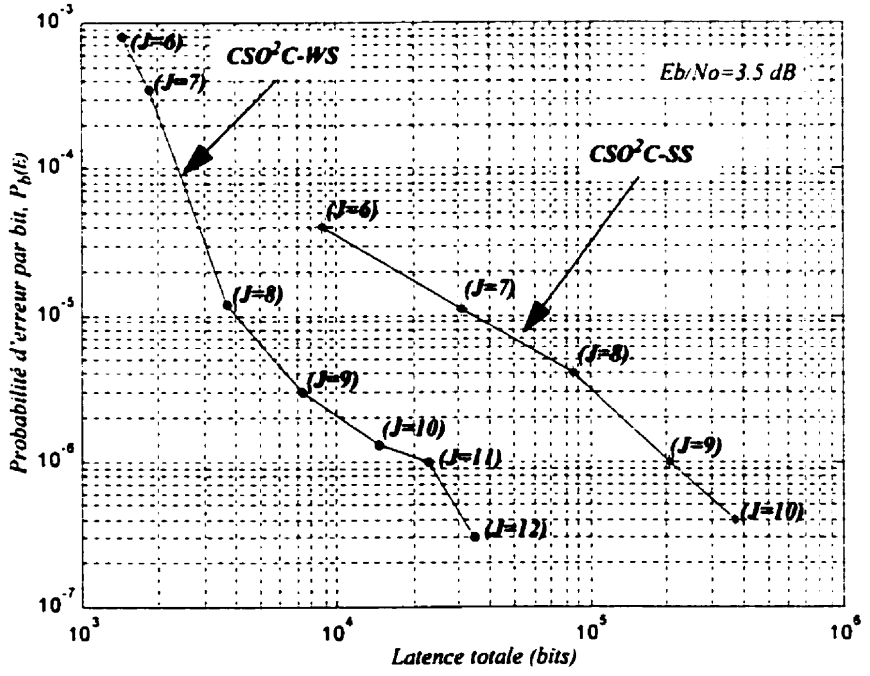


Figure 3.18 Comparaison entre les performances d'erreur des codes CSO^{2}C-WS et CSO^{2}C-SS en fonction de leur latence totale après convergence, E_b/N_o=3.5 dB

À de plus faibles rapports signal à bruit, la situation est différente. Par exemple, la

comparaison faite précédemment est reprise pour un rapport signal à bruit E_b/N_o = 2.5 dB et est montrée au Tableau 3.5 où certains résultats sont reportés à la Figure 3.19. À partir des résultats illustrés à la Figure 3.19, il est clair que les codes CSO^{2}C-SS offrent des performances d'erreur supérieures à celle obtenus avec des codes CSO^{2}C-WS. On observe une différence un peu inférieure à un ordre de grandeur entre les performances d'erreur des codes CSO^{2}C-SS avec J=6 et celle des codes CSO^{2}C-WS avec J=10. Cette différence augmente au fur et à mesure que la valeur de J des deux types de code augmente. Notons que dans le cas du code CSO^{2}C-SS, J=10, un ajustement des coefficients de pondération à une valeur de 0.6 a été nécessaire pour obtenir une probabilité d'erreur de 5×10^{-6} pour un E_b/N_o = 2.5 dB.

Tableau 3.5 : Comparaison entre sens large (WS) et sens strict (SS) pour E_b/N_o = 2.5 dB

	CSO^{2}C-WS				CSO^{2}C-SS			
J	Lat./It. (bits)	Nb. It.	Lat. tot. (bits)	Pb	Lat/It (bits)	Nb. It.	Lat. tot. (bits)	Pb
6	180	8	1440	1×10^{-3}	363x6	6	13068	3×10^{-4}
7	228	8	1824	4×10^{-4}	1102x7	6	42284	1×10^{-4}
8	459	8	3672	4×10^{-3}	2660x8	6	127680	3×10^{-5}
9	912	8	7296	1×10^{-3}	5707x9	6	308178	1.5×10^{-5}
10	1835	8	14680	1×10^{-3}	12426x10	6	745560	5×10^{-6}
11	2843	8	22744	3×10^{-3}	-----	-----	-----	-----
12	4951	10	49510	6×10^{-3}	-----	-----	-----	-----

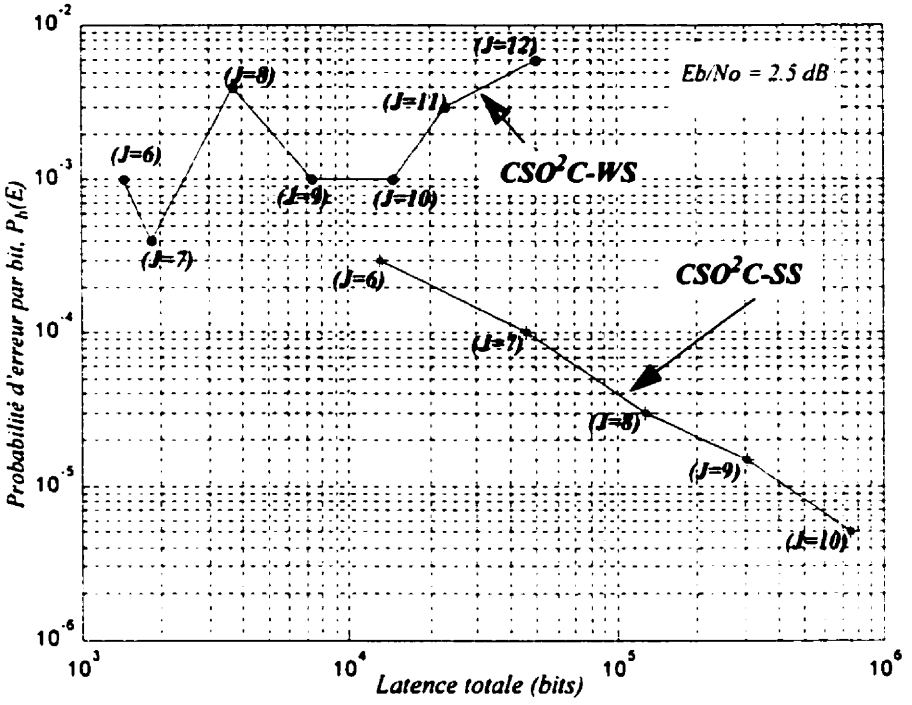


Figure 3.19 Comparaison entre les performances d'erreur des codes CSO^{2}C-WS et CSO^{2}C-SS en fonction de leur latence totale après convergence, E_b/N_o = 2.5 dB

Selon ces dernières observations, en présence d'un canal de transmission très bruité, c'est-à-dire lorsque le bruit du canal est de grande puissance, il est préférable d'utiliser un code CSO^{2}C-SS. D'autre part, pour un rapport signal à bruit plus fort, un code CSO^{2}C-WS procure de très bonnes performances d'erreur compte tenue de sa faible complexité et latence.

Dans ce chapitre, une méthode de décodage itératif sans entrelacement a été présentée. Cette technique étant basée sur les principes du décodage à seuil, des propriétés supplémentaires d'orthogonalité du code s'avèrent nécessaires pour s'assurer d'une indépendance des observables dans le processus de décodage itératif. Pour les deux

premières itérations, imposer l'indépendance des observables nécessite que le code convolutionnel soit doublement orthogonal. Le codage et le décodage itératif se fait alors sans faire intervenir d'entrelacement. Nous avons vu que ces codes convolutionnels doublement orthogonaux se définissent de deux façons : au sens large et au sens strict. Selon la première définition, la propriété de la double orthogonalité du code est partiellement satisfaite de sorte que le décodage s'effectue sur un ensemble d'observables dépendants. Cette dépendance des observables est causée par la présence de répétitions inévitables et indésirables des variables et cela, dès la deuxième itération. Des coefficients de pondération associés à chaque équation d'inversion de la parité s'avèrent très utiles pour améliorer les performances d'erreur des codes CSO^2C-WS. Les codes convolutionnels doublement orthogonaux peuvent être définis au sens strict en faisant appel à une structure parallèle du codeur. Avec cette définition, aucune répétition d'observables n'est possible à la deuxième itération. Le décodage s'effectue alors sur un ensemble d'observables indépendants jusqu'à la deuxième itération du processus. Au-delà de la deuxième itération, des répétitions d'observables se produisent ce qui amène le système de décodage itératif à converger plus rapidement. Pour de faibles rapports signal à bruit, des coefficients de pondération sont aussi nécessaires pour améliorer davantage les performances d'erreur. De façon générale, tous les coefficients de pondération peuvent être ajustés à la même valeur sans engendrer, à la dernière itération de décodage, une forte dégradation des performances d'erreur. Évidemment, il est toujours possible de développer des méthodes plus puissantes d'optimisation des coefficients de pondération à la condition de disposer d'un système de calcul de grande puissance.

CHAPITRE 4: ANALYSE DES PERFORMANCES D'ERREUR DU DÉCODAGE À SEUIL ITÉRATIF SANS ENTRELACEMENT

Dans le chapitre précédent, les codes convolutionnels doublement orthogonaux ont été définis au sens large et au sens strict. Le décodage d'un code convolutionnel doublement orthogonal se fait itérativement sans entrelacement en utilisant un décodage à seuil. Puisque le code est doublement orthogonal, il est clair qu'il se produit à toutes les itérations (μ>2) des répétitions d'observables. Par conséquent, la corrélation augmente entre les observables et les performances d'erreur s'en trouvent diminuées. Pour un code défini au sens large, ces répétitions sont nombreuses et affectent grandement les performances d'erreur. Des coefficients de pondérations sont alors nécessaires pour compenser les effets indésirables de ces répétitions. Pour des codes définis au sens strict, ces répétitions d'observables sont plus rares de sorte qu'une amélioration des performances d'erreur est possible par rapport aux codes CSO^{2}C-WS mais, au prix d'une augmentation de la longueur de contrainte. Sauf à de faibles rapports E_b/N_o, le décodage des CSO^{2}C-SS ne nécessite pas de coefficients de pondération.

Dans ce chapitre on propose une méthode permettant d'analyser et d'évaluer les performances d'erreur du processus itératif de décodage à seuil à sortie non-quantifiée [57]. Afin de simplifier l'analyse, on introduit de nouveau, le principe de codes convolutionnels multiplement orthogonaux définis au sens strict (CSO^{M}C-SS). Ce principe a déjà été introduit et utilisé au chapitre 3 dans le but d'analyser le comportement

du processus itératif de décodage à seuil en quantification ferme. L'ordre (M) de la multiplicité des CSO^{M}C-SS étant supérieur ou égal au nombre d'itérations du processus de décodage, on pose l'hypothèse que les variables aléatoires présentes dans le processus de décodage sont indépendantes à chaque itération. Cependant, il importe de noter qu'une telle extension de l'orthogonalité (d'ordre M>2) n'est habituellement pas nécessaire pour obtenir de bonnes performances d'erreur. Notons aussi que jusqu'à présent aucun de ces codes de multiplicité d'ordre M supérieur à deux n'a été recherché. Toutefois, cette analyse devrait fournir un moyen de prédire adéquatement la probabilité d'erreur par bit, vers laquelle le processus itératif de décodage à seuil à sortie non-quantifiée converge lorsque des codes CSO^{2}C-SS sont utilisés.

Dans la première section de ce chapitre, la méthode d'analyse est exposée en ne considérant qu'une seule itération. Selon l'hypothèse précédente, un code CSOC s'avère alors suffisant. La détermination de la fonction de densité de probabilité de la variable aléatoire qui résulte d'une opération addmin effectuée sur d'autres variables aléatoires constitue la partie la plus importante de cette méthode d'analyse. Cette fonction de densité de probabilité est développée pour le cas général d'un opérateur addmin de J entrées où chacune de ces entrées suit une distribution gaussienne. Dans la seconde section de ce chapitre, une extension de cette méthode d'analyse à un processus itératif de décodage sans entrelacement est présentée en considérant des codes CSO^{M}C-SS où M est supérieur ou égal au nombre d'itérations du processus de décodage. Les résultats obtenus théoriquement avec des codes CSO^{M}C-SS sont comparés à ceux obtenus par simulation

en considérant cette fois-ci, des codes CSO^2C-SS. Finalement, des conclusions sont présentées à la dernière section de ce chapitre.

4.1 Probabilité d'erreur par bit du décodage à seuil à sortie non-quantifiée pour des CSOC

La probabilité d'erreur par bit du processus de décodage à seuil à sortie non-quantifiée s'évalue difficilement car les estimations du symbole d'information à décoder fournies par les équations d'inversion de parité, $\psi_{j,i}$, ne sont pas identiquement distribuées [14]. Pour cette raison, les procédures habituelles permettant de manipuler des sommes de variables aléatoires indépendantes telles que la borne de Chernov ou le théorème limite centrale ne peuvent pas être, selon Massey [14], applicables. Dans [14], l'évaluation numérique de la probabilité d'erreur est essentiellement basée sur l'utilisation de fonctions d'énumération.

L'approche considérée dans ce chapitre consiste à poser l'hypothèse que les estimations fournies par les équations d'inversion de parité, $\psi_{j,i}$, quoiqu'elles ne soient pas identiquement distribuées (*i.d.*) sont toutefois indépendantes et *quasi-identiquement* distribuées. Le théorème limite centrale procure donc une bonne approximation de la fonction de distribution de la variable aléatoire λ_i. Cette approximation est tout particulièrement intéressante à de forts rapports signal à bruit E_b/N_o.

La probabilité d'erreur par bit du processus de décodage à seuil pour un code CSOC s'obtient, selon la règle de décision donnée par (2.24) comme suit :

$$P_b(E) = P\left\{\lambda_i \geq 0 | x_i^u = -1\right\} P\left\{x_i^u = -1\right\} + P\left\{\lambda_i < 0 | x_i^u = 1\right\} P\left\{x_i^u = 1\right\} \qquad (4.1)$$

où l'on rappelle que x_i^u représente le symbole d'information transmis à l'instant i et où la variable λ_i est donnée par (2.43). Puisque $P\left\{x_i^u = -1\right\} = P\left\{x_i^u = 1\right\} = 1/2$, la probabilité d'erreur par bit (4.1) devient :

$$P_b(E) = \frac{1}{2}\int_0^{\infty} f_\lambda(\lambda | x^u = -1) d\lambda + \frac{1}{2}\int_{-\infty}^{0} f_\lambda(\lambda | x^u = 1) d\lambda \qquad (4.2)$$

où $f_\lambda(\lambda | x^u = k)$, $k = 1$ ou -1, est la fonction de densité de probabilité de la variable aléatoire λ conditionnelle à $x^u = k$. Notons que l'aspect temporel est éliminé dans (4.2) car le processus aléatoire décrivant λ_i est, par hypothèse, stationnaire au sens large. Par symétrie, les probabilités $P\{\lambda \geq 0 | x^u = -1\}$ et $P\{\lambda < 0 | x^u = 1\}$ sont égales et donc la probabilité d'erreur par bit, $P_b(E)$, devient :

$$P_b(E) = \int_0^{\infty} f_\lambda(\lambda | x^u = -1) d\lambda = \int_{-\infty}^{0} f_\lambda(\lambda | x^u = 1) d\lambda \qquad (4.3)$$

Puisque λ s'obtient par une somme de variables aléatoires indépendantes, sa fonction de densité de probabilité (FDP) conditionnelle, $f_\lambda(\lambda | x^u = k)$, k=1 ou -1, se détermine en

effectuant la convolution multiple des FDP conditionnelles de chaque variable aléatoire qui appartient à la somme donnée par (2.43) :

$$f_{\lambda}(\lambda|x^{u}= k) = f_{\psi_1}(\lambda|x^{u}= k) \otimes \ldots \otimes f_{\psi_J}(\lambda|x^{u}= k) \otimes f_{y^{u}}(\lambda|x^{u}= k) \qquad (4.4)$$

où $\otimes$ représente l'opération de convolution, $f_{\psi_j}(\psi_j|x^{u}= k)$ est la FDP conditionnelle de $\psi_j, j = 1, 2, \ldots, J$, et où $f_{y^{u}}(y^{u}|x^{u}= k)$ est la FDP conditionnelle de la variable aléatoire y^{u} représentant le symbole d'information courant reçu. Pour une transmission à travers un canal à bruit blanc gaussien et additif, $f_{y^{u}}(y^{u}|x^{u}= k)$ est une fonction de densité de probabilité d'une variable aléatoire gaussienne de moyenne k et de variance $\sigma^{2}_{y^{u}} = N_o/2$:

$$f_{y^{u}}(y^{u}|x^{u}= k) = \frac{\exp\left\{-(y^{u}-k)^{2}/2\sigma^{2}_{y^{u}}\right\}}{\sqrt{2\pi\sigma^{2}_{y^{u}}}} \qquad (4.5)$$

La convolution multiple qui intervient dans (4.4) peut être difficile à évaluer car, il est d'une part nécessaire de connaître les FDP $f_{\psi_j}(\psi_j|x^{u}= k)$ et d'autre part, la complexité de calcul de cette convolution multiple augmente avec J. Les variables aléatoires faisant partie de la somme (2.43) ne sont pas *identiquement distribuées*. Toutefois, elles ont des distributions semblables et la somme (2.43) a tendance à se distribuer selon un fonction gaussienne. Comme le montre la Figure 4.1, cette assertion se vérifie facilement par simulation. Des résultats de simulations ont montré que la forme gaussienne de la distribution est peu sensible aux valeurs de E_b/N_o.

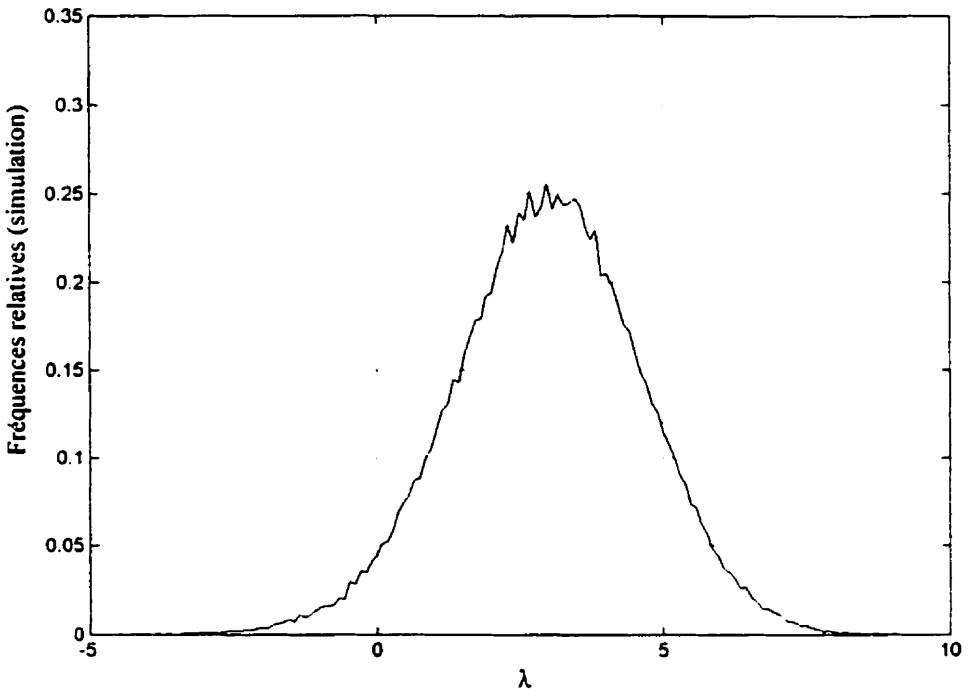


Figure 4.1 Simulation de la FDP de λ d'un code J=6 CSOC, E_b/N_o=2.0dB, x^u=1.0

Cette observation nous permet donc d'écrire l'approximation suivante, issue du théorème limite centrale :

$$f_\lambda(\lambda|x^u = k) \cong \frac{\exp\left\{-(\lambda-\bar{\lambda})^2/2\sigma_\lambda^2\right\}}{\sqrt{2\pi\sigma_\lambda^2}} \tag{4.6}$$

où $\bar{\lambda}$ et σ_λ^2 représentent la moyenne et la variance de λ respectivement. La valeur moyenne $\bar{\lambda}$ s'obtient par la somme des valeurs moyennes des variables aléatoires qui composent la somme λ :

$$\bar{\lambda} = E\left\{ y^u + \sum_{j=1}^{J} \psi_j \middle| x^u = k \right\} = \overline{y^u} + \sum_{j=1}^{J} \overline{\psi_j} = k + \sum_{j=1}^{J} \overline{\psi_j} \tag{4.7}$$

avec k=1 ou -1, et où $\overline{\psi_j}$ s'obtient en calculant l'espérance mathématique de ψ_j conditionnelle à $x^u = k$. Le canal étant à bruit blanc additif et gaussien de moyenne nulle, la moyenne de y^u conditionnelle à $x^u = k$ donne nécessairement la valeur de k. Puisque le code est supposé orthogonal (CSOC), toutes les variables aléatoires sont indépendantes de sorte que la variance σ_λ^2 s'exprime aussi comme étant la somme des variances des variables aléatoires qui composent la somme λ. La variance σ_λ^2 s'écrit donc :

$$\sigma_\lambda^2 = E\left\{ \left(y^u + \sum_{j=1}^{J} \psi_j - \bar{\lambda} \right)^2 \middle| (x^u = k) \right\} = \frac{N_o}{2} + \sum_{j=1}^{J} \sigma_{\psi_j}^2 \tag{4.8}$$

où $\sigma_{\psi_j}^2 = E\left\{ (\psi_j - \overline{\psi_j})^2 \middle| (x^u = k) \right\}$. En se référant à (4.3), et en ne considérant que la condition $x^u = 1$, la probabilité d'erreur par bit, $P_b(E)$, s'obtient approximativement par l'expression suivante :

$$P_b(E) \cong \frac{1}{\sqrt{2\pi\sigma_\lambda^2}} \int_{-\infty}^{0} \exp\left\{ -\frac{(\lambda - \bar{\lambda})^2}{2\sigma_\lambda^2} \right\} d\lambda = Q\left(\frac{\bar{\lambda}}{\sigma_\lambda} \right) \tag{4.9}$$

où $Q(z) = \frac{1}{\sqrt{2\pi}} \int_{z}^{\infty} \exp\left\{-\frac{t^2}{2}\right\} dt$. Dans le but d'évaluer (4.9), (4.8) et (4.7), il est nécessaire de connaître la FDP de chaque variable aléatoire ψ_j. Cette variable aléatoire qui représente une estimation du symbole d'information à décoder s'obtient, comme le montre (2.43), en utilisant un opérateur addmin de J entrées.

4.1.1 Fonction de densité de probabilité d'un opérateur addmin à deux entrées

Dans cette section, on développe une expression de la FDP, $f_{\psi}(\varphi)$, de la variable aléatoire ψ qui résulte d'une opération addmin sur deux variables aléatoires. La variable aléatoire ψ s'écrit en posant N=2 dans (2.41) de la façon suivante :

$$\begin{aligned} \psi &= (-1)^{N+1} \prod_{i=1}^{N} sign(L(\xi_i)) \min_{1 \le i \le N} \{|L(\xi_i)|\} \\ &= -\prod_{i=1}^{2} sign(L(\xi_i)) \min_{1 \le i \le 2} \{|L(\xi_i)|\} \end{aligned} \tag{4.10}$$

où chaque variable aléatoire z_l suit une distribution gaussienne de moyenne m_l et de variance σ_l^2.Tout d'abord, posons $F_{\psi}(\varphi) = P(\psi \le \varphi)$ comme étant la fonction de répartition de la variable ψ. La nature de () impose la division du problème de la détermination de $f_{\psi}(\varphi)$ en deux parties : le cas où $\varphi \ge 0$ et celui où $\varphi \le 0$. Considérons tout d'abord le premier cas où $\varphi \ge 0$. En se référant à (), définissons la variable aléatoire binaire $s_g = -sign(z_1)sign(z_2)$. Ainsi, s_g prend la valeur +1 ou -1 en fonction des

signes de z_1 et de z_2. On peut aussi définir l'ensemble d'événements $A^+ = \{\psi \le \varphi | \varphi \ge 0\}$ qui s'écrit, en se référant à (), de la manière suivante :

$$A^+ = \{|z_1| \le \varphi\} \cap \{s_g \ge 0\} \cup \{|z_2| \le \varphi\} \cap \{s_g \ge 0\} \cup$$

$$\{-|z_1| \le \varphi\} \cap \{s_g \le 0\} \cup \{-|z_2| \le \varphi\} \cap \{s_g \le 0\} \tag{4.11}$$

où dans (4.11) l'ensemble $\{s_g \ge 0\} = \{((z_1 \le 0) \cap (z_2 \ge 0)) \cup ((z_1 \ge 0) \cap (z_2 \le 0))\}$ et où son complément est l'ensemble :

$$\{s_g \le 0\} = \{((z_1 \le 0) \cap (z_2 \le 0)) \cup ((z_1 \ge 0) \cap (z_2 \ge 0))\} \tag{4.12}$$

Puisque $\varphi \ge 0$, il est aussi possible d'écrire la relation suivante :

$$\{-|z_1| \le \varphi\} \cap \{s_g \le 0\} \cup \{-|z_2| \le \varphi\} \cap \{s_g \le 0\}$$

$$= \{\{-|z_1| \le \varphi\} \cup \{-|z_2| \le \varphi\}\} \cap \{s_g \le 0\} = \{s_g \le 0\} \tag{4.13}$$

de sorte qu'en remplaçant (4.13) dans (4.11), l'ensemble d'événements A^+ se reduit à :

$$A^+ = \{\{|z_1| \le \varphi\} \cup \{|z_2| \le \varphi\}\} \cap \{s_g \ge 0\} \cup \{s_g \le 0\} \tag{4.14}$$

Une représentation graphique de l'ensemble A^+ dans un plan z_1z_2 est montrée à la Figure 4.2.

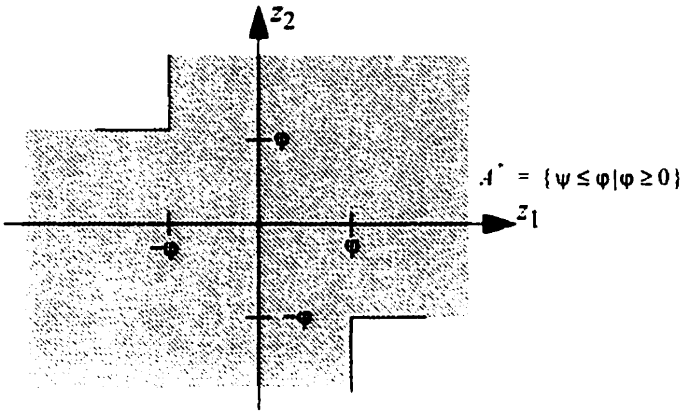


Figure 4.2 Représentation graphique dans le plan z_1z_2 de l'ensemble A^+

Considérons maintenant le deuxième cas où $\varphi \le 0$. Comme précédemment, en utilisant (), on définit l'ensemble des événements $A^- = \{\psi \le \varphi | \varphi \le 0\}$ de la manière suivante :

$$A^- = \{\{|z_1| \ge -\varphi\} \cap \{|z_2| \ge -\varphi\}\} \cap \{s_g \le 0\} \tag{4.15}$$

La relation (4.15) se vérifie facilement sachant que pour $\varphi \le 0$ et $s_g \le 0$, $min\{|z_1|, |z_2|\} \ge -\varphi \Rightarrow |z_1| \ge -\varphi$ et $|z_2| \ge -\varphi$. La Figure 4.3 montre une représentation graphique des points appartenant à l'ensemble A^- dans le plan z_1z_2.

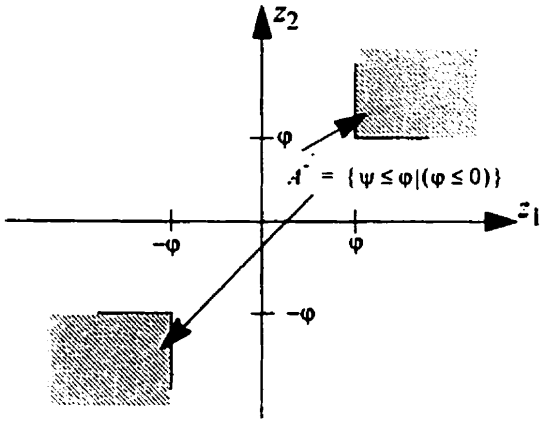


Figure 4.3 Représentation graphique de A^- dans le plan $z_1 z_2$

La fonction de répartition $F_\psi(\varphi) = P(\psi \le \varphi)$ s'obtient en utilisant les deux relations (4.14) et (4.15) de la manière suivante :

$$F_\psi(\varphi) = \begin{cases} P(A^+) & \text{si } \varphi \ge 0 \\ P(A^-) & \text{si } \varphi \le 0 \end{cases} \tag{4.16}$$

Lorsque $\varphi \ge 0$, on remarque, à partir de la Figure 4.2 et de la relation (4.14), que la probabilité $P(A^+)$ se transforme pour devenir :

$$P(A^+) = 1 - P\left\{\overline{A^+}\right\} \tag{4.17}$$

$$= 1 - P\{(z_1 \le -\varphi) \cap (z_2 \ge \varphi) \cup (z_1 \ge \varphi) \cap (z_2 \le -\varphi)\}$$

De plus, puisque les événements $(z_1 \le -\varphi) \cap (z_2 \ge \varphi)$ sont disjoints des événements $(z_1 \ge \varphi) \cap (z_2 \le -\varphi)$ et que les variables aléatoires z_1 et z_2 sont indépendantes, (4.17) devient :

$$P(A^+) = 1 - [P(z_1 \le -\varphi)P(z_2 \ge \varphi) + P(z_1 \ge \varphi)P(z_2 \le -\varphi)] \tag{4.18}$$

De la même manière, selon la représentation graphique de la Figure 4.3 et la relation (4.15), la probabilité $P(A^-)$ s'exprime comme suit :

$$P(A^-) = P(z_1 \ge -\varphi)P(z_2 \ge -\varphi) + P(z_1 \le \varphi)P(z_2 \le \varphi) \tag{4.19}$$

Ainsi, pour obtenir la FDP, $f_\psi(\varphi)$, la dérivée de (4.17) doit être déterminée. En combinant les équations (4.18) et (4.19) à (4.17) et en prenant la dérivée, on trouve :

$$f_\psi(\varphi) = \frac{\mathrm{d}}{\mathrm{d}\varphi}F_\psi(\varphi) = \begin{cases} -\dfrac{dP(z_1 \le -\varphi)}{d\varphi}P(z_2 \ge \varphi) - P(z_1 \le -\varphi)\dfrac{dP(z_2 \ge \varphi)}{d\varphi} - \\ \dfrac{dP(z_1 \ge \varphi)}{d\varphi}P(z_2 \le -\varphi) - P(z_1 \ge \varphi)\dfrac{dP(z_2 \le -\varphi)}{d\varphi}, & \text{si } \varphi \ge 0 \\ \dfrac{dP(z_1 \ge -\varphi)}{d\varphi}P(z_2 \ge -\varphi) + P(z_1 \ge -\varphi)\dfrac{dP(z_2 \ge -\varphi)}{d\varphi} + \\ \dfrac{dP(z_1 \le \varphi)}{d\varphi}P(z_2 \le \varphi) + P(z_1 \le \varphi)\dfrac{dP(z_2 \le \varphi)}{d\varphi}, & \text{si } \varphi \le 0 \end{cases} \tag{4.20}$$

Les deux variables aléatoires z_1 et z_2 suivent une distribution gaussienne de moyenne m_1 et m_2 et de variances σ_1^2 et σ_2^2 respectivement. Par conséquent les probabilités

nécessaires à l'évaluation de (4.20) peuvent être obtenues. Pour $\varphi \geq 0$, on calcule les deux relations suivantes :

$$P(z_l \leq -\varphi) = \frac{1}{\sqrt{2\pi\sigma_l^2}} \int_{-\infty}^{-\varphi} \exp\left\{-\frac{(z_l - m_l)^2}{2\sigma_l^2}\right\} dz_l = Q\left(\frac{\varphi + m_l}{\sigma_l}\right) \tag{4.21}$$

et

$$P(z_l \geq \varphi) = \frac{1}{\sqrt{2\pi\sigma_l^2}} \int_{\varphi}^{\infty} \exp\left\{-\frac{(z_l - m_l)^2}{2\sigma_l^2}\right\} dz_l = Q\left(\frac{\varphi - m_l}{\sigma_l}\right) \tag{4.22}$$

Dans le but d'évaluer (4.20), les dérivées de (4.21) et (4.22) sont obtenues et sont données par les relations suivantes :

$$\frac{d}{d\varphi} Q\left(\frac{\varphi \pm m_l}{\sigma_l}\right) = -\frac{1}{\sqrt{2\pi\sigma_l^2}} \exp\left\{-\frac{(\varphi \pm m_l)^2}{2\sigma_l^2}\right\} \tag{4.23}$$

Pour le seconde partie de (4.20), c'est-à-dire, lorsque $\varphi \leq 0$, on obtient aussi les trois relations suivantes :

$$P(z_l \geq -\varphi) = \frac{1}{\sqrt{2\pi\sigma_l^2}} \int_{-\varphi}^{\infty} \exp\left\{-\frac{(-z_l - m_l)^2}{2\sigma_l^2}\right\} dz_l = Q\left(\frac{-\varphi - m_l}{\sigma_l}\right) \tag{4.24}$$

$$P(z_l \leq \varphi) = \frac{1}{\sqrt{2\pi\sigma_l^2}} \int_{-\infty}^{\varphi} \exp\left\{-\frac{(-z_l - m_l)^2}{2\sigma_l^2}\right\} dz_l = Q\left(\frac{-\varphi + m_l}{\sigma_l}\right) \tag{4.25}$$

$$\frac{\mathrm{d}}{\mathrm{d}\varphi}Q\left(\frac{-\varphi \pm m_l}{\sigma_l}\right) = -\frac{1}{\sqrt{2\pi\sigma_l^2}}\exp\left\{-\frac{(-\varphi \pm m_l)^2}{2\sigma_l^2}\right\} \tag{4.26}$$

En remplaçant (4.21) à (4.26) dans (4.20), la fonction de densité de probabilité de la variable aléatoire φ qui résulte d'une opération addmin entre les deux variables aléatoires gaussiennes z_1 et z_2 devient,

$$f_\psi(\varphi) = \begin{cases} \displaystyle\sum_{l=1}^{2}\left[\frac{1}{\sqrt{2\pi\sigma_l^2}}\exp\left\{-\frac{(|\varphi| + m_l)^2}{2\sigma_l^2}\right\}\prod_{j=1, j\neq l}^{2} Q\left(\frac{|\varphi| - m_j}{\sigma_j}\right) + \right. \\ \left. \dfrac{1}{\sqrt{2\pi\sigma_l^2}}\exp\left\{-\dfrac{(|\varphi| - m_l)^2}{2\sigma_l^2}\right\}\displaystyle\prod_{j=1, j\neq l}^{2} Q\left(\frac{|\varphi| + m_j}{\sigma_j}\right)\right], & \text{si } \varphi \geq 0 \\ \displaystyle\sum_{l=1}^{2}\left[\frac{1}{\sqrt{2\pi\sigma_l^2}}\exp\left\{-\frac{(|\varphi| - m_l)^2}{2\sigma_l^2}\right\}\prod_{j=1, j\neq l}^{2} Q\left(\frac{|\varphi| - m_j}{\sigma_j}\right) + \right. \\ \left. \dfrac{1}{\sqrt{2\pi\sigma_l^2}}\exp\left\{-\dfrac{(|\varphi| + m_l)^2}{2\sigma_l^2}\right\}\displaystyle\prod_{j=1, j\neq l}^{2} Q\left(\frac{|\varphi| + m_j}{\sigma_j}\right)\right], & \text{si } \varphi \leq 0 \end{cases} \tag{4.27}$$

4.1.2 Généralisation de la fonction de densité de probabilité pour un opérateur addmin à *J* entrées

Dans cette section, une expression générale de la FDP, $f_\psi(\varphi)$, qui résulte d'une opération addmin sur J variables aléatoires est déterminée. Selon (2.41), la variable aléatoire ψ s'écrit comme suit :

$$\psi = \sum_{l=1}^{J} z_l = (-1)^{J+1}\left(\prod_{l=1}^{J} sign[z_l]\right) \min_{1 \le l \le J}\{|z_l|\} \tag{4.28}$$

Afin d'obtenir cette fonction de densité de probabilité, reprenons tout d'abord l'exemple traité à la section précédente où le nombre d'entrées de l'opérateur addmin est J=2. Nous avons déterminé, à la section 4.1.1, les probabilités $P(\bar{A}^+) = 1 - P(A^+)$ et $P(A^-)$ nécessaires au calcul de la fonction de répartition $F_\psi(\varphi)$. Pour chacune des deux variables aléatoires z_1 et z_2 que constituent les entrées de l'opérateur addmin, on peut définir les deux probabilités suivantes comme étant des fonctions de φ:

$$p_i(\varphi) = P(z_i \ge \varphi),\ i=1, 2 \tag{4.29}$$

$$q_i(\varphi) = P(z_i \le -\varphi),\ i=1, 2 \tag{4.30}$$

Les probabilités $P(\bar{A}^+) = 1 - P(A^+)$ et $P(A^-)$ données par (4.18) et (4.19) peuvent s'écrire en fonction de (4.29) et (4.30), ce qui donne :

$$\begin{aligned} P(\bar{A}^+) &= P(z_1 \le -\varphi)P(z_2 \ge \varphi) + P(z_1 \ge \varphi)P(z_2 \le -\varphi) \\ &= q_1(\varphi)p_2(\varphi) + p_1(\varphi)q_2(\varphi) \end{aligned} \tag{4.31}$$

et

$$\begin{aligned} P(A^-) &= P(z_1 \ge -\varphi)P(z_2 \ge -\varphi) + P(z_1 \le \varphi)P(z_2 \le \varphi) \\ &= p_1(-\varphi)p_2(-\varphi) + q_1(-\varphi)q_2(-\varphi) \end{aligned} \tag{4.32}$$

Il est maintenant utile de définir une variable auxiliaire aléatoire binaire $H_i(\varphi)$ qui correspond à la $i^{ième}$ entrée z_i, i=1, 2, comme suit :

$$H_i(\varphi) = \begin{cases} 1, & \text{si } z_i \geq \varphi \\ 0, & \text{si } z_i \leq -\varphi \end{cases} \tag{4.33}$$

Avec cette variable aléatoire binaire, on forme une fonction d'énumération $g_i(W, \varphi)$ de forme polynômiale dont le coefficient de W^h, h=0 ou 1, est la probabilité que la variable aléatoire $H_i(\varphi)$ soit égale à h. Cette fonction d'énumération s'écrit alors comme suit :

$$g_i(W, \varphi) = p_i(\varphi)W + q_i(\varphi) \tag{4.34}$$

où W est une variable «muette» dont l'exposant indique la valeur que prend la variable aléatoire $H_i(\varphi)$. Puisque les variables aléatoires z_i sont statistiquement indépendantes, les variables aléatoires $H_i(\varphi)$ le sont aussi de sorte que l'on peut former la fonction d'énumération $G(W, \varphi)$ qui correspond à la somme des variables $H_i(\varphi)$. Cette fonction d'énumération $G(W, \varphi)$ s'obtient en effectuant le produit $g_1(W, \varphi)g_2(W, \varphi)$. ce qui donne le polynôme de degrée 2 suivant :

$$\begin{aligned} G(W, \varphi) &= g_1(W, \varphi)g_2(W, \varphi) \\ &= \sum_{i=0}^{2} b_i(\varphi)W^i = b_2(\varphi)W^2 + b_1(\varphi)W + b_0(\varphi) \\ &= p_1(\varphi)p_2(\varphi)W^2 + (p_1(\varphi)q_2(\varphi) + q_1(\varphi)p_2(\varphi))W + q_1(\varphi)q_2(\varphi) \end{aligned} \tag{4.35}$$

Dans cette fonction d'énumération donnée par (4.35), le coefficient de W^h, h=0, 1, 2, représente la probabilité que la somme $H_1(\varphi) + H_2(\varphi) = h$. On remarque rapidement

que l'expression donnée par (4.31) correspond au coefficient $b_1(\varphi)$ dans (4.35) et s'obtient facilement par :

$$b_1(\varphi) = P(\bar{A}^+) = \frac{G(1,\varphi) - G(-1,\varphi)}{2} = \sum_{i \text{ impair}} b_i(\varphi) \tag{4.36}$$

De la même manière, la probabilité $P(A^-)$ donnée par (4.32) s'obtient à partir de (4.35) en effectuant la somme des coefficients de la fonction $G(W, -\varphi)$ ayant un indice pair. Par conséquent, on obtient :

$$P(A^-) = b_2(-\varphi) + b_0(-\varphi) = \frac{G(1,-\varphi) + G(-1,-\varphi)}{2} = \sum_{i \text{ pair}} b_i(-\varphi) \tag{4.37}$$

En utilisant ces deux dernières relations, (4.36) et (4.37), la fonction de répartition $F_\psi(\varphi)$ peut être déterminée par :

$$F_\psi(\varphi) = \begin{cases} 1 - \dfrac{G(1,\varphi) - G(-1,\varphi)}{2}, & \varphi \geq 0 \\ \dfrac{G(1,-\varphi) + G(-1,-\varphi)}{2}, & \varphi \leq 0 \end{cases} \tag{4.38}$$

La FDP, $f_\psi(\varphi)$, s'obtient, comme auparavent, en effectuant la dérivée de $F_\psi(\varphi)$ dont le résultat est :

$$f_\psi(\varphi) = \begin{cases} -\dfrac{G'(1,\varphi) - G'(-1,\varphi)}{2}, & \varphi \geq 0 \\ \dfrac{G'(1,-\varphi) + G'(-1,-\varphi)}{2}, & \varphi \leq 0 \end{cases} \tag{4.39}$$

où $G'(W, \varphi)$ signifie la dérivée de $G(W, \varphi)$ par rapport à φ. On peut montrer facilement que (4.39) donne le même résultat que (4.27). Il est aussi possible de déduire à partir de (4.39), la FDP $f_{\psi}(\varphi)$, dans le cas où $J > 2$ en généralisant (4.35) comme suit :

$$G(W, \varphi) = \prod_{i=1}^{J} g_i(W, \varphi) = \sum_{i=0}^{J} b_i(\varphi) W^i \tag{4.40}$$

où $g_i(W, \varphi)$ est donnée par (4.34) pour i=1, 2, ..., J. Cela provient du fait que le résultat d'une opération addmin sur J variables aléatoires s'obtient aussi par une cascade de J-1 opérations addmin effectuées sur deux variables aléatoires. Par exemple, considérons une opération addmin à trois entrées. Le résultat de cette opération peut être obtenu par une cascade de deux opérateurs addmin de deux entrées comme l'illustre la Figure 4.4.

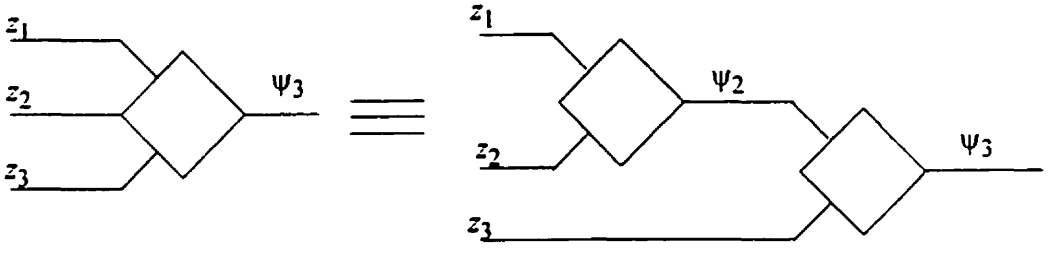


Figure 4.4 Opérateur addmin à trois entrées réalisé à partir d'une cascade de deux opérateurs addmin à deux entrées

La sortie du premier opérateur addmin de la cascade, identifiée par ψ_2 est combinée, dans un deuxième opérateur addmin avec l'entree z_3 pour produire la sortie désirée ψ_3. La fonction de distribution de la sortie dénotée par ψ_2 est donnée par (4.16) et par

conséquent, la fonction de distribution de la sortie ψ_3 s'obtient aussi en se référant à (4.16), c'est-à-dire :

$$F_{\psi_3}(\varphi) = \begin{cases} 1-P_3(\bar{A}^+) & \text{si } \varphi \geq 0 \\ P_3(A^-) & \text{si } \varphi \leq 0 \end{cases} \tag{4.41}$$

où $P_3(\bar{A}^+)$ et $P_3(A^-)$ sont données par :

$$P_3(\bar{A}^+) = P(\psi_2 \leq -\varphi)P(z_3 \geq \varphi) + P(\psi_2 \geq \varphi)P(z_3 \leq -\varphi), \quad \varphi \geq 0 \tag{4.42}$$

$$P_3(A^-) = P(\psi_2 \geq -\varphi)P(z_3 \geq -\varphi) + P(\psi_2 \leq \varphi)P(z_3 \leq \varphi), \quad \varphi \leq 0 \tag{4.43}$$

En se référant à (4.16), on trouve les probabilités $P(\psi_2 \leq -\varphi) = F_{\psi_2}(-\varphi)$ et $P(\psi_2 \geq \varphi) = 1 - F_{\psi_2}(\varphi)$ avec $\varphi \geq 0$ dans les deux cas. En utilisant (4.32), la fonction $F_{\psi_2}(-\varphi)$ est égale à la probabilité $P(A^-)$ donnée par (4.37) dans laquelle on remplace φ par $-\varphi$ pour obtenir la relation suivante :

$$F_{\psi_2}(-\varphi) = b_2(\varphi) + b_0(\varphi) = \frac{G(1,\varphi) + G(-1,\varphi)}{2} \tag{4.44}$$

$$= \frac{1}{2}\left[\prod_{i=1}^{2} g_i(W,\varphi)\Big|_{W=1} + \prod_{i=1}^{2} g_i(W,\varphi)\Big|_{W=-1}\right], \quad \varphi \geq 0$$

De plus, selon la relation (4.16), la probabilité $P(\psi_2 \geq \varphi) = 1 - F_{\psi_2}(\varphi)$ est égale à $1-(1-P(\bar{A}^+)) = P(\bar{A}^+)$ lorsque $\varphi \geq 0$. Cette probabilité s'écrit aussi en utilisant une fonction d'énumération semblable à (4.36) :

$$P(\bar{A}^{+}) = b_1(\varphi) = \frac{G(1,\varphi) - G(-1,\varphi)}{2}$$

$$= \frac{1}{2}\left[\prod_{i=1}^{2} g_i(W,\varphi)\Big|_{W=1} - \prod_{i=1}^{2} g_i(W,\varphi)\Big|_{W=-1}\right], \quad \varphi \geq 0 \tag{4.45}$$

En remplaçant (4.44) et (4.45) dans (4.42) et en utilisant les deux définitions (4.29) et (4.30) avec le fait que selon (4.34), $g_3(W,\varphi) = p_3(\varphi)W + q_3(\varphi)$, la probabilité $P_3(\bar{A}^{+})$ devient :

$$P_3(\bar{A}^{+}) = \frac{1}{2}\left[\prod_{i=1}^{3} g_i(W,\varphi)\Big|_{W=1} - \prod_{i=1}^{3} g_i(W,\varphi)\Big|_{W=-1}\right], \quad \varphi \geq 0 \tag{4.46}$$

Dans le cas général d'un opérateur addmin à J entrées, on détermine par induction la probabilité $P_J(\bar{A}^{+})$ de la manière suivante :

$$P_J(\bar{A}^{+}) = \frac{1}{2}\left[\prod_{i=1}^{J} g_i(W,\varphi)\Big|_{W=1} - \prod_{i=1}^{J} g_i(W,\varphi)\Big|_{W=-1}\right], \quad \varphi \geq 0 \tag{4.47}$$

Pour $\varphi \leq 0$, la probabilité $P_J(\bar{A}^{-})$ s'obtient en suivant le même raisonnement qui a conduit à (4.47) de sorte que :

$$P_J(\bar{A}^{-}) = \frac{1}{2}\left[\prod_{i=1}^{J} g_i(W,-\varphi)\Big|_{W=1} + \prod_{i=1}^{J} g_i(W,-\varphi)\Big|_{W=-1}\right], \quad \varphi \leq 0 \tag{4.48}$$

Ainsi, en se référant à (4.41) mais, pour un opérateur addmin de J entrées, la fonction de répartition $F_{\psi_J}(\varphi)$ est :

$$F_{\psi_J}(\varphi) = \begin{cases} 1 - P_J(\bar{A}^+) = 1 - \frac{1}{2}\left[\prod_{i=1}^{J} g_i(W,\varphi)\Big|_{W=1} - \prod_{i=1}^{J} g_i(W,\varphi)\Big|_{W=-1}\right], & \varphi \geq 0 \\ P_J(A^-) = \frac{1}{2}\left[\prod_{i=1}^{J} g_i(W,-\varphi)\Big|_{W=1} + \prod_{i=1}^{J} g_i(W,-\varphi)\Big|_{W=-1}\right], & \varphi \leq 0 \end{cases} \tag{4.49}$$

Finalement, la fonction de densité de probabilité pour un opérateur addmin de J entrées est la dérivée de $F_{\psi_J}(\varphi)$ par rapport à φ. Si chacune des J entrées z_i, $i = 1, \ldots, J$, de l'opérateur addmin suit une distribution gaussienne de moyenne m_i et de variance σ_i^2, la FDP, $f_{\psi_J}(\varphi)$ s'écrit de la manière suivante :

$$
f_{\psi_J}(\varphi) = \begin{cases}
\frac{1}{2}\sum_{i=1}^{J}\left\{\frac{\exp\left\{\frac{-(|\varphi|-m_i)^2}{2\sigma_i^2}\right\}+\exp\left\{\frac{-(|\varphi|+m_i)^2}{2\sigma_i^2}\right\}}{\sqrt{2\pi\sigma_i^2}}\prod_{j=1,j\neq i}^{J}\left[Q\left(\frac{|\varphi|-m_i}{\sigma_j}\right)+Q\left(\frac{|\varphi|+m_i}{\sigma_j}\right)\right]\right. \\
\left.-\frac{-\exp\left\{\frac{-(|\varphi|-m_i)^2}{2\sigma_i^2}\right\}+\exp\left\{\frac{-(|\varphi|+m_i)^2}{2\sigma_i^2}\right\}}{\sqrt{2\pi\sigma_i^2}}\prod_{j=1,j\neq i}^{J}\left[-Q\left(\frac{|\varphi|-m_i}{\sigma_j}\right)+Q\left(\frac{|\varphi|+m_i}{\sigma_j}\right)\right]\right\}, \quad \varphi \geq 0 \\
\frac{1}{2}\sum_{i=1}^{J}\left\{\frac{\exp\left\{\frac{-(|\varphi|-m_i)^2}{2\sigma_i^2}\right\}+\exp\left\{\frac{-(|\varphi|+m_i)^2}{2\sigma_i^2}\right\}}{\sqrt{2\pi\sigma_i^2}}\prod_{j=1,j\neq i}^{J}\left[Q\left(\frac{|\varphi|-m_i}{\sigma_j}\right)+Q\left(\frac{|\varphi|+m_i}{\sigma_j}\right)\right]\right. \\
\left.+\frac{-\exp\left\{\frac{-(|\varphi|-m_i)^2}{2\sigma_i^2}\right\}+\exp\left\{\frac{-(|\varphi|+m_i)^2}{2\sigma_i^2}\right\}}{\sqrt{2\pi\sigma_i^2}}\prod_{j=1,j\neq i}^{J}\left[-Q\left(\frac{|\varphi|-m_i}{\sigma_j}\right)+Q\left(\frac{|\varphi|+m_i}{\sigma_j}\right)\right]\right\}, \quad \varphi \leq 0
\end{cases} \tag{4.50}
$$

Les Figures 4.5 et 4.6 montrent deux comparaisons entre la fréquence relative de ψ obtenue par simulation et la fonction de densité de probabilité de ψ dans les deux cas où $J = 2$ et $J = 3$. Ces résultats sont obtenus en supposant que chaque entrée z_i de l'opérateur addmin a une moyenne et une variance unitaire. La grande similitude qu'il y a entre les résultats de simulation et ceux obtenus théoriquement permet de valider l'équation (4.50).

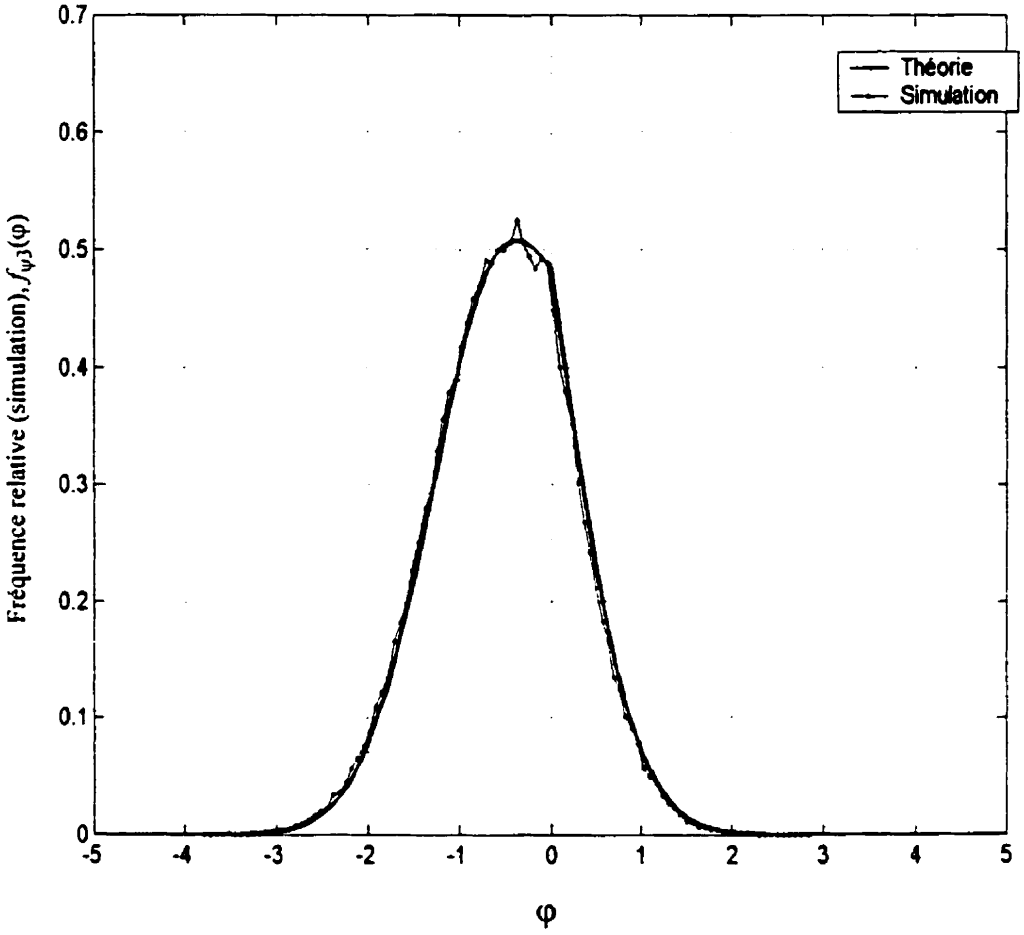


Figure 4.5 Comparaisons entre les résultats de simulation et la fonction de densité de probabilité donnée par (4.50) pour J=2, où la moyenne et la variance des variables z_i sont égales à 1

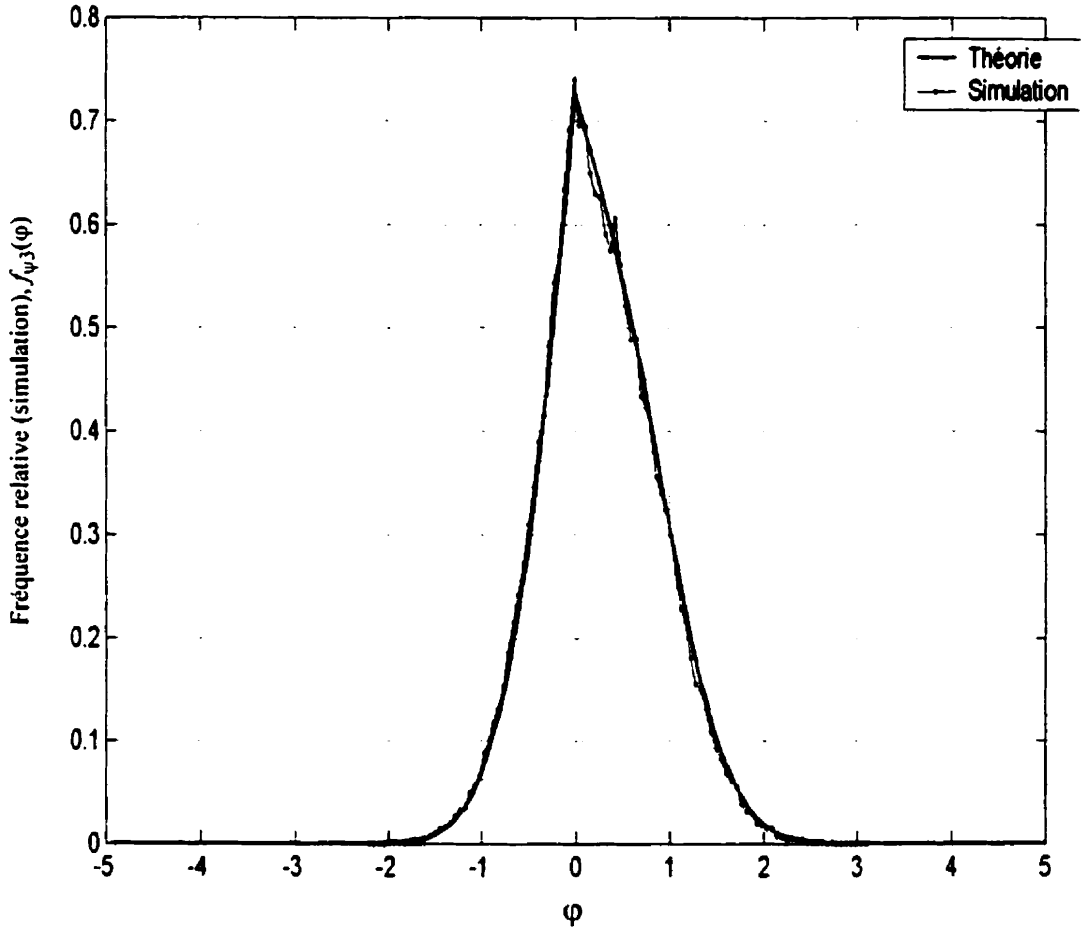


Figure 4.6 Comparaisons entre les résultats de simulation et la fonction de densité de probabilité donnée par (4.50) pour J=3, où la moyenne et la variance des variables z_i sont égales à 1

4.1.3 Evaluation de la probabilité d'erreur du décodage à seuil pour des CSOC

Tout comme l'indique (4.9), afin d'évaluer la probabilité d'erreur par bit que procure le décodage à seuil d'un code CSOC, il est nécessaire de déterminer la valeur moyenne $\bar{\lambda}$ et la variance σ_{λ}^2 en utilisant les deux relations (4.7) et (4.8). La moyenne $\bar{\lambda}$ dépend des J

valeurs moyennes $\bar{\psi}_j$, j=1, ..., J, lesquelles s'obtiennent comme suit :

$$\bar{\psi}_j = E\left\{\psi_j \middle| x^u = 1\right\} = \int_{-\infty}^{\infty} \varphi_j f_{\psi_j|x^u}(\varphi|x^u = 1)d\varphi_j \tag{4.51}$$

La variance σ_λ^2 est donnée par la somme des variances de chaque variable aléatoire ψ_j, j=1, ..., J. On calcule donc les variances $\sigma_{\psi_j}^2$, j=1, ..., J, en évaluant l'intégrale suivante :

$$\sigma_{\psi_j}^2 = E\left\{(\psi_j - \bar{\psi}_j)^2 \middle| x^u = 1\right\} = \int_{-\infty}^{\infty} \varphi_j^2 f_{\psi_j|x^u}(\varphi|x^u = 1)d\varphi_j - |\bar{\psi}_j|^2 \tag{4.52}$$

Dans (4.51) et (4.52), la fonction $f_{\psi_j|x^u}(\varphi|x^u = 1)$ représente la fonction de densité de probabilité conditionnelle à $x^u = 1$ de la variable aléatoire ψ_j résultant du $j^{ième}$ opérateur addmin ayant un certain nombre n d'entrées. Chacune de ces n entrées est représentée par une variable aléatoire gaussienne z_i de moyenne m_i et de variance σ_i^2. La fonction $f_{\psi_j|x^u}(\varphi|x^u = 1)$ se déduit directement de (4.50) en posant que les variables aléatoires z_i correspondantes aux symboles d'information reçus, y^u, sont toutes de moyenne $m_i = 1$ et de variance $\sigma_i^2 = N_0/2$. D'autre part, pour un code CSOC de J connexions, les variables aléatoires z_i qui correspondent aux symboles de parité reçus, y^p, ont une moyenne $m_i = (-1)^{J+1}$ et une variance $\sigma_i^2 = N_0/2$. Finalement, en se référant à l'équation de

décodage donnée par (2.43), d'autres entrées du $j^{ième}$ opérateur addmin peuvent provenir de la rétroaction de la décision. Il peut devenir difficile d'évaluer directement la moyenne et la variance des variables aléatoires z_i correspondantes aux termes de la rétroaction de la décision car, elles doivent être égales aux valeurs $\bar{\lambda}$ et σ_{λ}^2 respectivement. Pour les codes convolutionnels orthogonaux ayant un nombre de connexions J élevé, l'approche utilisée dans [42] qui consiste à considérer le décodeur avec rétroaction de la décision comme une machine séquentielle stochastique ne peut être appliquée, le nombre possible d'états du décodeur étant très grand. Pour résoudre ce problème, deux autres approches sont considérées.

La première approche consiste à utiliser une procédure de décodage modifiée dans laquelle les termes de la rétroaction de la décision $\lambda_{i+\alpha_j-\alpha_k}$, $(\alpha_j-\alpha_k)<0$ sont remplacés par les symboles d'information reçus $y^u_{i+\alpha_j-\alpha_k}$ où $(\alpha_j-\alpha_k)<0$. Cette procédure de décodage connue sous le nom de décodage défini [44] se décrit par l'équation de décodage suivante :

$$\lambda_i = y_i^u + \sum_{j=1}^{J}\left(y^p_{i+\alpha_j} \Diamond \sum_{k=1}^{j-1}\Diamond\, y^u_{i+\alpha_j-\alpha_k} \Diamond \sum_{k=j+1}^{J}\Diamond\, y^u_{i+\alpha_j-\alpha_k}\right) \tag{4.53}$$

Même si le décodage à seuil défini procure de moins bonnes performances d'erreur que le décodage à seuil avec rétroaction de la décision [42] [44], son analyse des performances d'erreur conduit à l'obtention d'une borne supérieure sur la probabilité d'erreur du décodage à seuil avec rétroaction de la décision. En considérant une procédure

de décodage défini, les variables aléatoires z_i qui correspondent aux symboles d'information $y^{u}_{i+\alpha_j-\alpha_k}$, où $(\alpha_j-\alpha_k)<0$, ont une moyenne m_i = 1 et une variance $\sigma_i^2=N_0/2$.

La seconde approche consiste à utiliser une procédure itérative de calcul pour déterminer les valeurs $\bar{\lambda}$ et σ^2_{λ} lorsqu'un décodage à seuil avec rétroaction de la décision est considéré. Cette procédure de calcul débute à la première étape en posant les valeurs initiales $\bar{\lambda}^{(0)}=\bar{\lambda}_{def}$ et $\sigma^2_{\lambda^{(0)}}=\sigma^2_{\lambda_{def}}$ où $\bar{\lambda}_{def}$ et $\sigma^2_{\lambda_{def}}$ sont obtenues selon la première approche alors qu'un décodage à seuil défini était utilisé. Par la suite, la moyenne et la variance des entrées des opérateurs addmin correspondantes aux termes provenant de la rétroaction de la décision, $\lambda_{i+\alpha_j-\alpha_k}$, $(\alpha_j-\alpha_k)<0$, prennent les valeurs $\bar{\lambda}_{def}$ et $\sigma^2_{\lambda_{def}}$ respectivement. À chaque étape k=1, 2, ..., de la procédure de calcul, les valeurs de $\bar{\lambda}^{(k)}$ et $\sigma^2_{\lambda^{(k)}}$ sont mises à jour en utilisant (4.51), (4.52) et finalement (4.9). Cette procédure de calcul est appliquée jusqu'à ce que la différence $\left|\frac{\bar{\lambda}^{(k)}}{\sigma_{\lambda^{(k)}}}-\frac{\bar{\lambda}^{(k-1)}}{\sigma_{\lambda^{(k-1)}}}\right|\le\varepsilon$ où ε doit être ajusté à une valeur suffisamment faible pour que le calcul soit fait avec une précision satisfaisante.

Il importe de noter que les intégrales que comportent les deux équations (4.51) et (4.52) sont évaluées numériquement. Les deux Figures 4.7 et 4.8 montrent des comparaisons entre des résultats de simulation et les résultats obtenus selon les deux approches considérées dans cette section. La Figure 4.7, illustre cette comparaison pour un

code CSOC, J=6. Comme mentionné auparavant, l'utilisation de la première approche qui fait appel à un décodage défini procure une borne supérieure qui a tendance à se reserrer lorsque E_b/N_o décroit. La grande similitude entre les résultats de simulation et ceux obtenus par le calcul utilisant la deuxième approche où le vrai décodeur à seuil avec rétroaction de la décision est considéré, confirme la validité de cette méthode. Comme le montre la Figure 4.8 où un CSOC, J=10 est utilisé, à de faibles rapports signal à bruit, E_b/N_o, la probabilité d'erreur calculée selon la deuxième approche devient supérieure à celle calculée selon la première approche. Ce dernier résultat est en contradiction avec l'hypothèse qui stipule que le décodage à seuil défini procure des performances d'erreur pires que celle du décodage à seuil avec rétroaction de la décision. Cela signifie que pour de grandes valeurs de J, à de faibles rapports signal à bruit, E_b/N_o et sous l'effet de la rétroaction de la décision, la procédure de calcul itérative utilisée ici ne semble pas valide. Cependant, la cause exacte de la divergence de cette procédure de calcul demeure une question ouverte qui pourra être traitée lors de travaux futurs.

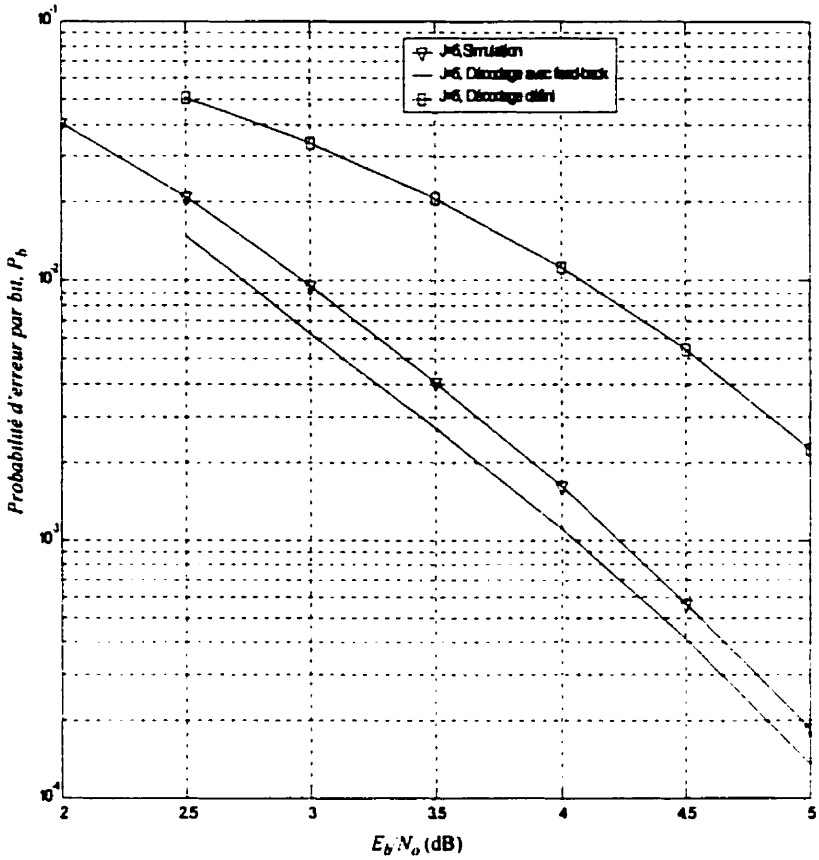


Figure 4.7 Comparaisons entre les résultats de simulation et les deux méthodes de calcul pour un code CSOC, J=6

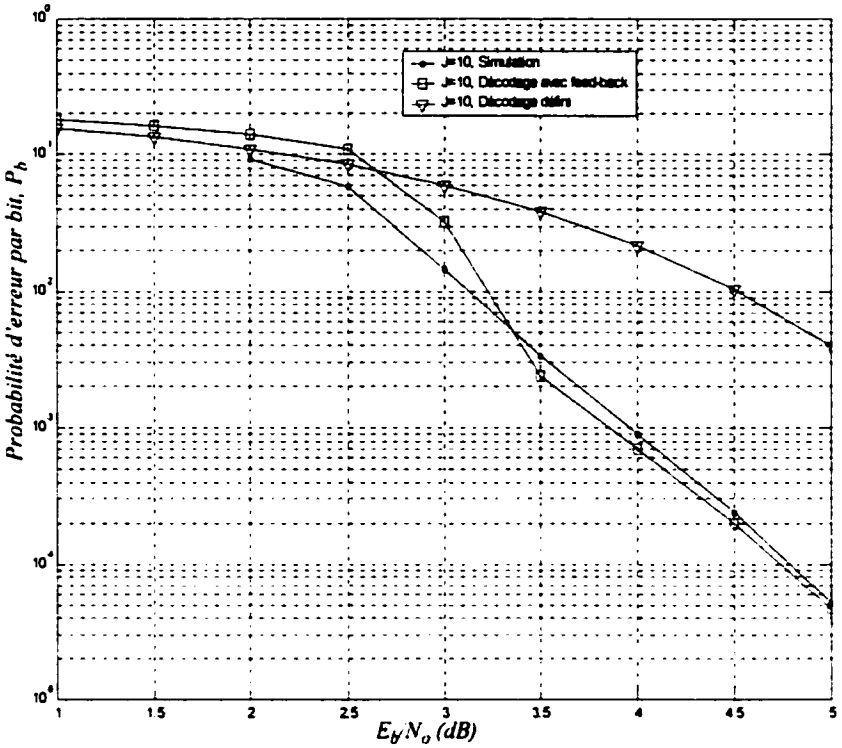


Figure 4.8 Comparaisons entre les résultats de simulation et les deux méthodes de calcul pour un code CSOC, J=10

4.2 Extension : Évaluation de la probabilité d'erreur pour *M* itérations de décodage

Dans cette section, on y présente une extension de l'analyse précédente dans le but d'évaluer les performances d'erreur du processus itératif de décodage à seuil. Les codes utilisés dans ce cas sont convolutionnels et multiplement orthogonaux au sens strict. Cette analyse est particulièrement utile pour prédire le point de convergence de la probabilité d'erreur que procure le processus itératif de décodage à seuil utilisant des codes

convolutionnels doublement orthogonaux définis au sens strict, CSO^{2}C-SS. La méthode d'analyse proposée ici, repose sur le principe suivant. Il s'agit principalement d'appliquer de manière récursive, la méthode développée plutôt à la section 4.1. Nous utilisons alors comme hypothèse, qu'à l'itération courante, μ, toutes les entrées identifiées par $\lambda_{i+\alpha_j-\alpha_k}^{(\mu-1)}$ dans (3.1) et qui proviennent de l'itération précédente (μ-1) sont distribuées selon une fonction gaussienne de moyenne $\bar{\lambda}^{(\mu-1)}$ et de variance $\sigma^2_{\lambda^{(\mu-1)}}$. L'équation (4.9) initialement prévue pour une seule itération devient alors pour l'itération (μ) :

$$P_b^{(\mu)}(E) \cong \frac{1}{\sqrt{2\pi\sigma^2_{\lambda^{(\mu)}}}} \int_{-\infty}^{0} \exp\left\{-\frac{(\lambda^{(\mu)}-\bar{\lambda}^{(\mu)})^2}{2\sigma^2_{\lambda^{(\mu)}}}\right\} d\lambda^{(\mu)} = Q\left(\frac{\bar{\lambda}^{(\mu)}}{\sigma_{\lambda^{(\mu)}}}\right) \qquad (4.54)$$

où $\bar{\lambda}^{(\mu)}$ et $\sigma^2_{\lambda^{(\mu)}}$ sont évaluées en utilisant (4.7) et (4.8). L'analyse présentée ici, est uniquement basée sur la deuxième approche exposée à la section précédente. L'effet de la rétroaction de la décision est donc considéré dans l'évaluation de $\bar{\lambda}^{(\mu)}$ et $\sigma^2_{\lambda^{(\mu)}}$ en utilisant une procédure itérative de calcul.

4.2.1 Comparaison des résultats analytiques et de simulation

Une comparaison des résultats de cette analyse obtenus pour un code CSO^{M}C-SS, J=5, avec des résultats de simulation pour un code CSO^{2}C-SS, J=5 est montrée à la Figure 4.9 où six itérations de décodage ont été effectuées. Les résultats de simulation concordent bien avec ceux obtenus par l'approche analytique. En plus de valider l'analyse, ces

résultats démontrent que la double orthogonalité du code convolutionnel est suffisante afin d'obtenir de bonnes performances d'erreur. En effet, comme le montre la Figure 4.9, pour de forts rapports signal à bruit, les dernières itérations des deux codes CSO^{M}C-SS et CSO^{2}C-SS procurent sensiblement les mêmes performances d'erreur.

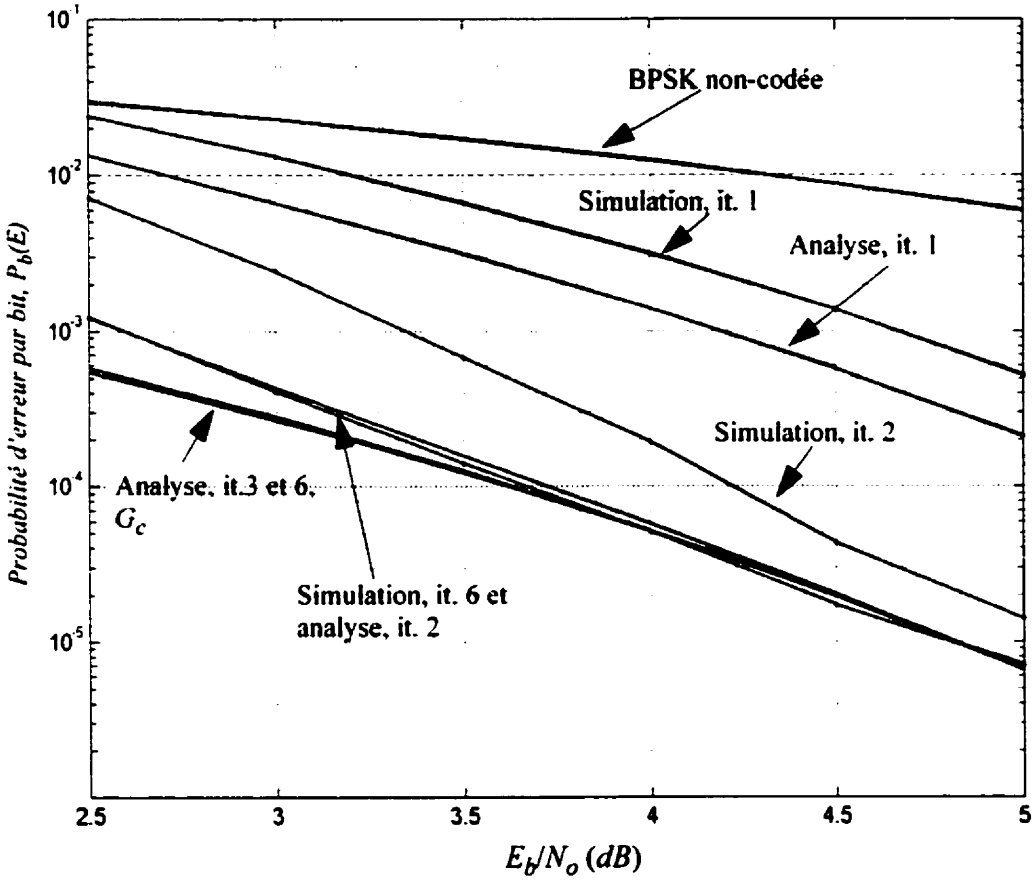


Figure 4.9 Comparaison entre les résultats analytiques obtenus pour un CSO^{M}C-SS, J=5 et les résultats de simulation pour un code CSO^{2}C-SS, J=5

Sur cette même figure, est aussi illustré le gain asymptotique de codage, G_c, d'un code convolutionnel orthogonal produisant J équations de contrôle de parité. Le gain asymptotique de codage, G_c, d'un code CSOC s'obtient de la façon suivante [2] :

$$G_c = 10\log_{10}(r_c d_{min}) = 10\log_{10}(r_c(J+1)) \quad (4.55)$$

Pour un taux de codage $r_c = 1/2$ et J=5, on trouve $G_c = 4.77$ dB. Les résultats montrés à la Figure 4.9 indiquent qu'avec un code CSO^{M}C-SS et un grand nombre d'itérations, les performances d'erreur peuvent s'approcher très près de la limite du code donnée par le gain asymptotique de codage. Ces résultats indiquent aussi que l'orthogonalité multiple permet au code de converger plus rapidement que lorsque l'orthogonalité n'est seulement que d'ordre 2.

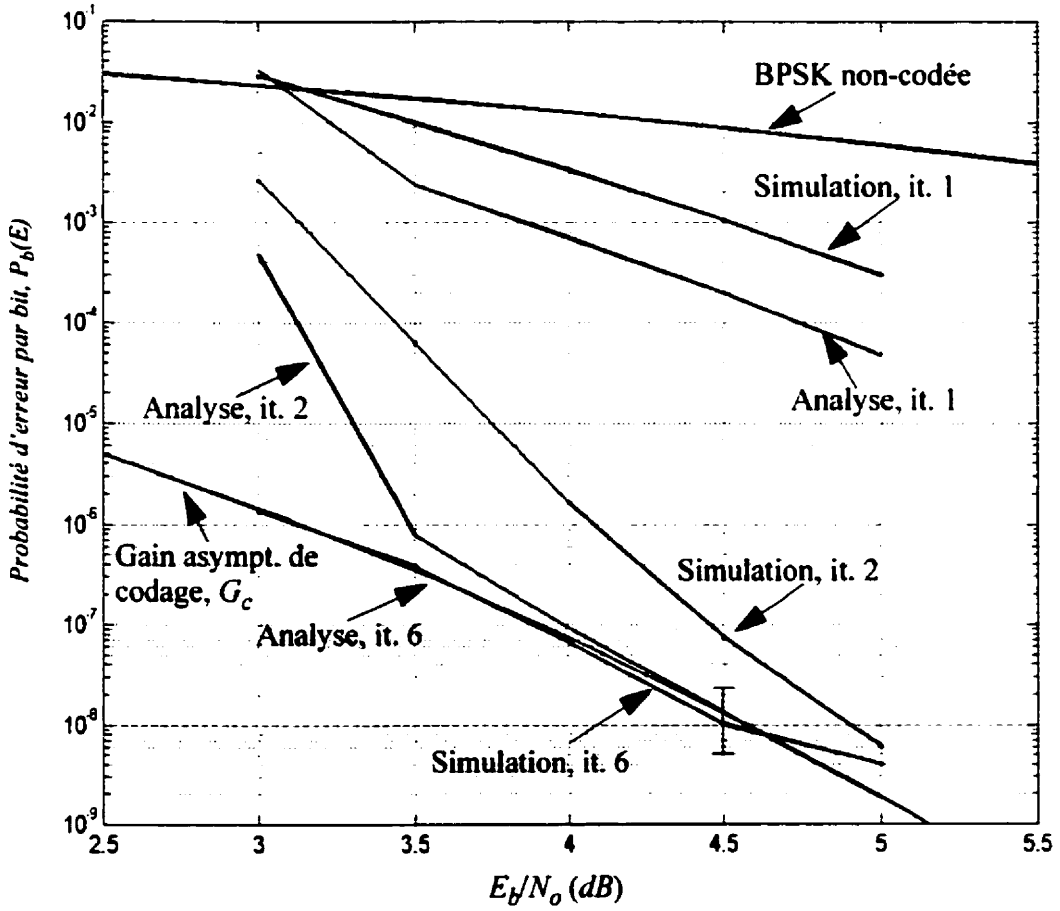


Figure 4.10 Comparaison entre les résultats analytiques obtenus pour un CSO^{M}C-SS, J=10 et les résultats de simulation pour un code CSO^{2}C-SS, J=10

Des résultats similaires sont obtenus pour J=10 et sont montrés à la Figure 4.10. Sur cette figure, le gain asymptotique de codage G_c est de nouveau tracé pour un code orthogonal J=10. On remarque que les comportements des codes multiplement orthogonaux et doublement orthogonaux observés pour une valeur de J faible se retrouvent aussi pour des valeurs de J plus grandes.

4.3 Conclusion

Une méthode permettant d'analyser les performances d'erreur du processus itératif de décodage à seuil a été présentée. L'utilisation des codes CSO^{M}C-SS permet de simplifier l'analyse. Grâce à l'orthogonalité d'ordre M définie au sens strict, les observables utilisés dans le processus de décodage itératif sont indépendants à chaque itération. Cette analyse repose essentiellement sur l'hypothèse simplificatrice que la valeur approximative du MAP déterminée à la sortie de chaque étape de décodage se distribue selon une gaussienne. Cette hypothèse a été vérifiée expérimentalement à l'aide de simulations. La fonction de densité de probabilité d'une variable aléatoire qui résulte d'une opération addmin effectuée sur un ensemble de variables aléatoires est nécessaire à l'analyse des performances d'erreur du décodage à seuil. Cette fonction a donc été développée en faisant appel à des fonctions d'énumération.

Compte tenu des résultats présentés dans ce chapitre, on a pu montrer que pour de fortes valeurs du rapport signal à bruit, l'orthogonalité d'ordre deux défini au sens strict suffit pour atteindre la limite du code donnée par le gain asymptotique de codage.

CHAPITRE 5: CODES CONVOLUTIONNELS DOUBLEMENT ORTHOGONAUX RÉCURSIFS (R-CSO^{2}C)

La technique de codage et de décodage développée jusqu'à présent dans cette thèse utilise des codes non-récursifs et systématiques doublement orthogonaux car, ils sont appropriés au décodage à seuil et ils peuvent être décodés par une procédure itérative. Au chapitre précédent, il a été démontré qu'une définition au sens strict de l'orthogonalité d'ordre deux suffit pour qu'un code atteigne sa limite de performance d'erreur dictée par le gain asymptotique de codage. Pour un code CSO^{2}C-SS produisant J équations de contrôle de parité, le processus de décodage ne permet pas d'améliorer, à chaque itération, la fiabilité de chaque symbole de parité reçu. Par conséquent, les meilleures performances d'erreur qu'un tel code puisse atteindre sont limitées par la valeur du gain asymptotique de codage donnée par $10log_{10}(r_c d_{min})$ où $d_{min}=(J+1)$. La faible complexité jumelée aux bonnes performances d'erreur que procure une telle technique de correction d'erreurs constitue une motivation à approfondir davantage les variantes de cette méthode de codage et de décodage correcteur d'erreurs.

Des codes en blocs doublement orthogonaux définis au sens strict (en anglais, Block self-doubly orthogonal codes in strict sense, BSO^{2}C-SS), ayant un taux de codage $J/(K+J)$, $K=1, 2, ...,J$, peuvent être aussi obtenus en utilisant la même approche décrite dans [21] et [22]. Comme il est mentionné dans [38] et [21], ces codes BSO^{2}C-SS sont équivalents aux codes à faible densité de parité (en anglais, low-density parity-check

codes LDPC), [20]. Le caractère quasi-cyclique des codes BSO^{2}C-SS permet la construction de codes CSO^{2}C-SS de taux de codage $r_c = J/(J+K)$. Le processus itératif de décodage proposé par Gallager [20] permettant de décoder les codes LDPC peut donc être utilisé pour le décodage des codes CSO^{2}C-SS. Une analyse du processus itératif de décodage de Gallager montre que la fiabilité de chaque symbole de parité reçu peut être améliorée à chaque itération du processus de décodage, si le code CSO^{2}C-SS est récursif. Par conséquent, il devrait en découler que l'utilisation d'un code CSO^{2}C-SS récursif, devrait conduire à une amélioration des performances d'erreur comparées à celles obtenues avec un code CSO^{2}C-SS non-récursif. Cependant, comment appliquer un décodage itératif basé sur la probabilité *a posteriori* (PAP) à un code systématique récursif et ainsi obtenir une amélioration des performances d'erreur ? Le présent chapitre répond à cette question en faisant appel à la notion de code dual et aux techniques de codage et décodage introduites par Gallager [20].

Un bref rappel concernant les codes en blocs à faible densité de parité (Low Density Parity Check), LDPC est tout d'abord présenté à la section 5.1. Dans cette même section, un parallèle est aussi fait entre les codes LDPC et les codes doublement orthogonaux. À la section 5.2, la technique de codage utilisant des codes convolutionnels orthogonaux récursifs, R-CSOC, est exposée. Un algorithme de décodage pour ces codes est par la suite développé. Les définitions de codes convolutionnels doublement orthogonaux récursifs au sens large R-CSO^{2}C-WS et au sens strict, R-CSO^{2}C-SS sont alors obtenues. Finalement, basées sur des résultats de simulation, des conclusions sont présentées à la section 5.3.

5.1 Équivalence entre les codes à faible densité de parité et les codes doublement orthogonaux

5.1.1 Codes en blocs à faible densité de parité

Un code bloc à faible densité de parité (Low Density Parity Check, LDPC) dénoté par (n, j, k) est spécifié par une matrice de contrôle de parité de longueur n correspondant aussi à la longueur du code. Dans cette matrice, il n'y a que j éléments de valeur 1 par colonne et $k > j$ éléments de valeur 1 par ligne. Le code bloc obtenu avec cette structure a un taux de codage $(n - nj/k)/n = 1 - j/k$, c'est-à-dire que la dimension du bloc de symboles d'information présent à l'entrée du codeur est $n(1-j/k)$ alors que la dimension du bloc de symboles codés est n. Cette matrice de contrôle de parité se subdivise en j sous-matrices de mêmes dimensions $n/k \times n$. Chaque colonne d'une sous-matrice ne doit contenir qu'un seul élément de valeur 1. Un exemple d'une telle matrice pour un code (20, 3, 4) est donné au tableau ci-dessous [20].

1 1 1 1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
0 0 0 0	1 1 1 1	0 0 0 0	0 0 0 0	0 0 0 0
0 0 0 0	0 0 0 0	1 1 1 1	0 0 0 0	0 0 0 0
0 0 0 0	0 0 0 0	0 0 0 0	1 1 1 1	0 0 0 0
0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	1 1 1 1
1 0 0 0	1 0 0 0	1 0 0 0	1 0 0 0	0 0 0 0
0 1 0 0	0 1 0 0	0 1 0 0	0 0 0 0	1 0 0 0
0 0 1 0	0 0 1 0	0 0 0 0	0 1 0 0	0 1 0 0
0 0 0 1	0 0 0 0	0 0 1 0	0 0 1 0	0 0 1 0
0 0 0 0	0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1
1 0 0 0	0 1 0 0	0 0 0 1	0 0 0 0	0 1 0 0
0 1 0 0	0 0 1 0	0 0 1 0	0 0 0 1	0 0 0 0
0 0 1 0	0 0 0 1	0 0 0 0	1 0 0 0	0 0 1 0
0 0 0 1	0 0 0 0	1 0 0 0	0 1 0 0	1 0 0 0
0 0 0 0	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1

Les deux méthodes de décodage présentées dans [20] utilisent essentiellement des procédures itératives de décodage. La première méthode de décodage constitue un décodage symbole par symbole en quantification ferme. Ce décodage qui est semblable à celui expliqué à la section 2.1.2 consiste à former, pour chaque symbole codé reçu, toutes les équations de contrôle de parité. La valeur binaire d'un symbole codé reçu est modifiée si plus d'un certain nombre fixe de ces équations de contrôle de parité ne sont pas satisfaites. En utilisant ces valeurs modifiées des symboles codés, toutes les équations de contrôle de parité sont recalculées. Cette procédure itérative est répétée jusqu'à ce que toutes les équations de contrôle de parité soient satisfaites.

La deuxième approche de décodage proposée par Gallager [20] consiste en une procédure itérative de décodage à sortie pondérée. Elle est aussi obtenue par l'extension de la méthode de décodage en quantification ferme. Cette procédure est donc semblable à

celle décrite au chapitre 3 avec la seule différence que les équations de contrôle de parité qui contiennent le symbole codé courant à décoder ne sont pas inclues. Quoique cette procédure de décodage soit appliquée à des codes en blocs, elle est aussi fortement similaire à celle décrite à la section 3.3.2 où la condition particulière $l \neq k$ dans (3.4) était imposée. Rappelons que cette condition sert à exclure l'équation de contrôle de parité qui introduit des répétitions de certains symboles de parité et d'information.

Comme nous l'avons vu aux chapitres 2 et 3, une telle procédure itérative nécessite que les équations de contrôle de parité excluant le symbole à décoder soient indépendantes. La matrice de contrôle de parité du code (20, 3, 4) de l'exemple précédent, comporte les propriétés nécessaires pour l'application d'une telle procédure de décodage sur au moins deux itérations. Un ensemble de symboles codés jumelé à un ensemble d'équations de contrôle de parité peut donc servir au décodage d'un autre symbole codé qui n'est pas directement relié à ces ensembles de symboles codés et d'équations de contrôle de parité [20]. C'est aussi ce que fait en partie le décodage itératif des codes convolutionnels doublement orthogonaux.

Il importe aussi de noter que l'algorithme proposé par Gallager dans [20] pour le décodage des codes LDPC s'apparente à un autre algorithme appelé «*belief propagation*», BP [45], [46], [48], [49], [50]. L'algorithme «*belief propagation*» a été introduit pour la première fois par Mackay et Neal dans [45] comme algorithme de décodage. Cette approche a été aussi exploitée par F. Kschischang *et al.* dans [49] alors qu'ils appliquent l'algorithme BP au décodage Turbo conventionnel et au décodage des codes LDPC. Cependant, l'étude de l'algorithme «*belief propagation*» et son application au décodage

Turbo et au décodage des codes LDPC dépassent le cadre de cette thèse. De plus, les développements des algorithmes présentés dans ce chapitre sont obtenus de manière indépendante au développement de l'algorithme «*belief propagation*». Cependant, le lecteur trouvera à l'Annexe VIII un résumé de l'algorithme de Pearl ou BP ainsi qu'une brève description de l'algorithme de Gallager.

Plusieurs résultats expérimentaux dont ceux de MacKay [50] ont montré que les performances d'erreur des codes LDPC s'approchent de la limite prédite par Shannon [1]. Par exemple, avec un code LDPC de longueur $n = 10^6$ et de taux de codage 1/2, les résultats de simulation présentés dans [51] indiquent qu'une probabilité d'erreur par bit de 10^{-6} est possible avec un rapport signal à bruit se situant à 0.13 dB de la capacité.

5.1.2 Codes en blocs doublement orthogonaux

Des codes en blocs doublement orthogonaux peuvent être aussi obtenus par extension des codes convolutionnels doublement orthogonaux en effectuant une opération modulo un entier f sur l'indice temporel i dans les équations qui décrivent le codage et le décodage des codes convolutionnels doublement orthogonaux [38]. De cette façon, il a été démontré dans [38] et [21] que ces codes en blocs doublement orthogonaux, lorsque définis au sens strict présentent aussi les mêmes caractéristiques d'indépendance que celles des codes LDPC. La procédure de génération des codes convolutionnels doublement orthogonaux au sens strict a donc pu servir à la génération de nouveaux codes en blocs à faible densité de parité de parité ou LDPC. Les méthodes de recherche de ces codes et les résultats obtenus sont présentés dans [21], [22]. Leur taux de codage est de la forme $K/(K+J)$, où $K = 1, ..., J$

et où J représente, comme auparavant, le nombre d'équations de contrôle de parité orthogonales au symbole à décoder. De plus, leur propriété cyclique permet aussi de les considérer comme des codes convolutionnels doublement orthogonaux au sens strict de taux de codage $K/(K+J)$. Comme nous le verrons à la section suivante, les principes de codage et décodage LDPC établis par Gallager ainsi que l'extension aux codes doublement orthogonaux au sens strict de taux de codage $K/(K+J)$ seront très utiles pour définir des codes convolutionnels doublement orthogonaux récursifs au sens strict.

5.2 Définition et décodage des codes R-CSO^{2}C

La définition des codes convolutionnels doublement orthogonaux récursifs, R-CSO^{2}C, fait intervenir la notion de code dual [2]. Pour chaque code linéaire C généré par une matrice de génératrice $\boldsymbol{G}$, il existe une matrice $\boldsymbol{H}$ telle que le produit $\boldsymbol{G} \cdot \boldsymbol{H}^T = 0$. Cette relation provient du fait que chaque ligne de la matrice $\boldsymbol{G}$ est orthogonale à chaque ligne de la matrice $\boldsymbol{H}$. Cette matrice $\boldsymbol{H}$ que l'on appelle matrice de contrôle de parité a été particulièrement utile au chapitre 2 où la technique de décodage à seuil a été présentée. Les combinaisons linéaires des lignes de la matrice $\boldsymbol{H}$ forment un autre code dénoté $C^{\perp}$ que l'on appelle code dual du code C généré par une matrice $\boldsymbol{G}$. Ainsi, pour tous mots de codes $\boldsymbol{v} \in C$ et $\boldsymbol{w} \in C^{\perp}$, on a que $\boldsymbol{v} \cdot \boldsymbol{w}^T = 0$. La matrice de contrôle de parité $\boldsymbol{H}$ d'un code linéaire C devient alors une matrice génératrice du code dual $C^{\perp}$.

Pour un code convolutionnel de taux de codage b/v, la matrice génératrice $\boldsymbol{G}$ peut s'écrire de manière compacte en utilisant une matrice de connexions $\boldsymbol{G}(D)$ dont l'élément

à la position (k, l), $k = 1, 2 \ldots, b$ et $l = 1, 2 \ldots, v$ est un polynôme $g_{k,l}(D)$. Un codeur convolutionnel non-systématique peut toujours être transformé sous une forme systématique par de simples opérations sur les lignes de la matrice d'origine. Même si les deux codeurs sont différents, ils génèrent le même code. La réalisation d'un codeur convolutionnel systématique obtenue à partir d'un codeur convolutionnel non-systématique impose des connexions de retour sur les registres à décalage du codeur systématique. Dans ce cas, le codeur convolutionnel est dit systématique et récursif. Par exemple, supposons un code convolutionnel non-systématique de taux de codage $r_c = 1/2$ généré par une matrice $\boldsymbol{G}(D)$ donnée par :

$$\boldsymbol{G}(D) = \left[g_1(D)\ g_2(D)\right] \tag{5.1}$$

Ce codeur peut être converti en un codeur convolutionnel systématique et récursif de la manière suivante :

$$\boldsymbol{G}^*(D) = \left[1 \quad \frac{g_2(D)}{g_1(D)}\right] \tag{5.2}$$

La matrice de contrôle de parité $\boldsymbol{H}(D)$ correspondante à $\boldsymbol{G}^*(D)$ est telle que $\boldsymbol{G}^*(D) \cdot \boldsymbol{H}^T(D) = 0$. Cette matrice $\boldsymbol{H}(D)$ peut alors s'écrire comme suit :

$$\boldsymbol{H}(D) = \left[g_2(D)\ g_1(D)\right] \tag{5.3}$$

Cette matrice $\boldsymbol{H}(D)$ génère aussi le code dual $C^{\perp}$ du code C généré par $\boldsymbol{G}^*(D)$. Si le code généré par $\boldsymbol{G}^*(D)$ est récursif, le code dual $C^{\perp}$ généré par $\boldsymbol{H}(D)$ peut être non-récursif.

Cette remarque devient particulièrement intéressante pour obtenir un décodage PAP des codes récursifs et systématiques. En effet, l'algorithme de décodage PAP impose comme restriction, que le code soit systématique et non-récursif. Le code dual $C^{\perp}$ généré par $\boldsymbol{H}(D)$ étant non-récursif, un algorithme de décodage permettant de décoder un code convolutionnel orthogonal récursif peut alors être facilement déduit à partir de l'algorithme de décodage à seuil ou PAP. Dans les sections qui suivent, cette notion de code dual servira à définir des codes convolutionnels récursifs doublement orthogonaux au sens large, R-CSO^{2}C-WS et par la suite ceux au sens strict, R-CSO^{2}C-SS. La définition de ces codes doit être telle qu'un décodage itératif PAP soit applicable, c'est-à-dire que ces codes doivent aussi obéir à certaines propriétés d'orthogonalité. Nous utiliserons donc la même procédure qui a mené à la définition, au chapitre 2, des codes convolutionnels orthogonaux et au chapitre 3, des codes convolutionnels doublement orthogonaux.

5.2.1 Codes convolutionnels orthogonaux (d'ordre 1) récursifs (R-CSOC)

La matrice de connexions $\boldsymbol{G}(D)$ d'un codeur convolutionnel orthogonal récursif de taux de codage r_c=1/2 est de la même forme que (5.2). Par conséquent, la matrice de contrôle de parité sans rétroaction $\boldsymbol{H}(D)$ correspondante est donnée par (5.3). Tout d'abord, dénotons par $J^{(1)}$, le nombre de coefficients non-nuls de $g_1(D)$ et par $J^{(2)}$, le nombre de coefficients non-nuls du polynôme $g_2(D)$. Le générateur $g_2(D)$ est alors décrit par les $J^{(2)}$ exposants β_j de D qui correspondent aux coefficients non-nuls de $g_2(D)$, c'est-à-dire :

$$g_2(D) = \sum_{j=1}^{J^{(2)}} \oplus D^{\beta_j} \tag{5.4}$$

Cette définition de $g_2(D)$ est équivalente à la définition de l'ensemble des J valeurs entières α_j qui spécifient un codeur CSOC. De la même manière, le générateur $g_1(D)$ représentant la partie récursive du codeur est décrit par les $J^{(1)}$ exposants ε_j de D qui correspondent aux coefficients non-nuls de $g_1(D)$:

$$g_1(D) = \sum_{j=1}^{J^{(1)}} \oplus D^{\varepsilon_j} \tag{5.5}$$

Avec ces deux définitions, (5.4) et (5.5), un code R-CSOC est complètement spécifié par les deux ensembles des positions de connexions $\{\beta_j\}$ et $\{\varepsilon_j\}$. Sans perte de généralité, on pose que $\beta_1=\varepsilon_1=0$. De plus, pour s'assurer de conserver la causalité du codeur [43], il faut que $\varepsilon_{J^{(1)}} \geq \beta_{J^{(2)}}$. La séquence $U(D)$, de symboles d'information s'écrit sous forme polynomiale comme suit:

$$U(D) = u_o + u_1 D + u_2 D^2 + \ldots \tag{5.6}$$

où u_i représente le symbole d'information à l'instant i. Le produit $U(D)G(D)$ forme la sortie du codeur $C(D) = [U(D)\ P(D)]$ où $P(D)$ représente la séquence de symboles de parité qui exprimée sous forme polynomiale donne :

$$P(D) = p_o + p_1 D + p_2 D^2 + \ldots \tag{5.7}$$

où p_i représente le symbole de parité à l'instant i. En suivant cette procédure de codage, le symbole de parité, p_i, s'écrit comme suit :

$$p_i = \sum_{j=1}^{J^{(2)}} \oplus u_{i-\beta_j} \oplus \sum_{k=2}^{J^{(1)}} \oplus p_{i-\varepsilon_k} \tag{5.8}$$

Considérons tout d'abord le cas où la sortie du codeur, $\boldsymbol{C}(D)$, est transmise sur un canal binaire symétrique. Le vecteur $\boldsymbol{R}(D) = \left[\tilde{U}(D)\ \tilde{P}(D)\right]$ formé des séquences reçues de symboles d'information $\tilde{U}(D) = \tilde{u}_o + \tilde{u}_1 D + \tilde{u}_2 D^2 + \ldots$ et de parité $\tilde{P}(D) = \tilde{p}_o + \tilde{p}_1 D + \tilde{p}_2 D^2 + \ldots$ est présent à l'entrée du décodeur. L'élément $\tilde{u}_i = u_i \oplus e_i^u$ représente le symbole d'information reçu à l'instant i et l'élément $\tilde{p}_i = p_i \oplus e_i^p$ représente le symbole de parité reçu à l'instant i. Les symboles e_i^u et e_i^p sont les symboles d'erreur correspondants aux symboles d'information u_i et de parité p_i respectivement. À la première étape de décodage, la séquence de symboles de syndrome $S(D)$ est calculée en effectuant le produit suivant :

$$S(D) = \boldsymbol{R}(D)\boldsymbol{H}^T(D) = \left[\tilde{U}(D)\ \tilde{P}(D)\right]\begin{bmatrix} g_2(D) \\ g_1(D) \end{bmatrix}$$

$$= \tilde{U}(D)g_2(D) + \tilde{P}(D)g_1(D) \pmod 2 \tag{5.9}$$

Le symbole de syndrome à l'instant i s'écrit alors comme suit :

$$s_i = \sum_{j=1}^{J^{(2)}} \oplus \tilde{u}_{i-\beta_j} \oplus \sum_{k=1}^{J^{(1)}} \oplus \tilde{p}_{i-\varepsilon_k} \tag{5.10}$$

À l'instant $(i + \beta_l)$, $l = 1, 2, ..., J^{(2)}$, le symbole de syndrome donnée par (5.10) devient :

$$s_{i+\beta_l} = \tilde{u}_i \oplus \sum_{\substack{j=1 \\ j \neq l}}^{J^{(2)}} \oplus \tilde{u}_{i+\beta_l-\beta_j} \oplus \sum_{k=1}^{J^{(1)}} \oplus \tilde{p}_{i+\beta_l-\varepsilon_k} \tag{5.11}$$

Pour que le symbole de syndrome $s_{i+\beta_l}$ formé à la première itération du processus de décodage n'utilise que des symboles codés reçus indépendants, les deux conditions d'orthogonalité (d'ordre 1) suivantes doivent être satisfaites :

1. Les différences $\{\beta_l - \beta_j\}$, $l \neq j$, sont distinctes ;

2. Les différences $\{\beta_l - \varepsilon_k\}$ sont distinctes.

On déduit alors de (5.11) un ensemble de $J^{(2)}$ symboles de syndrome orthogonaux au symbole d'information reçu $\tilde{u}_i$. De la même manière, à l'instant $(i + \varepsilon_l)$, $l = 1, 2, ..., J^{(1)}$, le symbole de syndrome formé à la première itération devient :

$$s_{i+\varepsilon_l} = \tilde{p}_i \oplus \sum_{j=1}^{J^{(2)}} \oplus \tilde{u}_{i+\varepsilon_l-\beta_j} \oplus \sum_{\substack{k=1 \\ k \neq l}}^{J^{(1)}} \oplus \tilde{p}_{i+\varepsilon_l-\varepsilon_k} \tag{5.12}$$

Deux conditions supplémentaires sont nécessaires pour que (5.12) soit une équation de variables indépendantes. Ces deux conditions sont :

3. Les différences $\{\varepsilon_l - \varepsilon_k\}$, $l \neq k$, sont distinctes ;

4. Les différences $\{\varepsilon_l - \beta_j\}$ sont distinctes.

La condition 4 est identique à celle de 2 puisqu'ici, toutes les différences positives et négatives sont considérées. Les $J^{(1)}$ symboles de syndrome obtenus par (5.12) forment un autre ensemble de symboles de syndromes orthogonaux au symbole de parité reçu $\tilde{p}_i$. En dénotant par $\{\varepsilon_j\}\backslash\varepsilon_1$, l'ensemble $\{\varepsilon_j\}$ excluant l'élément ε_1, il est clair que les conditions 1 à 3 reviennent aussi à respecter les conditions d'orthogonalité d'ordre 1 sur l'ensemble $\{\beta_j\} \cup \{\varepsilon_j\}\backslash\varepsilon_1$. Cette dernière remarque suggère que l'on peut générer des R-CSOC de taux de codage r_c = 1/2 à partir de codes CSOC non-récursifs de taux de codage r_c = 1/2. Il suffit alors de partitionner en deux groupes distincts, $\{\beta_j\}$ et $\{\varepsilon_j\}\backslash\varepsilon_1$, l'ensemble initial $\{\alpha_j\}$ qui spécifie un code CSOC. On remarque aussi que la structure d'un code R-CSOC nécessite deux générateurs. Les codes CSOC de taux de codage $r_c = 2/3$ nécessitent aussi deux générateurs qui sont spécifiés par des ensembles disjoints $\{\beta_j\}$ et $\{\varepsilon_j\}$. Si ces deux ensembles $\{\beta_j\}$ et $\{\varepsilon_j\}$ décrivant un code CSOC, r_c = 2/3 respectent les trois premières conditions données plus haut, alors ces deux générateurs peuvent être utilisés pour former les deux générateurs d'un code R-CSOC de taux de codage r_c = 1/2. Notons que se ne sont pas tous les codes CSOC, $r_c = 2/3$ qui satisfont les trois conditions sur $\{\beta_j\}$ et $\{\varepsilon_j\}$. En effet, il suffit simplement que les deux ensembles de différences $\{\beta_l - \beta_j\}$ et $\{\varepsilon_l - \varepsilon_k\}$ soient disjoints pour qu'un code convolutionnel $r_c = 2/3$ soit orthogonal.

Tout comme il a été fait à la section 2.2.2, il est possible de modifier les équations (5.10), (5.11) et (5.12) pour obtenir des équations d'inversion. Deux ensembles d'équations d'inversion peuvent être définis. Le premier ensemble, $\left\{B^u_{l,i}\right\}$, $l = 1, 2, ..., \mathcal{J}^{(2)}$, d'équations d'inversion est orthogonal au symbole d'information u_i tandis que le deuxième ensemble d'équations d'inversion, $\left\{B^p_{l,i}\right\}$, $l = 1, 2, ..., \mathcal{J}^{(1)}$, est orthogonal au symbole de parité p_i. Ces deux ensembles d'équations s'obtiennent de la manière suivante:

$$B^u_{l,i} = s_{i+\beta_l} \oplus \tilde{u}_i = \sum_{\substack{j=1 \\ j \neq l}}^{\mathcal{J}^{(2)}} \oplus \tilde{u}_{i+\beta_l-\beta_j} \oplus \sum_{k=1}^{\mathcal{J}^{(1)}} \oplus \tilde{p}_{i+\beta_l-\varepsilon_k}$$

$$= u_i \oplus \sum_{\substack{j=1 \\ j \neq l}}^{\mathcal{J}^{(2)}} \oplus e^u_{i+\beta_l-\beta_j} \oplus \sum_{k=1}^{\mathcal{J}^{(1)}} \oplus e^p_{i+\beta_l-\varepsilon_k} \quad , l = 1, 2, ..., \mathcal{J}^{(2)} \tag{5.13}$$

$$B^p_{l,i} = s_{i+\varepsilon_l} \oplus \tilde{p}_i = \sum_{j=1}^{\mathcal{J}^{(2)}} \oplus \tilde{u}_{i+\varepsilon_l-\beta_j} \oplus \sum_{\substack{k=1 \\ k \neq l}}^{\mathcal{J}^{(1)}} \oplus \tilde{p}_{i+\varepsilon_l-\varepsilon_k}$$

$$= p_i \oplus \sum_{j=1}^{\mathcal{J}^{(2)}} \oplus e^u_{i+\varepsilon_l-\beta_j} \oplus \sum_{\substack{k=1 \\ k \neq l}}^{\mathcal{J}^{(1)}} \oplus e^p_{i+\varepsilon_l-\varepsilon_k} \quad , l = 1, 2, ..., \mathcal{J}^{(1)} \tag{5.14}$$

La procédure utilisée à la section 2.2.2 qui a mené à la définition de l'algorithme de

décodage à probabilité *a posteriori* (PAP) peut être appliquée aux deux équations (5.13) et (5.14). L'algorithme de décodage doit déterminer la valeur du bit d'information à décoder u_i en lui assignant la valeur binaire $\xi \in \{0, 1\}$ de sorte que la probabilité conditionnelle:

$$Pr\left\{u_i = \xi \middle| \left\{B^u_{l,i}\right\}\right\}, \; i = 0, 1, 2, \ldots : \xi = 0, 1 ; l = 1, 2, \ldots, J^{(2)} \tag{5.15}$$

soit maximale. D'autre part, la valeur d'un symbole de parité p_i peut être aussi déterminée à partir de l'ensemble des équations $\left\{B^p_{l,i}\right\}$ orthogonales à p_i. La maximisation de la probabilité conditionnelle suivante permet de déterminer la valeur du symbole de parité p_i:

$$Pr\left\{p_i = \xi \middle| \left\{B^p_{l,i}\right\}\right\}, \; i = 0, 1, 2, \ldots : \xi = 0, 1 ; l = 1, 2, \ldots, J^{(1)} \tag{5.16}$$

La procédure décrite jusqu'ici permet donc une correction des symboles de parité. Cette procédure s'apparente aussi à l'algorithme de décodage proposé par Gallager pour les codes LDPC. La maximisation de (5.15) et (5.16) mène à l'algorithme de décodage d'un code R-CSOC. Cet algorithme est principalement décrit par les deux équations suivantes :

$$\lambda^{(1)}_u(i) = y^u_i + \sum_{l=1}^{J^{(2)}} \left(\sum_{\substack{j=1 \\ j \neq l}}^{J^{(2)}} \Diamond \, y^u_{i+\beta_l-\beta_j} \Diamond \sum_{k=1}^{J^{(1)}} \Diamond \, y^p_{i+\beta_l-\varepsilon_k} \right) \tag{5.17}$$

$$\lambda_p^{(1)}(i) = y_i^p + \sum_{l=1}^{J^{(1)}} \left(\sum_{j=1}^{J^{(2)}} \Diamond y_{i+\varepsilon_l-\beta_j}^u \Diamond \sum_{\substack{k=1 \\ k \neq l}}^{J^{(1)}} \Diamond y_{i+\varepsilon_l-\varepsilon_k}^p \right) \tag{5.18}$$

Les deux quantités $\lambda_i^{(1)}(u)$ et $\lambda_i^{(1)}(p)$ représentent les approximations du logarithme du rapport de vraisemblance (LRV) des symboles d'information et de parité u_i et p_i respectivement, obtenues à l'itération 1. Il importe de signaler que les deux équations (5.17) et (5.18) ne font pas intervenir une rétroaction de la décision comme c'est le cas pour le décodage à seuil décrit aux chapitres précédents. Les deux équations (5.17) et (5.18) prennent donc la forme d'un décodage défini [44] et sont aussi du même type que celles qui décrivent le décodage probabiliste des codes LDPC [20].

5.2.2 Codes convolutionnels doublement orthogonaux (ordre 2) récursifs au sens large

Pour obtenir une définition des codes convolutionnels doublement orthogonaux récursifs au sens large, R-CSO^{2}C-WS, il suffit d'appliquer la même procédure utilisée à la section 3.2 qui a conduit à la définition des CSO^{2}C-WS. Ainsi, l'application récursive des deux équations (5.17) et (5.18) donne, à l'itération μ, le résultat suivant :

$$\lambda_u^{(\mu)}(i) = y_i^u + \sum_{m=1}^{J^{(2)}} \left(\sum_{\substack{n=1 \\ n \neq m}}^{J^{(2)}} \Diamond \lambda_u^{(\mu-1)}(i+\beta_m-\beta_n) \Diamond \sum_{q=1}^{J^{(1)}} \Diamond \lambda_p^{(\mu-1)}(i+\beta_m-\varepsilon_q) \right) \tag{5.19}$$

$$\lambda_p^{(\mu)}(i) = y_i^p + \sum_{m=1}^{J^{(1)}} \left(\sum_{n=1}^{J^{(2)}} \Diamond \lambda_u^{(\mu-1)}(i+\varepsilon_m-\beta_n) \sum_{\substack{q=1 \\ q\neq m}}^{J^{(1)}} \Diamond \lambda_p^{(\mu-1)}(i+\varepsilon_m-\varepsilon_q) \right) \qquad (5.20)$$

Les conditions de la double orthogonalité s'obtiennent en posant μ=2 dans (5.19) et (5.20), ce qui donne :

$$\lambda_u^{(2)}(i) = y_i^u + \sum_{m=1}^{J^{(2)}} \left\{ \sum_{n=1}^{J^{(2)}} \Diamond \left(y_{i+\beta_m-\beta_n}^u + \sum_{\substack{l=1 \\ l\neq n}}^{J^{(2)}} \left[\sum_{\substack{j=1 \\ j\neq l}}^{J^{(2)}} \Diamond y_{i+(\beta_m-\beta_n)-(\beta_j-\beta_l)}^u \Diamond \right. \right. \right.$$

$$\left. \left. \left. \sum_{k=1}^{J^{(1)}} \Diamond y_{i+(\beta_m-\beta_n)-(\varepsilon_k-\beta_l)}^p \right] \right) \Diamond \right.$$

$$\left. \sum_{q=1}^{J^{(1)}} \Diamond \left(y_{i+\beta_m-\varepsilon_q}^p + \sum_{\substack{l=1 \\ l\neq q}}^{J^{(1)}} \left[\sum_{j=1}^{J^{(2)}} \Diamond y_{i+(\beta_m-\varepsilon_q)-(\beta_j-\varepsilon_l)}^u \Diamond \sum_{\substack{k=1 \\ k\neq l}}^{J^{(1)}} \Diamond y_{i+(\beta_m-\varepsilon_q)-(\varepsilon_k-\varepsilon_l)}^p \right] \right) \right\} \qquad (5.21)$$

$$\lambda_p^{(2)}(i) = y_p^u + \sum_{m=1}^{J^{(1)}} \left\{ \sum_{n=1}^{J^{(2)}} \Diamond \left(y_{i+\varepsilon_m-\beta_n}^u + \sum_{\substack{l=1 \\ l\neq n}}^{J^{(2)}} \left[\sum_{\substack{j=1 \\ j\neq l}}^{J^{(2)}} \Diamond y_{i+(\varepsilon_m-\beta_n)-(\beta_j-\beta_l)}^u \Diamond \right. \right. \right.$$

$$\left. \left. \left. \sum_{k=1}^{J^{(1)}} \Diamond y_{i+(\varepsilon_m-\beta_n)-(\varepsilon_k-\beta_l)}^p \right] \right) \Diamond \right.$$

$$\sum_{\substack{q=1 \\ q\neq m}}^{J^{(1)}} \Diamond \left(y^{p}_{i+\varepsilon_m-\varepsilon_q} + \sum_{\substack{l=1 \\ l\neq q}}^{J^{(1)}} \left[\sum_{j=1}^{J^{(2)}} \Diamond y^{u}_{i+(\varepsilon_m-\varepsilon_q)-(\beta_j-\varepsilon_l)} \Diamond \sum_{\substack{k=1 \\ k\neq l}}^{J^{(1)}} \Diamond y^{p}_{i+(\varepsilon_m-\varepsilon_q)-(\varepsilon_k-\varepsilon_l)} \right] \right) \right\} \tag{5.22}$$

À partir de (5.21) et (5.22), on déduit les conditions pour que $\lambda_u^{(2)}(i)$ et $\lambda_p^{(2)}(i)$ soient des équations de variables indépendantes. Ces conditions de double orthogonalité des codes convolutionnels récursifs sont les suivantes :

1. $(\beta_m - \beta_n)$, $n \neq m$, sont distincts (orthogonalité d'ordre 1) ;
2. $(\beta_m - \beta_n) - (\beta_j - \beta_l)$, $n \neq m$ et $j \neq l$, $l \neq n$ sont distincts (orthogonalité d'ordre 2) ;
3. $(\beta_m - \beta_n) - (\varepsilon_j - \beta_l)$, $n \neq m$ et $j \neq l$, $l \neq n$ sont distincts (orthogonalité d'ordre 2 entre $\{\beta_m\}$ et $\{\varepsilon_q\}$) ;
4. $(\beta_m - \varepsilon_q)$ distincts (orthogonalité d'ordre 1 entre $\{\beta_m\}$ et $\{\varepsilon_q\}$) ;
5. $(\beta_m - \varepsilon_q) - (\beta_j - \varepsilon_l)$ sont distincts (orthogonalité d'ordre 2 entre $\{\beta_m\}$ et $\{\varepsilon_q\}$) ;
6. $(\beta_m - \varepsilon_q) - (\varepsilon_k - \varepsilon_l)$ sont distincts (orthogonalité d'ordre 2 entre $\{\beta_m\}$ et $\{\varepsilon_q\}$) ;
7. $(\varepsilon_m - \varepsilon_q)$, $q \neq m$, sont distincts (orthogonalité d'ordre 1) ;
8. $(\varepsilon_m - \varepsilon_q) - (\varepsilon_k - \varepsilon_l)$, $q \neq m$ et $k \neq l$ sont distincts (orthogonalité d'ordre 2).

La condition 2 implique nécessairement la condition 1 et de la même manière la condition 8 implique la condition 7. Les conditions 3, 5 et 6 représentent les conditions de

double orthogonalité entre les ensembles $\{\beta_m\}$ et $\{\varepsilon_q\}$ alors que la condition 4 conduit à une condition d'orthogonalité d'ordre 1 (simplement orthogonal) entre les deux ensembles $\{\beta_m\}$ et $\{\varepsilon_q\}$. Tout comme pour les codes CSO^{2}C-WS, pour toutes les conditions de double orthogonalité (conditions 2, 3, 5 et 8), le problème de permutation et donc de répétitions indésirables de certains symboles codés se pose à la deuxième itération de décodage. Certaines répétitions peuvent être évitées alors que d'autres ne le peuvent pas. Les deux conditions, $l \neq n$ et $q \neq l$ introduites dans les équations (5.21) et (5.22) permettent d'éliminer les répétitions évitables de certains symboles. Cependant, d'autres répétitions de symboles demeurent car, elles sont produites par des permutations inévitables d'indices. Par exemple, en se référant à (5.22), avec $k \neq l$ et $q \neq l$, le symbole de parité $y^p_{i+(\varepsilon_m-\varepsilon_q)-(\varepsilon_k-\varepsilon_l)}$ se répète au moins $(J^{(1)}-2)$ fois. L'ajout des deux conditions $l \neq n$ et $q \neq l$ permettent, comme l'indique la procédure de Gallager [20], de réduire la corrélation entre les observables puisqu'à une itération donnée, les équations qui contiennent le symbole à décoder sont aussi exclues. Les conditions énumérées ci-dessus représentent alors les conditions nécessaires pour qu'un code convolutionnel récursif et systématique de taux de codage r_c=1/2 soit doublement orthogonal au sens large. Elles sont aussi équivalentes aux conditions de double orthogonalité d'un code J=($J^{(2)}$+$J^{(1)}$-1), CSO^{2}C-WS, r_c=1/2, spécifié par l'ensemble $\{\alpha_j\} = \{\beta_m\} \cup \{\varepsilon_q\} \backslash \varepsilon_1$. L'algorithme de décodage des codes R-CSO^{2}C-WS se décrit alors de la façon suivante :

ÉTAPE 1 (initialisation)

Itération 1: Pour chaque instant i, $h = 1, 2, \ldots J^{(2)}$ et $v = 1, 2, \ldots J^{(1)}$ évaluer

$$\lambda^{(1)}_{u,h}(i) = y^u_i + \sum_{\substack{l=1 \\ l \neq h}}^{J^{(2)}} \left(\sum_{\substack{j=1 \\ j \neq l}}^{J^{(2)}} \Diamond\, y^u_{i+\beta_l-\beta_j} \Diamond \sum_{k=1}^{J^{(1)}} \Diamond\, y^p_{i+\beta_l-\varepsilon_k} \right) \tag{5.23}$$

$$\lambda^{(1)}_{p,v}(i) = y^p_i + \sum_{\substack{l=1 \\ l \neq v}}^{J^{(2)}} \left(\sum_{j=1}^{J^{(2)}} \Diamond\, y^u_{i+\varepsilon_l-\beta_j} \Diamond \sum_{\substack{k=1 \\ k \neq l}}^{J^{(1)}} \Diamond\, y^p_{i+\varepsilon_l-\varepsilon_k} \right) \tag{5.24}$$

ÉTAPE 2 (Procédure itérative)

Itération μ, μ=2, 3, ... M : Pour chaque instant i, $h = 1, 2, \ldots J^{(2)}$ et $v = 1, 2, \ldots J^{(1)}$ évaluer

$$\lambda^{(\mu)}_{u,h}(i) = y^u_i + \sum_{\substack{m=1 \\ m \neq h}}^{J^{(2)}} \left(\sum_{\substack{n=1 \\ n \neq m}}^{J^{(2)}} \Diamond\, \lambda^{(\mu-1)}_{u,m}(i+\beta_m-\beta_n) \Diamond \sum_{q=1}^{J^{(1)}} \Diamond\, \lambda^{(\mu-1)}_{p,q}(i+\beta_m-\varepsilon_q) \right) \tag{5.25}$$

$$\lambda^{(\mu)}_{p,v}(i) = y^p_i + \sum_{\substack{m=1 \\ m \neq v}}^{J^{(1)}} \left(\sum_{n=1}^{J^{(2)}} \Diamond\, \lambda^{(\mu-1)}_{u,n}(i+\varepsilon_m-\beta_n) \sum_{\substack{q=1 \\ q \neq m}}^{J^{(1)}} \Diamond\, \lambda^{(\mu-1)}_{p,m}(i+\varepsilon_m-\varepsilon_q) \right) \tag{5.26}$$

ÉTAPE 3 (Décision finale)

a) Pour chaque instant i et à l'itération μ, $\mu=M$, évaluer

$$\lambda_u^{(M)}(i) = y_i^u + \sum_{m=1}^{J^{(2)}} \left(\sum_{\substack{n=1 \\ n \neq m}}^{J^{(2)}} \Diamond \lambda_{u,m}^{(M-1)}(i+\beta_m-\beta_n) \Diamond \sum_{q=1}^{J^{(1)}} \Diamond \lambda_{p,q}^{(M-1)}(i+\beta_m-\varepsilon_q) \right) \quad (5.27)$$

b) Décision : si $\lambda_u^{(M)}(i) \geq 0$ alors $\hat{u}_i = 1$, sinon $\hat{u}_i = 0$.

L'algorithme décrit par les relations (5.23) à (5.27) est similaire à celui présenté dans [46] et [47] où l'on présente une méthode de décodage itératif de faible complexité des codes LDPC. Les différences entre ces deux algorithmes résident surtout dans le fait que les codes LDPC sont des codes en blocs alors que les codes considérés dans ce travail sont convolutionnels.

À partir d'un code CSO^{2}C-WS ayant un taux de codage r_c =2/3 obtenu par une recherche à l'ordinateur, un code R-CSO^{2}C-WS, J=4, r_c=1/2 satisfaisant les huit conditions pour qu'un code convolutionnel systématique et récursif soit doublement orthogonal a été déterminé. La méthode utilisée pour la génération des codes est essentiellement basée sur une géométrie projective. Puisqu'aucune méthode d'optimisation n'a été appliquée ici, le code obtenu n'est pas nécessairement de longueur minimale. Ce code spécifié par les deux ensembles $\{\beta_j\}$ = {0, 958, 959, 973} et $\{\varepsilon_j\}$={0, 200, 1113, 1659} a été simulé en utilisant l'algorithme décrit par les relations (5.23) à (5.27) et les résultats de simulation sont montrés à la Figure 5.1.

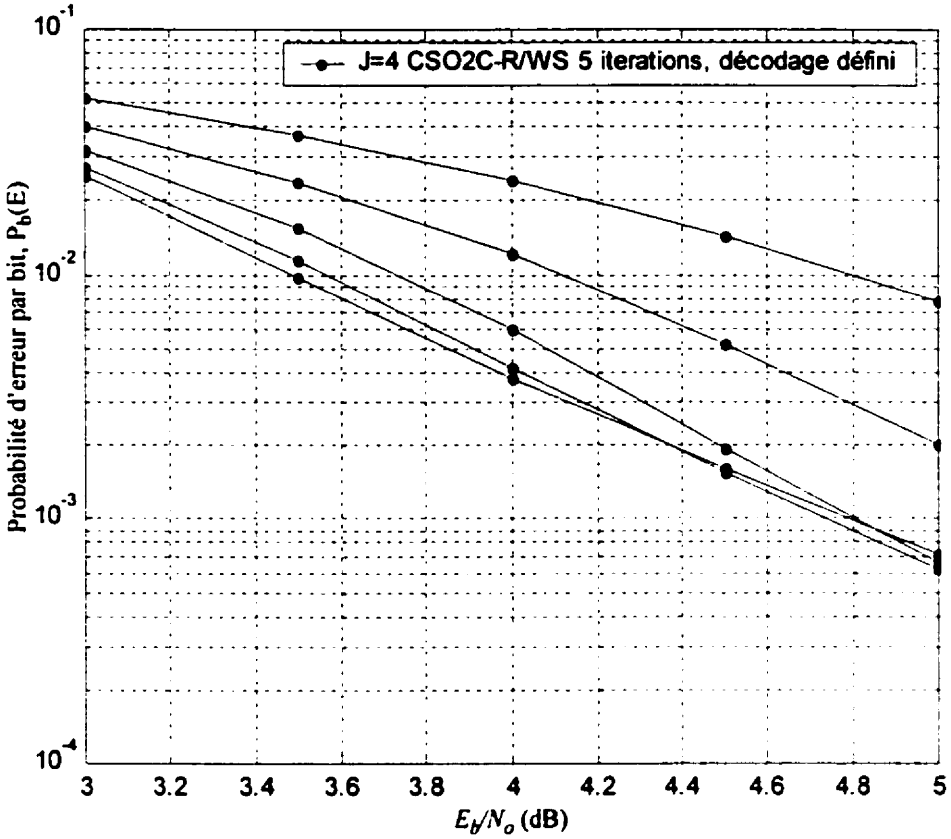


Figure 5.1 Résultats de simulation d'un code R-CSO^{2}C-WS, J=4, r_c=1/2 obtenu avec un décodage défini PAP itératif

Comme nous l'avons mentionné aux chapitres 2 et 4, le décodage défini procure généralement de moins bonnes performances d'erreur que le décodage à seuil ou PAP avec rétroaction de la décision. L'introduction d'une rétroaction de la décision à chaque étape de décodage se fait facilement en apportant une simple modification de l'algorithme décrit par les équations (5.23) à (5.27). Cette modification consiste à remplacer, dans les équations (5.23) à (5.27), tous les termes dont leur indice temporel est ($i+f$) avec $f < 0$, par le terme de la rétroaction de la décision correspondant. En utilisant le même code R-

CSO^{2}C-WS, J=4, r_c=1/2 que précédemment, des résultats de simulation ont été obtenus avec un décodage avec rétroaction de la décision à chaque itération. Ces résultats qui sont présentés à la Figure 5.2 montrent en effet une amélioration substantielle des performances d'erreur par rapport celles obtenues avec un décodage défini à chaque itération. Par exemple, pour un rapport signal à bruit de 5 dB, on observe une amélioration des performances d'erreur d'environ deux ordres de grandeur.

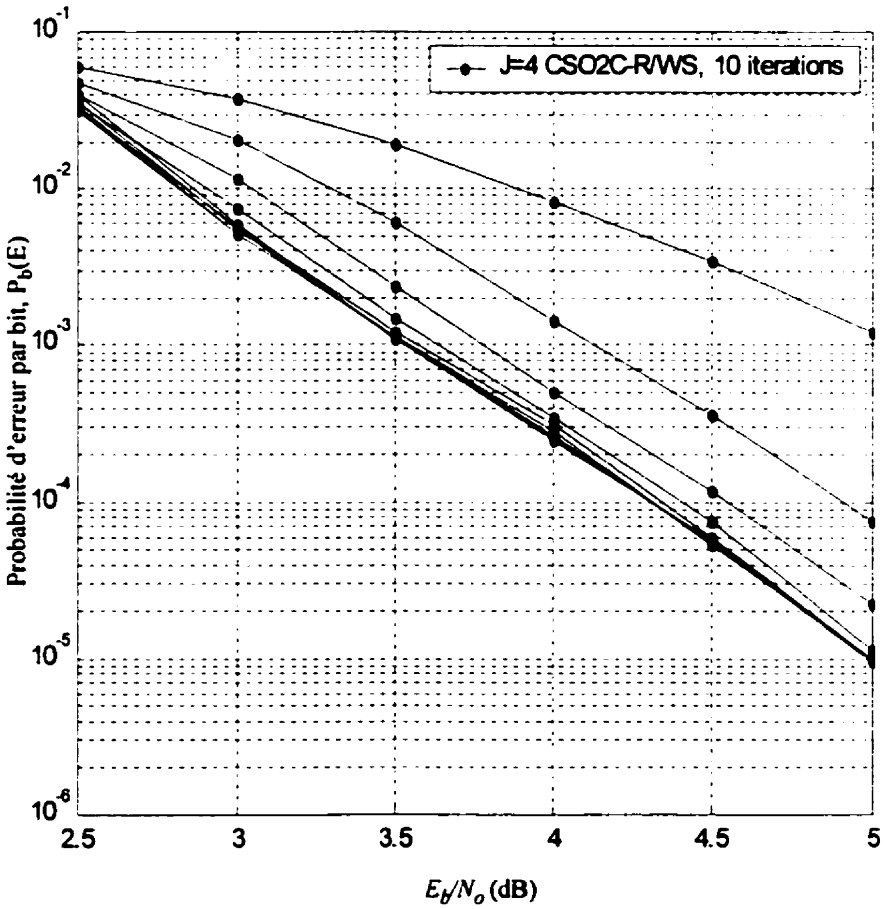


Figure 5.2 Résultats de simulation d'un code R-CSO^{2}C-WS, J=4, r_c=1/2 obtenu avec une procédure itérative de décodage PAP avec rétroaction de la décision

D'autre part, une amélioration d'environ 1.3 dB pour une probabilité d'erreur par bit $P_b(E)=10^{-3}$ est possible en introduisant une rétroaction de la décision à chaque étape de décodage. Ces deux résultats de simulation montrent aussi qu'environ 5 à 6 itérations sont nécessaires pour atteindre la convergence du processus.

5.2.3 Codes convolutionnels doublement orthogonaux (d'ordre 2) récursifs au sens strict

Les principes établis à la sections 5.2.2 et qui ont conduit à la définition au sens large des codes convolutionnels doublement orthogonaux récursifs peuvent être étendus afin d'obtenir une définition au sens strict des codes convolutionnels doublement orthogonaux récursifs, R-CSO^{2}C-SS. Les codes en bloc doublement orthogonaux définis au sens strict et de taux de codage $r_c = K/(K+J)$ présentés à la section 5.1.2 peuvent être aussi considérés comme des codes convolutionnels doublement orthogonaux au sens strict de taux de codage $K/(K+J)$. Les matrices de connexions qui les représentent peuvent être à leur tour considérées comme la transposé des matrices de contrôle de parité de codes récursifs. Si un code est récursif alors sa matrice de contrôle de parité doit être sans rétroaction pour qu'un décodage PAP soit applicable. Dans ce cas, la matrice de contrôle de parité sans rétroaction génère aussi le code dual non-récursif correspondant au code d'origine récursif. De plus, il est possible de construire des codes R-CSO^{2}C-SS ayant un taux de codage $r_c=J/2J=1/2$ à partir de codes CSO^{2}C-SS de taux de codage équivalent à $r_c=(2J)/(3J)=2/3$. Par conséquent, les matrices de contrôle de parité transposées, $\boldsymbol{H}^T$, des

codes R-CSO^{2}C-SS doivent avoir comme dimension $2J \times J$. Les matrices de connexions qui spécifient les codes CSO^{2}C-SS de taux de codage $J/(K+J)$ présentées dans [21] respectent aussi les conditions pour qu'un code convolutionnel systématique et récursif spécifié par une matrice $\boldsymbol{H}^T$ soit doublement orthogonal.

5.2.4 Définition des codes R-CSO^{2}C-SS

La matrice de contrôle de parité transposée, $\boldsymbol{H}^T$, qui spécifie le code, R-CSO^{2}C-SS ayant J équations de contrôle de parité a comme dimension $2J \times J$. Pour faciliter la représentation du code, la matrice transposée $\boldsymbol{H}^T$ est divisée en deux sous-matrices, $\boldsymbol{B}$ et $\boldsymbol{E}$ de même dimension $J \times J$. La matrice $\boldsymbol{H}^T$ s'écrit alors comme suit :

$$\boldsymbol{H}^T = \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{E} \end{bmatrix} \quad \begin{matrix} \\ \updownarrow 2J \\ \\ \leftrightarrow J \end{matrix} \tag{5.28}$$

L'élément $\beta_{j,k}$ de la première matrice, $\boldsymbol{B}$, représente la position, dans le registre à décalage qui contient ces symboles d'information, du $j^{ième}$ symbole d'information connecté au $k^{ième}$ symbole de parité. Ces éléments $\beta_{j,k}$ de la première matrice, $\boldsymbol{B}$ représentent donc les connexions de la partie dite «directe» du codeur récursif. De la même manière, l'élément $\varepsilon_{j,k}$ de la deuxième matrice, $\boldsymbol{E}$ représente la position, dans le registre à décalage contenant les symboles de parité, du $j^{ième}$ symbole de parité connecté au $k^{ième}$ symbole de parité. Les éléments $\varepsilon_{j,k}$ de la deuxième matrice, $\boldsymbol{E}$ représentent alors les connexions de la partie dite

«récursive» du codeur. Notons que puisqu'une valeur $\varepsilon_{j,k}=0$ signifie la position d'un symbole de parité pris à l'instant i, cette valeur signifie aussi que ce symbole de parité n'est pas connecté à d'autres symboles de parité. Chaque symbole de parité obtenu au codage s'écrit alors de la manière suivante :

$$p_{n,i} = \sum_{k=1}^{J} \oplus u_{k,i-\beta_{k,n}} \oplus \sum_{\substack{m=1 \\ \varepsilon_{m,n}>0}}^{J} \oplus p_{n,i-\varepsilon_{m,n}}, \; n = 1, 2, \ldots, J \tag{5.29}$$

La Figure 5.3 illustre un exemple d'un codeur R-CSO^{2}C-SS, J=2, obtenu par les matrices $\boldsymbol{B}$ et $\boldsymbol{E}$ suivantes :

$$\boldsymbol{B} = \begin{bmatrix} 0 & 1 \\ 0 & 3 \end{bmatrix} \text{ et } \boldsymbol{E} = \begin{bmatrix} 2 & 0 \\ 3 & 0 \end{bmatrix}$$

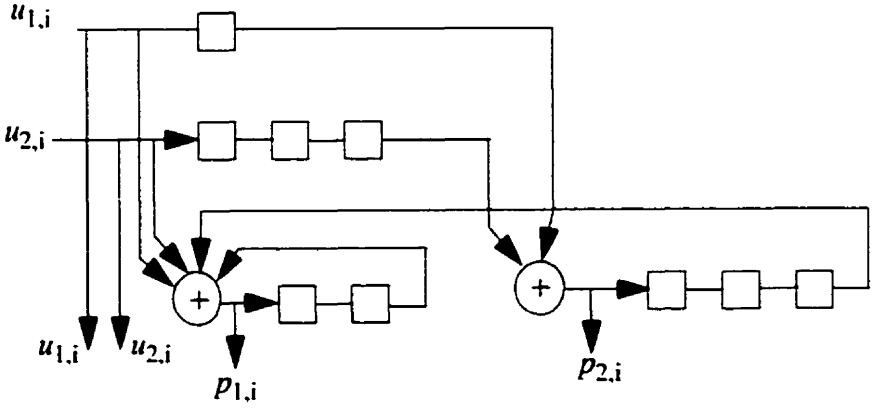


Figure 5.3 Exemple d'un codeur R-CSO^{2}C-SS, J=2, r_c=1/2

Évidemment, cette configuration n'est pas minimale puisqu'il serait toujours possible de combiner les registres à décalage utilisés pour les symboles de parité avec les registres

à décalage utilisés pour les symboles d'information. Le symbole de syndrome associé au $n^{ième}$ symbole de parité peut être déduit directement de l'équation (5.29). À la première étape de décodage, on obtient pour le symbole de syndrome $s_n(i)$ à l'instant i, l'équation suivante :

$$s_n(i) = \tilde{p}_{n,\,i} \oplus \sum_{k=1}^{J} \oplus \tilde{u}_{k,\,(i-\beta_{k,n})} \oplus \sum_{\substack{m=1 \\ \varepsilon_{m,n}>0}}^{J} \oplus \tilde{p}_{m,\,(i-\varepsilon_{m,n})},\ n=1, 2, ..., J \qquad (5.30)$$

Deux ensembles de symboles de syndrome peuvent encore être définis à partir de (5.30). Dans le premier ensemble, les symboles de syndrome sont orthogonaux au symbole d'information $\tilde{u}_{l,\,i}$ et sont obtenus par :

$$s_n(i+\beta_{l,n}) = \tilde{u}_{l,\,i} \oplus \tilde{p}_{n,\,(i+\beta_{l,n})} \oplus \sum_{\substack{k=1 \\ k\neq l}}^{J} \oplus \tilde{u}_{k,\,i+(\beta_{l,n}-\beta_{k,n})} \oplus \sum_{\substack{m=1 \\ \varepsilon_{m,n}>0}}^{J} \oplus \tilde{p}_{m,\,i+(\beta_{l,n}-\varepsilon_{m,n})} \qquad (5.31)$$

où $l = 1, 2, ..., J$, et $n=1, 2, ..., J$.

D'autre part, dans le deuxième ensemble, les symboles de syndrome qui sont orthogonaux au symbole de parité $\tilde{p}_{l,\,i}$ s'obtiennent comme suit :

$$s_n(i+\varepsilon_{l,n}) = \tilde{p}_{l,\,i} \oplus \tilde{p}_{n,\,(i+\varepsilon_{l,n})} \oplus \sum_{k=1}^{J} \oplus \tilde{u}_{k,\,i+(\varepsilon_{l,n}-\beta_{k,n})} \oplus \sum_{\substack{m=1 \\ m\neq l \\ \varepsilon_{m,n}>0}}^{J} \oplus \tilde{p}_{m,\,i+(\varepsilon_{l,n}-\varepsilon_{m,n})}, \qquad (5.32)$$

$$l = 1, 2, ..., J,\ n=1, 2, ..., J,\ \varepsilon_{l,n} > 0$$

et

$$s_l(i) = \tilde{p}_{l,i} \oplus \sum_{k=1}^{J} \oplus \tilde{u}_{k,i-\beta_{k,l}} \oplus \sum_{\substack{m=1 \\ \varepsilon_{m,l}>0}}^{J} \oplus \tilde{p}_{m,i-\varepsilon_{m,l}}, \quad l=1, 2, ..., J \tag{5.33}$$

Notons que (5.33) qui décrit le symbole de syndrome à l'instant i est identique à (5.30). Il y a donc au plus (J+1) symboles de syndrome orthogonaux à $\tilde{p}_{l,i}$, et J symboles de syndrome orthogonaux à $\tilde{u}_{l,i}$. Un algorithme de décodage PAP pour les R-CSO^{2}C-SS s'obtient facilement en suivant la même procédure qui a mené à l'algorithme de décodage pour les R-CSO^{2}C-WS. À la première itération, l'algorithme de décodage PAP se décrit par les deux équations suivantes :

$$\lambda_{u,l}^{(1)}(i) = y_{l,i}^{u} + \sum_{n=1}^{J} \left(y_{n,i+\beta_{l,n}}^{p} \Diamond \sum_{\substack{k=1 \\ k\neq l}}^{J} \Diamond y_{k,i+\beta_{l,n}-\beta_{k,n}}^{u} \Diamond \sum_{\substack{m=1 \\ \varepsilon_{m,n}>0}}^{J} \Diamond y_{m,i+\beta_{l,n}-\varepsilon_{m,n}}^{p} \right) \tag{5.34}$$

$$l=1, 2, ..., J$$

$$\lambda_{p,l}^{(1)}(i) = y_{l,i}^{p} + \left(\sum_{k=1}^{J} \Diamond y_{k,i-\beta_{k,l}}^{u} \Diamond \sum_{\substack{m=1 \\ \varepsilon_{m,l}>0}}^{J} \Diamond y_{m,i-\varepsilon_{m,l}}^{p} \right) +$$

$$\sum_{\substack{n=1 \\ \varepsilon_{l,n}>0}}^{J}\left(y^{p}_{n,\,i+\varepsilon_{l,n}} \Diamond \sum_{k=1}^{J}\Diamond\, y^{u}_{k,\,i+\varepsilon_{l,n}-\beta_{k,n}} \Diamond \sum_{\substack{m=1 \\ m\neq l \\ \varepsilon_{m,n}>0}}^{J}\Diamond\, y^{p}_{m,\,i+\varepsilon_{l,n}-\varepsilon_{m,n}} \right) \tag{5.35}$$

$$l=1, 2, ..., J$$

Une définition au sens strict des codes doublement orthogonaux récursifs qui utilise une structure parallèle pour le codage, s'obtient en effectuant une application récursive des deux équations (5.34) et (5.35). L'ensemble des conditions nécessaires pour qu'un code convolutionnel récursif soit doublement orthogonal au sens strict inclut toutes les conditions nécessaires pour obtenir un code CSO^{2}C-SS non-récursif de taux de codage $J/(K+J)$. Cependant, quelques conditions s'ajoutent aux conditions nécessaires qui ont mené à la définition d'un code CSO^{2}C-SS non-récursif de taux de codage $J/(K+J)$. Ces conditions supplémentaires qui sont au nombre de deux, sont les suivantes :

1. Pour tous les $i \neq j$, $\varepsilon_{r,\,n_i} \neq \varepsilon_{r,\,n_j}$, $\varepsilon_{r,\,n_i} \neq 0$, i=1, 2, ..., J

2. Pour tous les $i \neq j$, $\varepsilon_{r_i,\,n} \neq \varepsilon_{r_j,\,n}$, $\varepsilon_{r_i,\,n} \neq 0$, i=1, 2, ..., J

Ces deux conditions signifient que tous les éléments $\varepsilon_{r,\,n} \neq 0$ qui composent une ligne, r, de la matrice $\boldsymbol{E}$ doivent être distincts et tous les éléments $\varepsilon_{r,\,n} \neq 0$ qui composent une colonne, n, doivent être aussi dinstincts. En utilisant la même procédure qui a mené à l'algorithme de décodage des codes R-CSO^{2}C-WS, on obtient, pour les codes R-CSO^{2}C-SS, l'algorithme de décodage suivant :

ÉTAPE 1 (initialisation)

Itération 1: Pour chaque instant i, $h = 1, 2, \ldots J$ et $\rho = 1, 2, \ldots J$, $l = 1, 2, \ldots, J$, évaluer

$$\lambda^{(1)}_{u,l,h}(i) = y^{u}_{l,i} + \sum_{\substack{n=1 \\ n \neq h}}^{J} \left(y^{p}_{n,\, i+\beta_{l,n}} \Diamond \sum_{\substack{k=1 \\ k \neq l}}^{J} \Diamond\, y^{u}_{k,\, i+\beta_{l,n}-\beta_{k,n}} \Diamond \sum_{\substack{m=1 \\ \varepsilon_{m,n}>0}}^{J} \Diamond\, y^{p}_{m,\, i+\beta_{l,n}-\varepsilon_{m,n}} \right)$$

$$= y^{u}_{l,i} + \sum_{\substack{n=1 \\ n \neq h}}^{J} \psi^{(1)}_{u,l,n}(i) \qquad (5.36)$$

$$\lambda^{(1)}_{p,l,\rho}(i) = y^{p}_{l,i} + \left(\sum_{k=1}^{J} \Diamond\, y^{u}_{k,\, i-\beta_{k,l}} \Diamond \sum_{\substack{m=1 \\ \varepsilon_{m,l}>0}}^{J} \Diamond\, y^{p}_{m,\, i-\varepsilon_{m,l}} \right) +$$

$$\sum_{\substack{n=1 \\ n \neq \rho,\, \varepsilon_{l,n}>0}}^{J} \left(\sum_{k=1}^{J} \Diamond\, y^{u}_{k,\, i+\varepsilon_{l,n}-\beta_{k,n}} \Diamond \sum_{\substack{m=1 \\ m \neq l,\, \varepsilon_{m,n}>0}}^{J} \Diamond\, y^{p}_{m,\, i+\varepsilon_{l,n}-\varepsilon_{m,n}} \Diamond\, y^{p}_{n,\, i+\varepsilon_{l,n}} \right)$$

$$= y^{p}_{l,i} + \psi^{(1)}_{p,l,0}(i) + \sum_{\substack{n=1 \\ n \neq \rho,\, \varepsilon_{l,n}>0}}^{J} \psi^{(1)}_{p,l,n}(i) \qquad (5.37)$$

pour $\rho=0$:

$$\lambda^{(1)}_{p,l,0}(i) = y^{p}_{l,i} + \sum_{\substack{n=1 \\ \varepsilon_{l,n}>0}}^{J} \left(\sum_{k=1}^{J} \Diamond\, y^{u}_{k,\, i+\varepsilon_{l,n}-\beta_{k,n}} \Diamond \sum_{\substack{m=1 \\ m \neq l,\, \varepsilon_{m,n}>0}}^{J} \Diamond\, y^{p}_{m,\, i+\varepsilon_{l,n}-\varepsilon_{m,n}} \Diamond\, y^{p}_{n,\, i+\varepsilon_{l,n}} \right)$$

$$= y^{p}_{l,i} + \sum_{\substack{n=1 \\ \varepsilon_{l,n}>0}}^{J} \psi^{(1)}_{p,l,n}(i) \tag{5.38}$$

ÉTAPE 2 (Procédure itérative)

Itération μ, μ=2, 3, ..., M-1 : Pour chaque instant i, h = 1, 2, ... J et ρ=1, 2, ... J, l = 1, 2, ..., J, évaluer

$$\lambda^{(\mu)}_{u,l,h}(i) = y^{u}_{l,i} + \sum_{\substack{n=1 \\ n\neq h}}^{J} \left(\lambda^{(\mu-1)}_{p,n,0}(i+\beta_{l,n}) \Diamond \sum_{\substack{k=1 \\ k\neq l}}^{J\Diamond} \lambda^{(\mu-1)}_{u,k,n}(i+\beta_{l,n}-\beta_{k,n}) \Diamond \right.$$

$$\left. \sum_{\substack{m=1 \\ \varepsilon_{m,n}>0}}^{J\Diamond} \lambda^{(\mu-1)}_{p,m,n}(i+\beta_{l,n}-\varepsilon_{m,n}) \right)$$

$$= y^{u}_{l,i} + \sum_{\substack{n=1 \\ n\neq h}}^{J} \psi^{(\mu)}_{u,l,n}(i) \tag{5.39}$$

$$\lambda^{(\mu)}_{p,l,\rho}(i) = y^{p}_{l,i} + \left(\sum_{k=1}^{J\Diamond} \lambda^{(\mu-1)}_{u,k,l}(i-\beta_{k,l}) \Diamond \sum_{\substack{m=1 \\ \varepsilon_{m,l}>0}}^{J\Diamond} \lambda^{(\mu-1)}_{p,m,l}(i-\varepsilon_{m,l}) \right) +$$

$$\sum_{\substack{n=1 \\ n\neq\rho,\ \varepsilon_{l,n}>0}}^{J} \left(\lambda^{(\mu-1)}_{p,n,0}(i+\varepsilon_{l,n}) \Diamond \sum_{k=1}^{J\Diamond} \lambda^{(\mu-1)}_{u,k,n}(i+\varepsilon_{l,n}-\beta_{k,n}) \Diamond \right.$$

$$\left. \sum_{\substack{m=1 \\ m \neq l,\ \varepsilon_{m,n}>0}}^{J} \lambda_{p,m,n}^{(\mu-1)}(i+\varepsilon_{l,n}-\varepsilon_{m,n}) \right)$$

$$= y_{l,i}^{p} + \psi_{p,l,0}^{(\mu)}(i) + \sum_{\substack{n=1 \\ n \neq \rho,\ \varepsilon_{l,n}>0}}^{J} \psi_{p,l,n}^{(\mu)}(i) \qquad (5.40)$$

pour $\rho=0$:

$$\lambda_{p,l,0}^{(\mu)}(i) = y_{l,i}^{p} + \sum_{\substack{n=1 \\ \varepsilon_{l,n}>0}}^{J} \left(\lambda_{p,n,0}^{(\mu-1)}(i+\varepsilon_{l,n}) \Diamond \sum_{k=1}^{J} \lambda_{u,k,n}^{(\mu-1)}(i+\varepsilon_{l,n}-\beta_{k,n}) \Diamond \right.$$

$$\left. \sum_{\substack{m=1 \\ m \neq l,\ \varepsilon_{m,n}>0}}^{J} \lambda_{p,m,n}^{(\mu-1)}(i+\varepsilon_{l,n}-\varepsilon_{m,n}) \right)$$

$$= y_{l,i}^{p} + \sum_{\substack{n=1 \\ \varepsilon_{l,n}>0}}^{J} \psi_{p,l,n}^{(\mu)}(i) \qquad (5.41)$$

ÉTAPE 3 (Décision finale)

a) Pour chaque instant i et à l'itération μ, $\mu=M$, évaluer :

$$\lambda_{u,l}^{(M)}(i) = y_{l,i}^{u} + \sum_{n=1}^{J} \left(\lambda_{p,n,0}^{(M-1)}(i+\beta_{l,n}) \Diamond \sum_{\substack{k=1 \\ k \neq l}}^{J} \lambda_{u,k,n}^{(M-1)}(i+\beta_{l,n}-\beta_{k,n}) \Diamond \right.$$

$$\left. \sum_{\substack{m=1 \\ \varepsilon_{m,n}>0}}^{J} \lambda_{p,m,n}^{(M-1)}(i+\beta_{l,n}-\varepsilon_{m,n}) \right)$$

$$= y_{l,i}^{u} + \sum_{n=1}^{J} \psi_{u,l,n}^{(M)}(i) \tag{5.42}$$

b) Décision : si $\lambda_{u,l}^{(M)}(i) \geq 0$ alors $\hat{u}_{l,i} = 1$, sinon $\hat{u}_{l,i} = 0$.

L'algorithme décrit par (5.36) à (5.42) est aussi similaire à celui utilisé par Gallager pour décoder les codes LDPC et est par conséquent semblable à celui décrit dans [46]. Évidemment, comme on le remarque, cet algorithme est aussi basé sur un décodage défini utilisant la probabilité *a posteriori*.

Dans le but de valider ce principe de codage récursif et cet algorithme de décodage, un code R-CSO^{2}C-SS, J=5, r_c=1/2 a été déterminé à partir d'une matrice, $\boldsymbol{H}^T$, de contrôle de parité de dimension $2J \times J = 10 \times 5$. Cette matrice, $\boldsymbol{H}^T$, obtenue par Baechler et présentée dans [21] est la suivante :

$$\boldsymbol{H}^T = \begin{bmatrix} \underline{\boldsymbol{B}} \\ \underline{\boldsymbol{E}} \end{bmatrix} = \left[\begin{array}{c} \begin{bmatrix} 0 & 299 & 0 & 142 & 1135 \\ 0 & 320 & 146 & 107 & 794 \\ 0 & 679 & 5 & 977 & 172 \\ 87 & 333 & 0 & 106 & 310 \\ 0 & 418 & 60 & 123 & 99 \end{bmatrix} \\ \hline \begin{bmatrix} 0 & 945 & 210 & 90 & 92 \\ 1135 & 0 & 552 & 192 & 0 \\ 39 & 296 & 0 & 1003 & 97 \\ 387 & 198 & 193 & 0 & 0 \\ 132 & 208 & 1073 & 3 & 0 \end{bmatrix} \end{array}\right] \tag{5.43}$$

Notons que cette matrice représente aussi un code CSO^{2}C-SS, J=5, de taux de codage $r_c = 10/15 = 2/3$. En utilisant ce code spécifié par la matrice $\boldsymbol{H}^T$ donnée par (5.43) et l'algorithme de décodage décrit par les équations (5.36) à (5.42), des résultats de simulation ont été obtenus. Une pondération des équations $\psi^{(\mu)}_{u,l,n}(i)$ par des coefficients $a^{(\mu)}_{u,l,n}$ et des équations $\psi^{(\mu)}_{u,l,n}(i)$ par des coefficients $a^{(\mu)}_{p,l,n}$ a été considérée dans ces simulations. Ces coefficients ont été déterminés selon la même méthode qu'utilisée précédemment pour des codes CSO^{2}C-WS et CSO^{2}C-SS. Les meilleures performances d'erreur sont obtenues en fixant tous les coefficients $a^{(\mu)}_{u,l,n}$=0.6 alors que tous les coefficients $a^{(\mu)}_{p,l,n}$ demeurent à la valeur 1.0. Les résultats de simulation montrés à la Figure 5.4 indiquent qu'après 16 itérations, une probabilité d'erreur d'environ 3×10^{-6} est possible avec un rapport signal à bruit $E_b/N_o = 2.5$ dB.

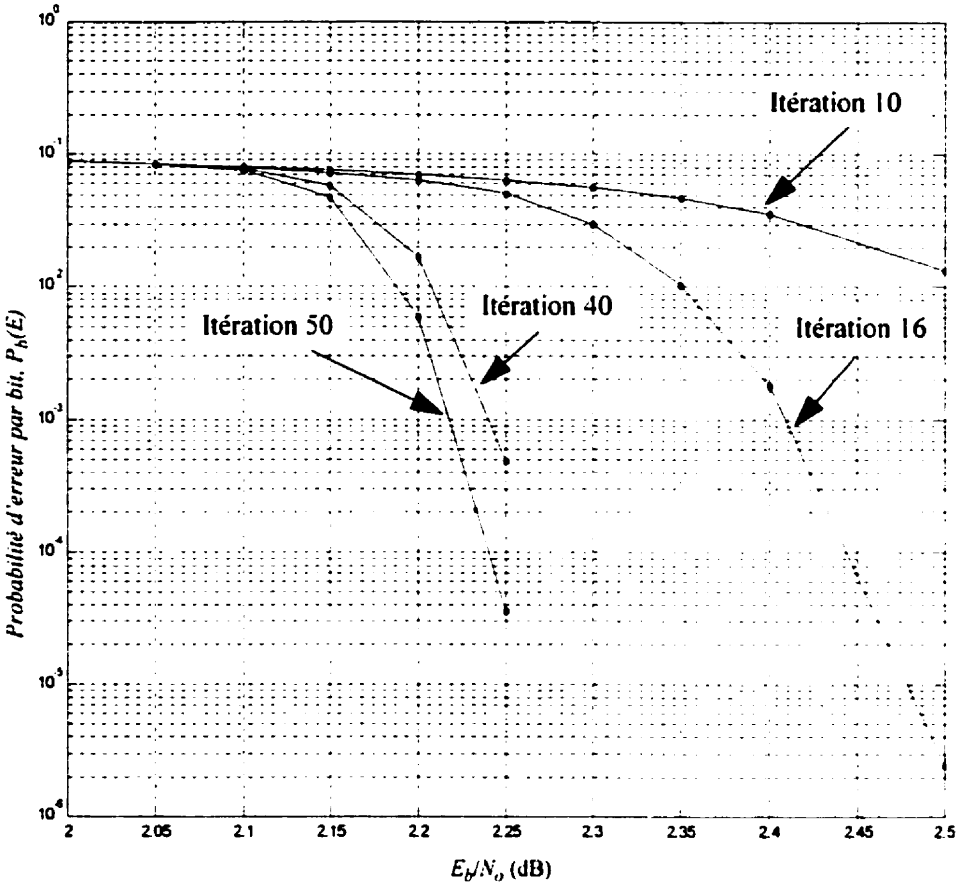


Figure 5.4 Résultats de simulation pour le code R-CSO^{2}C-SS, J=5, r_c=1/2

Cependant, pour des rapports signal à bruit E_b/N_o inférieurs à 2.0 dB, le processus de décodage itératif converge difficilement. Par exemple, pour une valeur de E_b/N_o = 2.0 dB, même après 50 itérations, la probabilité d'erreur se stabilise autour de 9×10^{-2}.

Toutefois, de meilleures performances d'erreur sont obtenues en apportant des modifications aux codes R-CSO^{2}C-SS. Il s'agit de réduire le nombre de connexions dans la partie récursive du codeur laquelle est spécifiée par la matrice $\boldsymbol{E}$. Il en résulte de cette

réduction, une diminution du nombre de symboles d'erreur qui interviennent dans les calculs des symboles de syndrome effectués à chaque itération. Par conséquent, même si le pouvoir de correction du code semble réduit, à de faibles rapports signal à bruit, E_b/N_o, les symboles de syndrome qui correspondent à un symbole de parité donné sont moins affectés par les erreurs du canal de sortes qu'ils demeurent de bons indicateurs d'erreur pour ce symbole de parité. Même à de faibles rapports signal à bruit, la convergence du processus de décodage vers de meilleures performances d'erreur devrait alors être possible. Évidemment, l'élimination complète de la rétroaction du codeur (i.e. $\boldsymbol{E} = 0$), revient à définir un code CSO^{2}C-SS. Quoique la validité de ces affirmations devrait être vérifiée théoriquement, les résultats de simulation qui suivent indiquent que ces modifications apportées à la matrice $\boldsymbol{E}$ permettent effectivement des améliorations substantielles des performances d'erreur.

Pour le code R-CSO^{2}C-SS, J=5, spécifié par la matrice $\boldsymbol{H}^T$ de l'équation (5.43), plusieurs configurations de la matrice $\boldsymbol{E}$ ont été vérifiées par simulation. À chaque essai, une connexion par symbole de parité est retirée en conservant au moins une connexion par symbole de parité. Par exemple, pour ce code R-CSO^{2}C-SS, J=5, les meilleures performances d'erreur ont été obtenues en modifiant la sous-matrice $\boldsymbol{E}$ de la façon suivante :

$$\boldsymbol{E} = \begin{bmatrix} 0 & 945 & 0 & 0 & 92 \\ 1135 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1003 & 0 \\ 0 & 0 & 193 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.44}$$

Dans cette configuration de la sous-matrice $\boldsymbol{E}$, une seule connexion est conservée par symbole de parité. D'autres configurations qui n'ont pas été explorées peuvent aussi donner de bonnes performances d'erreur. La matrice de contrôle de parité transposée, $\boldsymbol{H}^T$, pour ce code devient alors:

$$\boldsymbol{H}^T = \begin{bmatrix} \underline{\boldsymbol{B}} \\ \boldsymbol{E} \end{bmatrix} = \left[\begin{array}{c} \begin{bmatrix} 0 & 299 & 0 & 142 & 1135 \\ 0 & 320 & 146 & 107 & 794 \\ 0 & 679 & 5 & 977 & 172 \\ 87 & 333 & 0 & 106 & 310 \\ 0 & 418 & 60 & 123 & 99 \end{bmatrix} \\ \hline \begin{bmatrix} 0 & 945 & 0 & 0 & 92 \\ 1135 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1003 & 0 \\ 0 & 0 & 193 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}\right] \tag{5.45}$$

Les résultats de simulation présentés à la Figure 5.5 montrent les performances d'erreur pour le code R-CSO^{2}C-SS, J=5, spécifié par la matrice de contrôle de parité transposée donnée par (5.45). L'utilisation de coefficients de pondération a été aussi considérée dans ces simulations. Les meilleures performances d'erreur sont encore une fois obtenues lorsque tous les coefficients $a_{u,l,n}^{(\mu)}$=0.6 et que tous les coefficients $a_{p,l,n}^{(\mu)}$ demeurent à la valeur 1.0. Ces résultats de simulation indiquent que pour un rapport signal à bruit, E_b/N_o d'environ 1.25 dB, une probabilité d'erreur par bit de 10^{-6} peut être atteinte après 39 itérations. On observe aussi que l'ajout d'itérations supplémentaires permet d'améliorer davantage les performances d'erreur. En effet, pour une probabilité

d'erreur par bit de 10^{-3}, un gain supplémentaire de codage d'environ 0.05 dB peut être obtenu entre la $39^{ième}$ et la $50^{ième}$ itération. Il est clair que d'autres configurations de la sous-matrice $\boldsymbol{E}$ sont possibles, lesquelles procurent des performances d'erreur similaires à celles montrées à la Figure 5.5.

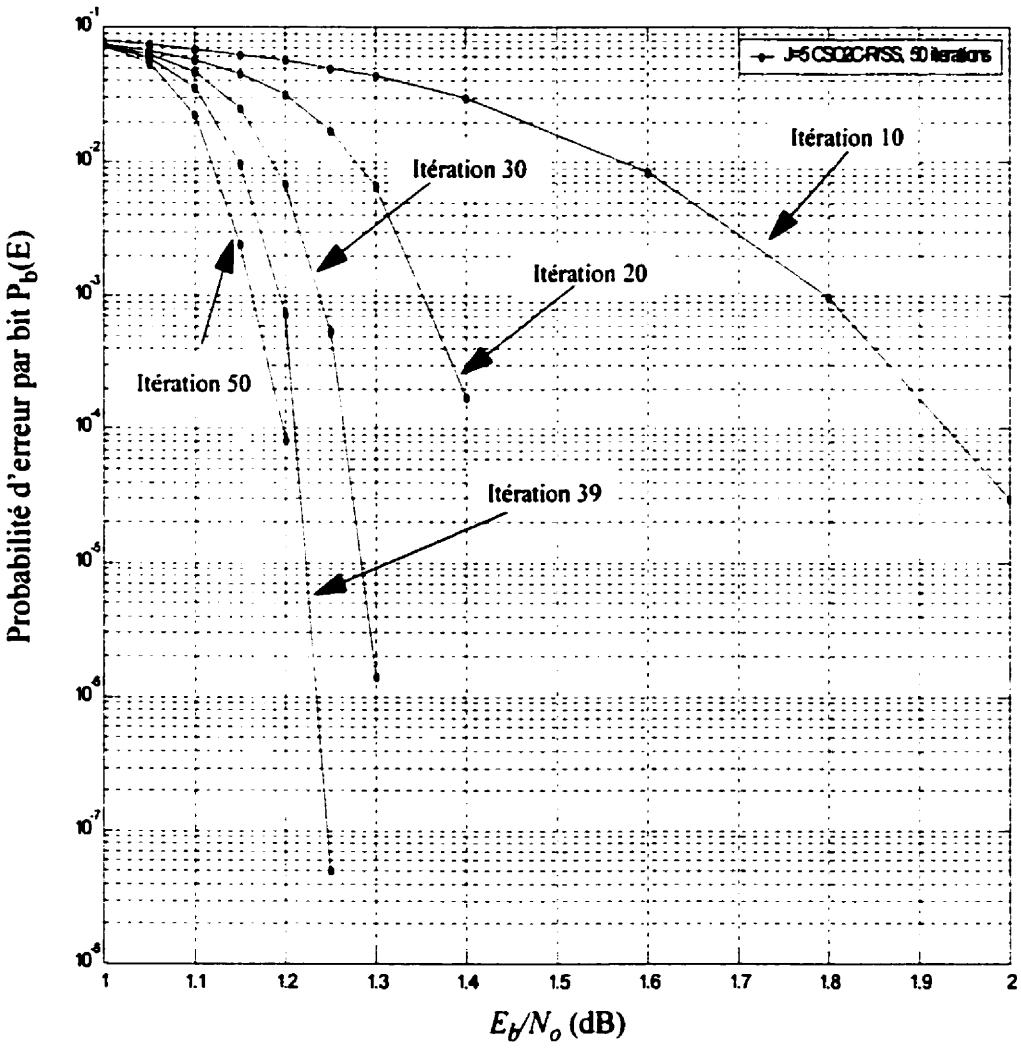


Figure 5.5 Résultats de simulation pour un R-CSO^{2}C-SS, J=5, r_c=1/2, matrice de retour modifiée

Bien que ces résultats de simulation ne fournissent pas de moyens permettant

d'expliquer théoriquement ces améliorations de performances d'erreur, ils démontrent la validité du principe de la réduction du nombre de connexions dans la partie récursive du codeur.

Quoique les performances d'erreur que procure cette technique de codage et de décodage itératif soient intéressantes, elles sont obtenues au prix d'un grand nombre d'itérations et donc d'une augmentation assez importante de la latence. Par exemple, ce processus de décodage itératif appliqué à ce code récursif R-CSO^{2}C-SS, J=5, introduit à l'itération 39, une latence considérable de $39 \times 1135 \times 5 = 221325$ bits. D'autre part, une comparaison de ces résultats de simulation avec ceux d'un code CSO^{2}C-SS, J=10, révèle que pour une probabilité d'erreur par bit de 10^{-6}, ce code R-CSO^{2}C-SS, J=5 procure un gain supplémentaire de codage d'environ 1.75 dB par rapport à un code CSO^{2}C-SS, J=10 de longueur 12426 bits. Notons qu'avec ce code CSO^{2}C-SS, J=10, quatre itérations sont nécessaires afin d'atteindre une probabilité d'erreur par bit de 10^{-6}. Par conséquent, une latence de $4 \times 12426 \times 10 = 497040$ bits est introduite par le décodage d'un code CSO^{2}C-SS, J=10 de longueur 12426 bits. Quoique cette latence est environ deux fois plus grande que celle introduite par le décodage du code récursif, ces deux valeurs de latence sont dans un même ordre de grandeur.

L'effet de la présence d'une rétroaction de la décision à chaque étape de décodage a été aussi vérifié par simulation. Les résultats de simulation montrés à la Figure 5.6 illustrent l'effet obtenu lorsqu'une rétroaction de la décision est introduite à chaque étape de décodage pour le code R-CSO^{2}C-SS, J=5 spécifié par la matrice de contrôle de parité

transposée donnée par (5.45). On constate une dégradation des performances d'erreur par rapport à l'utilisation d'un décodage défini. La convergence est atteinte plus rapidement avec un décodeur avec rétroaction de la décision qu'avec un décodage défini. Il ne suffit que de 15 itérations pour que le processus itératif de décodage atteigne la convergence.

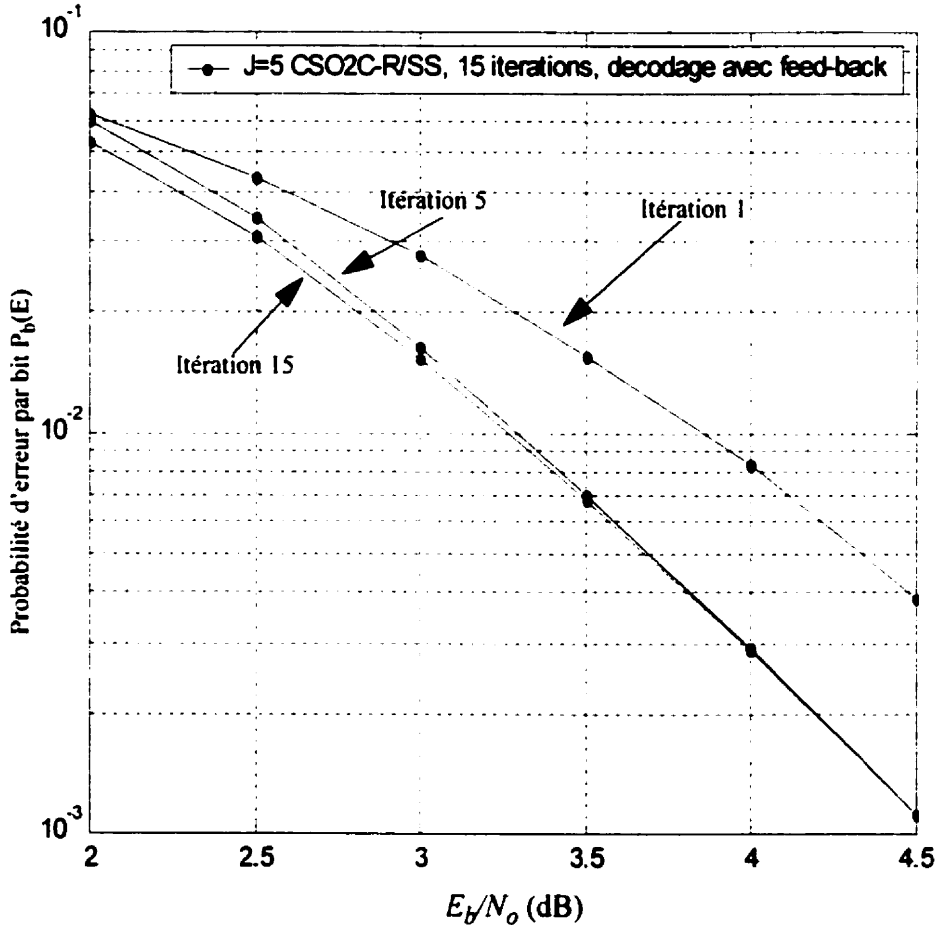


Figure 5.6 Résultat de simulation montrant l'effet d'une rétroaction de la décision à chaque étape de décodage du processus itératif de décodage d'un code J=5, R-CSO^{2}C-SS, $r_c = 1/2$

Cette dégradation des performances s'explique possiblement par l'introduction de corrélation entre les observables à chaque itération. Elle peut être aussi une conséquence

d'une faible propagation d'erreurs introduite par la présence de la rétroaction de la décision à chaque itération. Une propagation d'erreurs se produit lorsque le nombre d'erreurs introduites par le canal de transmission est supérieur au nombre d'erreurs corrigeables par un seul décodeur avec rétroaction de la décision [44].

Pour que le processus de décodage itératif puisse converger, il faut nécessairement que le nombre d'erreurs introduites dans chaque symbole de syndrome puisse diminuer d'une itération à l'autre. Sans aucun doute, une propagation d'erreurs à la première itération provoquera une propagation d'erreurs aux itérations suivantes. Avec un décodage défini et donc sans rétroaction de la décision, la possibilité d'une propagation d'erreurs est réduite de sorte que le processus de décodage itératif a une plus grande facilité à converger et ce, même à de très faibles rapports signal à bruit.

5.3 Conclusion

Dans ce chapitre on a présenté une autre technique de codage utilisant des codes convolutionnels doublement orthogonaux récursifs. Ceux-ci sont encore définis au sens large et au sens strict. L'aspect qui ressort le plus avec l'utilisation de ces codes récursifs et cette méthode de décodage itératif est certainement le fait que, non seulement la fiabilité des symboles d'information peut être améliorée à chaque itération mais aussi, qu'il en résulte à chaque itération, une amélioration de la fiabilité des symboles de parité. Selon les résultats de simulation présentés ici, les codes R-CSO^{2}C procurent de très bonnes performances d'erreur et ce, tout particulièrement avec des codes R-CSO^{2}C-SS. Cependant, on a pu constater qu'un nombre important d'itérations est nécessaire pour

atteindre ces performances d'erreur. De plus, il semble que les performances d'erreur dépendent largement du nombre de connexions dans la partie récursive du codeur. Notamment, on constate qu'une réduction du nombre de connexions de la partie récursive du codeur procure une amélioration substantielle des performances d'erreur. Cette remarque pourrait donc servir de ligne directrice pour la recherche des meilleurs codes R-CSO^{2}C-SS. La comparaison entre un décodage utilisant une rétroaction de la décision à chaque itération et un décodage défini révèle que ce dernier fournit de meilleures performances d'erreur et cela, même à de très faibles rapports signal à bruit.

Le problème de l'implantation de ces codes dans des systèmes de communication n'entre pas dans le cadre de ce travail. Cependant, il est intéressant de noter que la chute rapide de la probabilité d'erreur obtenue avec ces codes R-CSO^{2}C-SS peut poser un problème de stabilité pour les circuits de contrôle automatique de gain et de verrouillage de phase utilisés dans la plupart des récepteurs numériques. En effet, une variation du rapport signal à bruit aussi faible que 0.1 dB provoque une dégradation des performances d'erreur d'environ deux ordres de grandeur lorsqu'un code J=5, R-CSO^{2}C-SS, r_c = 1/2 est utilisé.

CHAPITRE 6: ANALYSE DE LA COMPLEXITÉ ET DE LA LATENCE DU DÉCODAGE ITÉRATIF SANS ENTRELACEMENT

La faible complexité du processus de décodage itératif des codes CSO^2C devrait améliorer l'efficacité des systèmes de communication à haute vitesse. En particulier, on peut cibler des applications où une contrainte de délai est imposée. Par exemple, dans un système de transmission de la parole, le délai de transmission limite la dimension des blocs d'information transmis à environ 200 bits [52], [53]. De plus, la complexité dépend essentiellement de la quantité de mémoire requise pour l'implantation matérielle des algorithmes de décodage itératif. Dans cette optique, les techniques de codage et de décodage qui ont été développées dans cette thèse doivent être comparées à d'autres techniques de correction d'erreurs. Les comparaisons sont faites en fonction de la complexité, de la latence et des performances d'erreur.

Les calculs de complexité sont réalisés en considérant le nombre d'opérations à effectuer pour décoder un bit d'information. Quant à la latence, elle est évaluée en fonction du nombre de bits que le processus doit attendre avant de délivrer la décision finale sur un symbole d'information donnée. Le temps de traitement requis pour chaque bit ne sera donc pas considéré car, ce temps dépend d'une réalisation matérielle particulière du processus de décodage. On y compare les processus de décodage à seuil des différents types de codes CSO^2C (sens large, sens strict et récursif au sens strict) avec

d'autres techniques conventionnelles de décodage itératif «Turbo». Une de ces techniques de décodage itératif «Turbo», qui utilise entre autres un décodage à seuil des codes convolutionnels orthogonaux, CSOC, mais, cette fois-ci avec entrelacement [12] sera tout particulièrement considérée. Dans ce dernier cas, le code est obtenu par une concaténation parallèle de deux codeurs convolutionnels CSOC.

À la section 6.1, la complexité définie en nombre d'opérations élémentaires est déterminée pour le décodage des trois types de codes CSO^{2}C-WS, CSO^{2}C-SS et R-CSO^{2}C-SS. Par la suite, à la section 6.2, on y compare tout d'abord les complexités et latence des techniques de décodage itératif à seuil de quelques codes CSO^{2}C-WS et CSO^{2}C-SS avec la complexité et la latence du décodage itératif à seuil d'un code obtenu par concaténation parallèle de deux codeurs CSOC. Par la suite, les résultats des calculs de complexité et de latence du décodage des codes CSO^{2}C-WS et CSO^{2}C-SS sont comparés à la complexité et à la latence du décodage Turbo d'un code obtenu par une concaténation parallèle de deux codeurs convolutionnels systématiques et récursifs (CPCCSR). Dans ces comparaisons, les codes CPCCSR utilisent un entrelacement de petite taille. Finalement, une comparaison est aussi obtenue entre la technique de décodage itératif d'un code R-CSO^{2}C-SS et celle d'un code CPCCSR ayant une grande taille d'entrelacement.

6.1 Calculs de la complexité et de la latence du décodage itératif des CSO^{2}C sans entrelacement

Les calculs de complexité de chaque algorithme de décodage développé dans cette

thèse sont obtenus en fonction du nombre d'opérations nécessaires par bit décodé que ces algorithmes de décodage doivent effectuer à chaque itération. Le calcul de la latence engendrée par ces processus de décodage itératif, a déjà été traité à la section 3.7 et à la section 5.2.4. Rappelons que la latence du processus de décodage telle que définie dans ce travail s'exprime en nombre de bits. Dans ce qui suit, on décrit la complexité et la latence de l'algorithme de décodage itératif pour les trois types de codes suivants: CSO^{2}C-WS, CSO^{2}C-SS et R-CSO^{2}C-SS.

6.1.1 Complexité et latence du décodage itératif des CSO^{2}C-WS

Comme décrit au chapitre 2 et utilisé au chapitre 3, l'algorithme de décodage à seuil utilisé pour les codes CSO^{2}C-WS est basé sur un décodage à rétroaction de la décision. Le calcul de complexité est fait ici en déterminant le nombre d'opérations élémentaires que le décodeur doit effectuer par itération et pour chaque bit décodé. Les codes CSO^{2}C-WS considérés dans ce calcul de complexité sont de taux de codage r_c=1/2, ont une mémoire de longueur m=α_J et produisent, au décodage, J équations de syndromes.

6.1.1.1 Calcul du nombre d'opérations add-min à deux entrées par itération

Pour un décodeur à seuil à rétroaction de la décision, J opérations add-min de J entrées sont nécessaires. Chaque opération add-min de J entrées peut être décomposée en (J-1) opérations add-min de deux entrées. Ainsi, au total, le nombre nécessaire d'opérations add-min de deux entrées par itération est $J(J$-1). Ce nombre d'opérations add-min peut être réduit en utilisant une architecture semblable à celle décrite dans [19] où les termes de

la rétroaction sont combinés aux symboles de parité reçus (voir Figure 2.7). Dans ce cas, le nombre d'opérations add-min de deux entrées nécessaires par itération et par bit d'information décodé devient $(J(J\text{-}1))/2+(J\text{-}1)$ soit une réduction qui tend vers 50 % pour une valeur de J élevé. Notons aussi que chaque opérateur add-min de deux entrées peut se décomposer en deux opérations : addition modulo-2 et minimum.

6.1.1.2 Calcul du nombre d'additions de deux valeurs réelles

À chaque itération, le décodeur à seuil doit effectuer une somme de $J+1$ valeurs réelles. Par conséquent, pour chaque somme de $(J+1)$ valeurs réelles, J additions de deux valeurs réelles sont requises.

6.1.1.3 Calcul du nombre de multiplications de valeurs réelles

Pour une pondération des équations ψ_{ij} et du symbole d'information courant, le nombre de multiplications de valeurs réelles requises est $(J+1)$ par itération. Cependant, comme il a été démontré au chapitre 3, les coefficients de pondération sont généralement tous identiques de sorte que, le nombre de multiplications requises par itération se réduit à une seule opération de multiplication.

Le tableau 6.1 résume le nombre requis de chaque type d'opération que le décodeur à seuil à rétroaction de la décision doit effectuer à chaque itération et ce, pour chaque bit d'information décodé.

Tableau 6.1: Décodage à seuil à rétroaction de la décision pour des CSO^{2}C-WS

Type d'opération	Nombre d'opérations par itération et par bit d'information décodé
Addition mod-2	$(J(J-1)/2+(J-1)$
Minimum (min)	$(J(J-1)/2+(J-1)$
Somme réelle	J
Multiplication	$(J+1)$, si coefficients de pondération distincts 1, si coefficients de pondération identiques

La latence du processus de décodage itératif à seuil dépend de la longueur de la mémoire du codeur CSO^{2}C-WS. Ainsi, pour chaque itération, la latence L_{DAS} qu'introduit le décodeur à seuil est égale à α_J. La latence totale $L_{DAS\text{-}TOT}$ du processus de décodage itératif à seuil s'obtient simplement en multipliant le nombre total d'itérations, M, par la valeur α_J : $L_{DAS\text{-}TOT} = M\alpha_J$.

6.1.2 Complexité et latence du décodage itératif à seuil des CSO^{2}C-SS

Le taux de codage des codes CSO^{2}C-SS considérés dans cette thèse étant de la forme $J/2J = 1/2$, le nombre d'opérations par itération de décodage à seuil pour ces CSO^{2}C-SS devient J fois plus grande que la complexité du décodage des codes CSO^{2}C-WS de taux de codage r_c=1/2. Cependant, puisque J bits d'information sont décodés à chaque itération, la complexité de l'algorithme de décodage itératif à seuil pour les codes CSO^{2}C-SS calculé en nombre d'opérations par itération et par bit d'information décodé demeure

identique à la complexité de l'algorithme de décodage itératif pour les codes CSO^{2}C-WS. Contrairement aux codes CSO^{2}C-WS, il a été déterminé au chapitre 3 qu'il n'est généralement pas nécessaire d'appliquer une pondération des équations de parité pour des codes CSO^{2}C-SS. Dans de tels cas, aucune opération de multiplication n'est requise. Un résumé des calculs de la complexité du décodage itératif à seuil pour des codes CSO^{2}C-SS de taux de codage $r_c=J/2J=1/2$ est donné au tableau 6.2.

Tableau 6.2: Décodage à seuil à rétroaction de la décision pour des CSO^{2}C-SS

Type d'opérations	Nombre d'opérations par itération et par bit d'information décodé
Addition mod-2	$(J(J-1)/2+(J-1)$
Minimum (min)	$(J(J-1)/2+(J-1)$
Somme réelle	J
Multiplication	$(J+1)$, si coefficients de pondération distincts 1, si coefficients de pondération identiques 0, si aucun coefficient de pondération

En ce qui concerne le calcul de la latence de chaque étape de décodage, on trouve aussi une latence proportionnelle à la taille de la mémoire du codeur CSO^{2}C-SS, $r_c=J/2J$. Puisqu'il y a J registres à décalage, la taille maximale de la mémoire d'un codeur CSO^{2}C-SS est donnée par $m = J \times \max_{1 \le k,n \le J} \{\alpha_{k,n}\}$ où les valeurs $\alpha_{k,n}$ sont les entiers qui spécifient le code CSO^{2}C-SS. Évidemment, la latence totale $L_{DAS\text{-}TOT}$ du processus de décodage

itératif s'obtient toujours en effectuant le produit de la taille de la mémoire du codeur avec le nombre total d'itérations M : $L_{DAS\text{-}TOT}=mM$.

6.1.3 Complexité et latence du décodage itératif des R-CSO^{2}C-SS

Dans le but d'évaluer la complexité de l'algorithme de décodage des codes R-CSO^{2}C-SS, on se réfère aux équations (5.36) à (5.42) qui décrivent l'algorithme de décodage. Comme auparavant, la complexité de l'algorithme utilisé dans ce cas dépend essentiellement de la valeur de J. Elle est aussi fonction du nombre d'éléments non-nuls dans la sous-matrice $\boldsymbol{E}$ qui spécifie les connexions de la partie arrière du code R-CSO^{2}C-SS. Nous avons constaté au chapitre 5 que les meilleures performances d'erreur sont obtenues lorsque le nombre d'éléments non-nuls constituant la matrice $\boldsymbol{E}$ est petit. En particulier, pour le code R-CSO^{2}C-SS, J=5 présenté à la section 5.2.4, chaque colonne de la matrice $\boldsymbol{E}$ comporte un seul élément non-nul. Évidemment, si tous les J^2 éléments de la matrice $\boldsymbol{E}$ sont non-nuls, la complexité de l'algorithme pour ce code est maximale.

Tout d'abord, définissons le nombre d'éléments $\varepsilon_{m,n}$, non-nuls dans la colonne n de la sous-matrice $\boldsymbol{E}$ par J_n^c et le nombre d'éléments $\varepsilon_{m,n}$, non-nuls dans la ligne m par J_m^r. En utilisant ces définitions et en se basant sur les équations (5.36) à (5.42), on détermine la complexité de l'algorithme de décodage pour les codes R-CSO^{2}C-SS. Il suffit alors de déterminer le nombre nécessaire d'opérations «add-min» et «addition» par itération et par bit. Cependant, pour tenir compte de la condition $\varepsilon_{m,n}>0$ qui apparaît dans les équations (5.36) à (5.42), des bornes supérieures sur les nombres d'opérations «add-min» et

«addition» sont calculées. Les résultats de ces calculs sont montrés au tableau 6.3.

Tableau 6.3: Complexité de l'algorithme de décodage itératif pour les codes R-CSO^{2}C-SS

Paramètres	nombre de add-min (2 entrées) par itération et par bit	nombre d'additions par itération et par bit
$\psi_{u,l,n}^{(\mu)}(i)$	$J^2 - J + \sum_{n=1}^{J} f_n^c$	0
$\psi_{p,l,n}^{(\mu)}(i)$	$\leq \frac{1}{J} \sum_{l=1}^{J} \sum_{n=1}^{f_l^r} f_n^c + \sum_{n=1}^{J} f_n^r$	0
$\psi_{p,l,0}^{(\mu)}(i)$	$J - 1 + \frac{1}{J} \sum_{m=1}^{J} f_m^c$	0
$\lambda_{u,l,h}^{(\mu)}(i)$	0	$J(J-1)$
$\lambda_{p,l,p}^{(\mu)}(i)$	0	$\leq J + \sum_{m=1}^{J} f_m^r$
$\lambda_{p,l,0}^{(\mu)}(i)$	0	$\leq \frac{1}{J} \sum_{m=1}^{J} f_m^r$
Total (μ=1 à (M-1))	$\leq J^2 - 1 + \frac{(J+1)}{J} \sum_{n=1}^{J} f_n^c + \sum_{m=1}^{J} f_m^r + \frac{1}{J} \sum_{l=1}^{J} \sum_{n=1}^{f_l^r} f_n^c$	$\leq J^2 + \frac{(J+1)}{J} \sum_{m=1}^{J} f_m^r$

De plus, en considérant une pondération des équations $\psi_{u,l,n}^{(\mu)}(i)$ et $\psi_{p,l,n}^{(\mu)}(i)$, l et $n = 1, 2, \ldots, J$, par des coefficients $a_{u,l,n}^{(\mu)}$ et $a_{p,l,n}^{(\mu)}$, le nombre nécessaire de multiplications par itération est $2J^4$. Cependant, comme c'est le cas avec le code R-CSO^{2}C-SS, J=5 utilisé au chapitre 5, tous les coefficients $a_{u,l,n}^{(\mu)}$ sont identiques alors que tous les coefficients $a_{p,l,n}^{(\mu)}$ prennent la valeur 1. Par conséquent, avec ce code, le nombre de multiplications

par itérations se réduit à J^2. Finalement, la latence engendrée par le décodage des codes R-CSO^{2}C-SS dépend uniquement de la longueur de la mémoire du code qui est déterminé par $m = J \times \underset{1 \le k, n \le J}{max} \{\beta_{k,n}, \varepsilon_{l,n}\}$. Si le nombre d'itérations est M, alors la latence totale engendrée par tout le processus de décodage itératif devient mM bits.

6.2 Comparaisons avec d'autres techniques de décodage itératif avec entrelacement

Dans cette section, on analyse quelques comparaisons entre les codes CSO^{2}C présentés dans cet ouvrage et d'autres techniques de codage et de décodage Turbo. Ces comparaisons devraient permettre d'identifier les domaines d'applications les plus appropriés pour la technique de décodage itératif à seuil des codes CSO^{2}C. Ces comparaisons sont effectuées en fonction de la latence, la complexité et des performances d'erreur que produit chaque technique de décodage.

Les décodeurs constituants utilisés dans le processus itératif (Turbo) peuvent être réalisés selon différents algorithmes. Par exemple, les algorithmes de Bahl ou SOVA (Soft-in Soft-out Viterbi Algorithm) peuvent être utilisés comme décodeur élémentaire. D'autre part, le décodage à seuil peut être aussi considéré comme algorithme de base dans le décodage Turbo conventionnel [12]. Dans ce dernier cas, une concaténation parallèle de codeurs CSOC avec entrelacement est nécessaire.

6.2.1 Comparaisons entre le processus de décodage à seuil itératif sans entrelacement et le processus de décodage à seuil Turbo avec entrelacement

Le tableau 6.4 montre quelques comparaisons entre le décodage de codes CSO^2C-WS et le décodage Turbo d'un code utilisant une concaténation parallèle de deux codeurs CSOC [12]. Dans ces comparaisons, les deux codes CSO^2C-WS utilisés ont des valeurs de J=6 et J=7. Le lecteur trouvera les résultats de simulation de ces deux codes à l'annexe IV. Les deux codeurs CSOC constituant la concaténation parallèle sont de taux de codage $r_{c,i}$=2/3, i=1, 2 et ont une valeur de J=4 [12]. Le taux de codage global de la concaténation parallèle devient alors r_c=1/2. L'entrelaceur utilisé par les auteurs de [13] et [12] a une taille de 1000 bits et est de type aléatoire. À chaque itération, le bloc d'information courant doit être entièrement décodé par le premier décodeur de l'itération courante avant que le second décodeur de cette même itération puisse utiliser la version entrelacée des valeurs pondérées obtenues à la sortie du premier décodeur. Puisqu'un entrelaceur et un délaceur sont utilisés à chaque itération, la latence de chacune d'elle est au moins de 2000 bits.

Comme il a été mentionné par les auteurs de [13] et [12], la complexité de leur algorithme dépend aussi du nombre d'opérations «add-min» et «addition» que requiert chaque itération. Les auteurs de [13] et [12] déterminent le nombre d'opérations add-min requis par chaque décodeur d'une itération. Évidemment, ce nombre dépend aussi de la valeur de J. Lorsqu'un code CSOC, J=4, $r_{c,i}$=2/3, i=1, 2 est utilisé, ce nombre est égal à 28. Cependant, ce nombre d'opérations add-min déterminé par les auteurs de [13] et [12],

Tableau 6.4: Comparaisons entre un décodage Turbo CSOC avec entrelaceur et CSO^{2}C-WS

E_b/N_o =4 dB	Additions (par itération)	Add-min (par itération)	Multiplications (par itération)	Latence (bits/it.)	Latence totale (bits)
J=4 CSOC, $r_{c,i}$=2/3, i=1, 2 Entrelacement N=1000 bits, itération 6, $P_b(E)$=1.5x10^{-5}, r_c=1/2 (Résultats provenant de [12])	16	56	-----	2000	12000
J=6, CSO^{2}C-WS, itération 4, $P_b(E)$=3x10^{-5}	6	20	7	84	336
J=7, CSO^{2}C-WS, itération 4, $P_b(E) < 10^{-5}$	7	27	8	190	760

ne tient pas compte d'un décodage utilisant une rétroaction de la décision. En considérant un décodage avec rétroaction de la décision, ce nombre d'opérations «add-min» devrait augmenter. Puisque deux décodeurs sont nécessaires à chaque itération, le nombre d'opérations add-min requis par itération devient $2 \times 28 = 56$. Finalement, les comparaisons sont obtenues en fixant le rapport signal à bruit E_b/N_o à 4 dB. À ce rapport signal à bruit, une probabilité d'erreur d'environ 10^{-5} peut être atteinte avec ces deux types de techniques de décodage.

Comme on l'observe à partir des résultats montrés au tableau 6.4, la latence engendrée par le décodage itératif des codes CSO^{2}C-WS est beaucoup plus faible que celle engendrée par le décodage Turbo des codes CSOC utilisés en concaténation parallèle. Il faut aussi rappeler que la procédure de codage utilisant des codes CSO^{2}C étant de type

convolutionnel, celle-ci est insensible à la taille d'un bloc d'informations puisque le codage et la transmission des symboles codés peuvent se faire de manière continue. Contrairement au codage CSO^{2}C, le décodage Turbo de codes utilisant une concaténation parallèle de codeurs convolutionnels nécessite que les symboles d'information soient transmis par bloc dont la taille est généralement égale à celle des entrelaceurs.

Le tableau 6.5 montre d'autres comparaisons entre le décodage Turbo de codeurs CSOC en concaténation parallèle et le décodage itératif de codes CSO^{2}C-SS. Les codes constituant la concaténation parallèle sont les mêmes que précédemment. Seule la taille de l'entrelaceur est augmentée à une valeur de 9990 bits [12]. Le code CSO^{2}C-SS utilisé dans ces comparaisons a une valeur J=8. Les résultats de simulations pour ce code sont donnés à l'annexe V. Comme auparavant, les comparaisons sont faites en fixant la valeur de E_b/N_o =4 dB.

Tableau 6.5: Comparaisons entre un décodage Turbo CSOC avec entrelaceur et CSO^{2}C-SS

E_b/N_o =4 dB	Additions (par itération)	Add-min (par itération.)	Multiplications (par itération)	Latence (bits/it.)	Latence totale (bits.)
J=4 CSOC, $r_{c,i}$=2/3, i=1, 2 Entrelacement N=9990 bits, Itération 6, $P_b(E)$~10^{-6} r_c=1/2 (Résultats provenant de [12])	16	56	-----	19980	119880
J=8, CSO^{2}C-SS, Itération 3, $P_b(E)$=10^{-6}	8	35	-----	21280	63840

Les comparaisons montrées au tableau 6.5 indiquent que le nombre d'opérations par

bit et par itération nécessaire au décodage d'un code CSO^2C-SS est plus petit que celui requis pour un code utilisant une concaténation de deux codeurs CSOC, J=4. Quoique les probabilités d'erreur obtenues avec ces deux méthodes soient semblables (environ 10^{-6}), le code CSO^2C-SS, J=8 requiert un nombre inférieur d'itérations à celui nécessaire pour la concaténation parallèle de deux CSOC, J=4. Par conséquent, sachant que la mémoire du code CSO^2C-SS, J=8 est de $8 \times 2660 = 21280$ et que trois itérations sont nécessaires, la latence devient 63840 bits. D'autre part, la latence par itération engendrée par le décodage du code CSOC avec entrelacement est de $2 \times 9990 = 19980$ bits. Puisque six itérations sont nécessaires, la latence totale introduite par le processus de décodage itératif du code obtenu par la concaténation des deux codeurs CSOC, J=4 est de 119880 bits. Le décodage du code CSO^2C-SS, J=8 engendre donc une latence substantiellement inférieure à celle du décodage Turbo des codes CSOC considérés ici.

6.2.2 Comparaisons entre le processus de décodage à seuil itératif sans entrelacement et le décodage Turbo conventionnel

Une comparaison juste et équitable des performances des codes CSO^2C définis au sens large et au sens strict avec les performances de la technique de décodage Turbo conventionnel peut, dans certains cas, s'avérer difficile. Les difficultés se posent surtout au niveau des différences qu'il y a entre les types d'opérations utilisés par ces deux techniques de décodage. À l'origine, l'algorithme de Bahl (BCJR) [5], [4] utilisé dans le décodage Turbo détermine le MAP (maximum *a posteriori*) du bit à décoder. Cet algorithme MAP nécessite plusieurs calculs d'exponentielles et requiert plusieurs

opérations de multiplications et de divisions. On peut montrer que la complexité de l'algorithme de Bahl original a une complexité d'environ 3 à 4 fois celle du décodage de Viterbi [5]. Bien sûr, une réduction de cette complexité fût possible en transformant l'algorithme MAP dans un domaine logarithmique [6]. Cette modification a pour conséquence de transformer les multiplications en additions et les divisions en soustractions. Toutefois, pour résoudre le problème du calcul de sommes de fonctions exponentielles, des approximations sont faites en définissant une opération **E** comme suit [6]:

$$\ln(e^{-x} + e^{-y}) = min(x, y) - \ln(1 + e^{-|y-x|}) \cong min(x, y) = x \mathbf{E} y \qquad (6.1)$$

L'algorithme résultant de ces simplifications permet de déterminer le logarithme du rapport de vraisemblance (LRV) du bit d'information à décoder à un instant *i* donné. Cet algorithme appelé log-MAP est donc entièrement décrit par l'utilisation de l'opérateur **E** [6], [54]. Les résultats des calculs de la complexité de l'algorithme log-MAP ont déjà été obtenus par Osseiran [54]. Le tableau suivant résume ces résultats.

Tableau 6.6: Complexité de l'algorithme log-MAP [54]

Type d'opération	Nombre d'opérations par décodeur
Addition	$6 \times 2^m - 1$
Soustraction	$4 \times 2^m + 1$
Opération **E**	$5 \times 2^m - 2$

On observe que la complexité de l'algorithme log-MAP dépend essentiellement de la mémoire *m* de chaque codeur constituant qui est en général inférieure à 5. Elle peut aussi

dépendre de la taille de l'entrelaceur car l'entrelacement requiert des opérations de lecture ou d'écriture en mémoire [55]. Cependant, dans cette analyse, ces opérations de lecture et d'écriture ne sont pas considérées. Notons aussi que la complexité montrée au tableau 6.6 n'est valable que pour un seul décodeur utilisant l'algorithme log-MAP. Puisque deux décodeurs sont utilisés à chaque itération du décodage Turbo, cette complexité doit être multipliée par 2 afin d'obtenir la complexité d'une seule itération. Quant à la latence d'une itération de décodage Turbo conventionnel, elle dépend de la taille N, des entrelaceurs (délaceurs) et de l'algorithme de décodage. Chaque bloc transmis correspond à une séquence de symboles codés de longueur N. Les symboles codés appartenant à une séquence de longueur N sont continuellement transmis à travers le canal de transmission en utilisant, par exemple, une modulation BPSK. À chaque itération du décodage Turbo, les symboles codés reçus sont retardés d'un délai correspondant à la latence engendrée par les deux décodeurs et les entrelaceurs (ou délaceurs) avant d'être réutilisés à l'itération suivante. La latence introduite par chaque entrelaceur ou délaceur est alors de N bits. Puisque chaque décodeur d'une itération doit attendre que la séquence d'information soit complètement reçue avant de fournir sa première valeur de sortie, la latence totale d'une itération est donnée par:

$$L_{Turbo} = 4N \text{ bits/itération} \tag{6.2}$$

Au tableau 6.7, on compare la complexité et la latence de quelques codes CSO^2C-WS et CSO^2C-SS avec la complexité et la latence d'un système de codage et de décodage Turbo. Le codeur utilisé avec le décodage Turbo est formé d'une concaténation de deux

codeurs convolutionnels systématiques et récursifs (CPCCSR) de mémoire m=4 et dont l'entrelaceur de type aléatoire a une taille N=400 bits. En utilisant une perforation du code, le taux de codage devient r_c=1/2. Le nombre nécessaire d'opérations de chaque type est évalué en tenant compte du nombre requis d'itérations pour atteindre la convergence du processus de décodage Turbo. En utilisant le même algorithme de décodage log-MAP implanté par Osseiran [54], des résultats de simulation pour le code CPCCSR, m=4 et N=400 ont été obtenus. À l'aide de ces résultats de simulation, le nombre d'itérations nécessaire permettant à ce code d'atteindre la convergence a été évalué à cinq itérations. La latence exprimée en nombre de bits est aussi évaluée en fonction de ce nombre d'itérations.

Le code convolutionnel doublement orthogonal défini au sens large utilisé ici a une valeur J égale à neuf et utilise sept itérations avant d'atteindre la convergence. Un code convolutionnel doublement orthogonal défini au sens strict avec J=8 est aussi considéré dans cette comparaison. Pour chacun de ces deux codes CSO^2C, le nombre total de chaque type d'opérations et la latence totale sont aussi déterminés en fonction du nombre nécessaire d'itérations de décodage. La courbe de la Figure 6.1 permet de comparer ces trois méthodes de codage et de décodage uniquement en termes de leurs performances d'erreur.

Tableau 6.7: Comparaisons de la complexité et de la latence de quelques codes CSO^{2}C avec celles d'un code CPCCSR, m=4, N=400

	Addition (total)	Sous-traction (total)	Opération E (total)	addmin (total)	multiplication (total)	Latence totale (bits)
J=9, CSO^{2}C-WS 7 itérations	63	-----	------	308	7 Coefficients identiques	6394
J=8, CSO^{2}C-SS, 3 itérations (6 itérations)	24 (48)	-----	------	105 (210)	------ (aucun coefficient requis)	63840 (127680)
Turbo, m=4, N=400, 5 itérations, r_c=1/2	950	650	780	------	------	8000

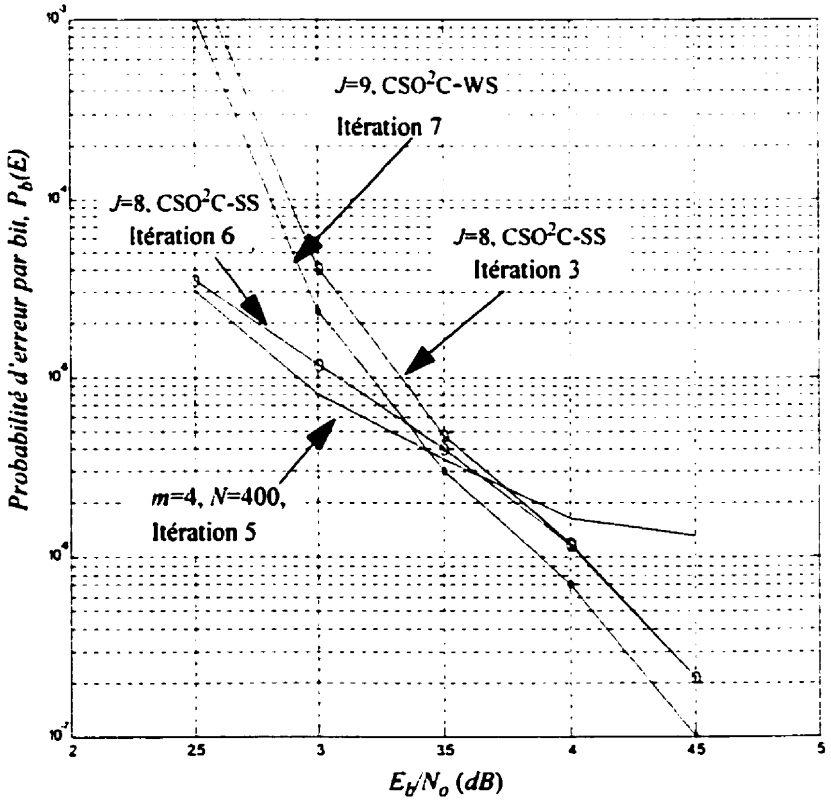


Figure 6.1 Comparaisons des performances d'erreur de quelques codes CSO^{2}C avec un code CPCCSR, m=4, N=400

Le jumelage des résultats présentés au tableau 6.7 avec les résultats de simulation illustrés à la Figure 6.1, nous permet de faire les observations suivantes. Tout d'abord, pour un rapport E_b/N_o supérieur à 3.3 dB, les courbes de la Figure 6.1 indiquent que le code CSO^{2}C-WS, J=9, fournit de meilleures performances d'erreur que le code CPCCSR m=4, N=400. De plus, selon les valeurs inscrites au tableau 6.7, on remarque que le décodage du code CSO^{2}C-WS, J=9 requiert un nombre d'opérations nettement inférieur à celui nécessaire pour le décodage du code CPCCSR utilisé dans ces comparaisons. On observe aussi qu'une latence de 6394 bits est introduite par le décodage (après sept itérations) du code CSO^{2}C-WS, J=9 alors que le décodage Turbo du code CPCCSR, m=4, N=400 introduit une latence de 8000 bits après cinq itérations. Même si le décodage du code CSO^{2}C-WS, J=9 nécessite sept itérations de décodage à seuil, la latence totale introduite par ce décodage itératif à seuil demeure inférieure à celle du décodage Turbo du code CPCCSR utilisé ici. La Figure 6.1 montre aussi une comparaison entre un code CSO^{2}C-SS, J=8 et le même code CPCCSR, m=4, N=400. À des rapports signal à bruit E_b/N_o inférieurs à 3.6 dB, les performances d'erreurs des deux codes sont proches. Plus précisément, on observe que le code CPCCSR procure un gain de codage supplémentaire d'environ 0.1 dB par rapport au code CSO^{2}C-SS, J=8 lorsque le nombre d'itérations est de 6. Cependant, pour de plus forts rapports signal à bruit, le code CSO^{2}C-SS, J=8 offre de meilleures performances d'erreur avec seulement trois itérations. Quoique le nombre nécessaire d'opérations au décodage du code CSO^{2}C-SS, J=8 soit inférieur à celui requis

par le décodage du code CPCCSR, le gain de codage que procure le code CSO^{2}C-SS, J=8 s'obtient au prix d'une augmentation de la latence. En effet, la latence introduite par le décodage du code CSO^{2}C-SS, J=8 est de beaucoup supérieure à celle engendrée par le décodage du code CPCCSR, m=4 et N=400.

La Figure 6.2 montre d'autres comparaisons de la latence des codes CSO^{2}C-WS et CSO^{2}C-SS pour différentes valeurs de J avec celle des codes CPCCSR utilisant différentes valeurs de N. Ces comparaisons sont obtenues en fixant le rapport signal à bruit E_b/N_o à une valeur de 3.5 dB. On observe sur cette figure que, pour une probabilité d'erreur de 3×10^{-6}, le décodage du code CSO^{2}C-WS, J=9 engendre une latence totale d'environ 7×10^3 bits alors que pour le décodage Turbo du code CPCCSR utilisant un entrelaceur de taille N=1024, la latence totale est de 2×10^4 bits. On remarque aussi de façon générale, que pour une même probabilité d'erreur, le décodage des codes CSO^{2}C-WS introduit une latence environ 2 à 3 fois plus petite que celle introduite par le décodage Turbo. Par contre, pour des codes CSO^{2}C-SS, la latence est supérieure à celle engendrée par le décodage Turbo.

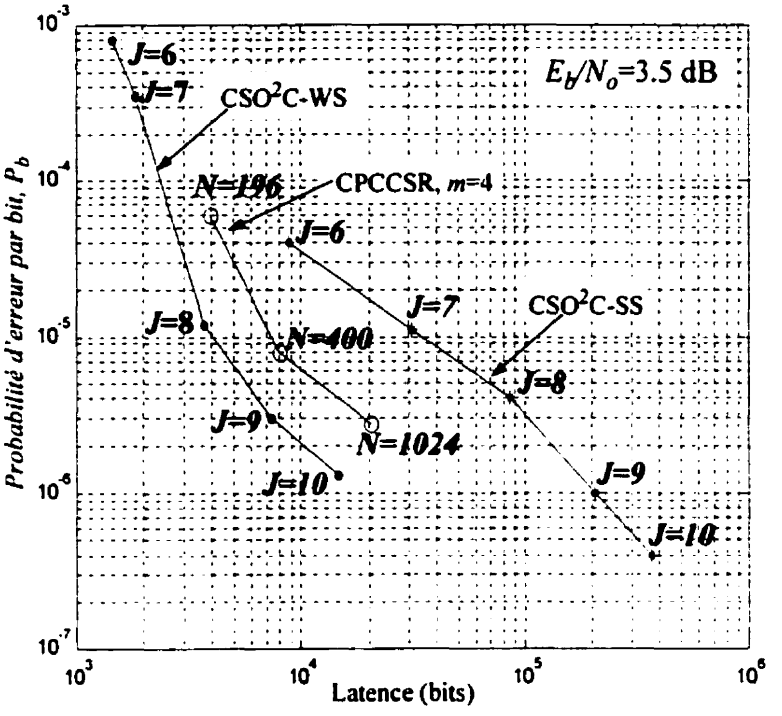


Figure 6.2 Comparaisons de la latence de quelques codes CSO^{2}C-WS et CSO^{2}C-SS avec celle des codes CPCCSR

À la suite de ces observations, il apparaît clairement qu'il existe des codes CSO^{2}C définis au sens large et au sens strict permettant d'atteindre de très bonnes performances d'erreur dépassant celles des codes CPCCSR utilisant une petite taille d'entrelacement. Toutefois, la plupart des codes CSO^{2}C étudiés dans cette thèse ne peuvent offrir ces performances d'erreur sur une plage étendue de rapports signal à bruit. On ne peut donc pas affirmer de façon catégorique que les codes CSO^{2}C-WS et CSO^{2}C-SS procurent de meilleures performances d'erreur que les codes CPCCSR avec une complexité et une latence réduite et ce, pour de grandes variations de E_b/N_o. En revanche, pour des

rapports E_b/N_o élevés, les codes CSO^{2}C-WS offrent généralement de très bonnes performances d'erreur avec une complexité et une latence faible.

Il peut être aussi intéressant de comparer les codes R-CSO^{2}C-SS avec les codes CPCCSR conventionnels. En se référant au tableau 6.3, on détermine la complexité en nombre d'opérations que nécessite le décodage itératif du code R-CSO^{2}C-SS, J=5 utilisé à la section 5.2.4 et dont la matrice $\boldsymbol{H}^T$ est :

$$\boldsymbol{H}^T = \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{E} \end{bmatrix} = \left[\begin{array}{c} \begin{bmatrix} 0 & 299 & 0 & 142 & 1135 \\ 0 & 320 & 146 & 107 & 794 \\ 0 & 679 & 5 & 977 & 172 \\ 87 & 333 & 0 & 106 & 310 \\ 0 & 418 & 60 & 123 & 99 \end{bmatrix} \\ \hline \begin{bmatrix} 0 & 945 & 0 & 0 & 92 \\ 1135 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1003 & 0 \\ 0 & 0 & 193 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}\right] \tag{6.3}$$

Comme il a été montré à la section 5.2.4, en utilisant ce code et l'algorithme de décodage itératif décrit à la section 5.2.4, une probabilité d'erreur de 10^{-5} peut être atteinte avec un rapport signal à bruit E_b/N_o un peu inférieur à 1.3 dB (environ 1.27 dB) et ce, après 30 itérations de décodage. Ce code R-CSO^{2}C-SS, J=5 peut donc être comparé à un code CPCCSR dont la taille de l'entrelaceur est élevée. Les résultats de simulation des codes CPCCSR obtenus par Berrou et *al.* et présentés dans [4] sont donc utilisés dans cette dernière comparaison. Ce code CPCCSR utilise un entrelaceur bloc ayant une taille de

256x256=65536 bits et une concaténation de deux codeurs convolutionnels systématiques et récursifs chacun de mémoire m=4. Pour atteindre la convergence, les auteurs de [4] montrent, à l'aide de résultats de simulation, que 18 itérations sont nécessaires. Avec ce nombre d'itérations, une probabilité d'erreur de 10^{-5} est possible avec un rapport signal à bruit E_b/N_o de seulement 0.7 dB. Notons que la limite théorique prédite par Shannon se situe à un rapport signal à bruit E_b/N_o = 0 dB. Cependant, avec ce même code CPCCSR, il est possible d'atteindre une probabilité d'erreur de 10^{-5} avec des rapports E_b/N_o se situant entre 1.0 et 1.7 dB lorsque seulement 3 à 6 itérations sont utilisées. La Figure 6.3 reproduit les courbes des performances d'erreur en fonction de E_b/N_o obtenues par Berrou et *al.* [4].

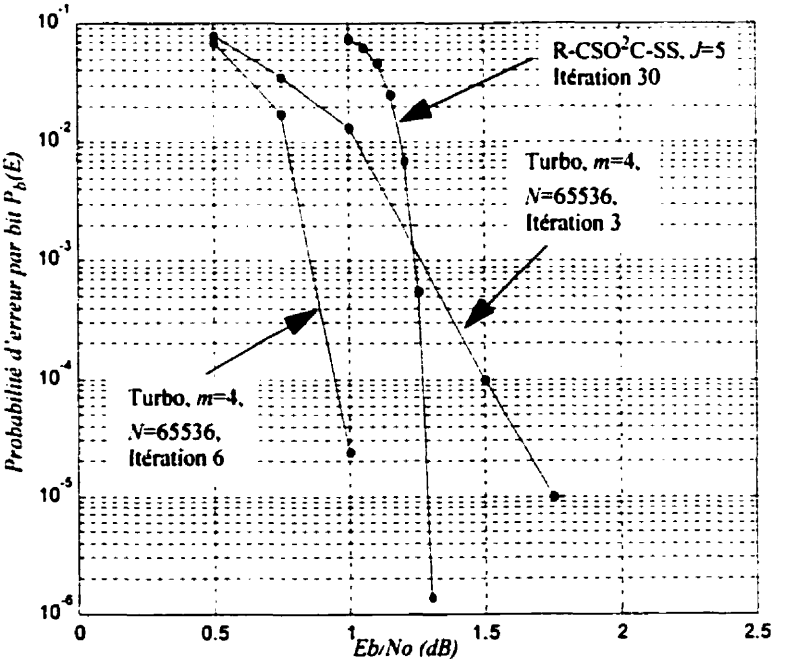


Figure 6.3 Comparaison du code CPCCSR, m=4, N=65536 avec le code R-CSO²C-SS, J=5

Sur cette même figure, on y montre aussi les résultats de simulation du code R-CSO^{2}C-SS, J=5 lorsque le nombre d'itérations égal à 30. Cette comparaison indique que pour un rapport E_b/N_o se situant autour de 1.3 dB, le code R-CSO^{2}C-SS, J=5 procure de meilleures performances d'erreur que le code CPCCSR proposé par Berrou et *al.* [4]. Afin de compléter cette comparaison, le tableau 6.8 indique le nombre de chaque type d'opérations effectuées par chacune des deux techniques de codage et de décodage considérées dans cette comparaison.

Tableau 6.8: Comparaisons de la complexités et de la latence du code R-CSO^{2}C-SS, J=5 avec celles d'un code Turbo m=4, N=65536

	Addition (total)	Sous-traction (total)	Opération E (total)	addmin (total)	multi-plication (total)	Latence totale (bits)
J=5, R-CSO^{2}C-SS, 30 itérations	930	-----	------	1320	150	170250
Turbo, m=4, N=65536, 3 itérations, r_c=1/2	570	390	468	------	------	786432

Selon les résultats des calculs de complexité montrés au tableau 6.8, il apparaît très clairement que la complexité du décodage itératif du code R-CSO^{2}C-SS, J=5 est plus élevé que celle du code CPCCSR. Toutefois, la présence des entrelaceurs et des délaceurs dans le processus de décodage de ce code CPCCSR engendre une latence beaucoup plus importante que celle introduite par le décodage itératif du code R-CSO^{2}C-SS, J=5. Il est important de préciser que la complexité réelle d'une réalisation matérielle d'un algorithme de décodage de type Turbo ou itératif dépend non seulement du nombre d'opérations

élémentaires mais, elle est aussi fortement déterminée par la taille de la mémoire nécessaire. La taille de la mémoire requise par le processus de décodage est donc proportionnelle à la latence engendrée par l'algorithme de décodage itératif. Il est alors évident, selon les résultats présentés dans cette comparaison que le code R-CSO^{2}C-SS, J=5 ne nécessite seulement que 21.6 % de la quantité de mémoire requise par le décodage itératif du code CPCCSR, m=4 et N=65636.

6.3 Conclusion

Pour conclure ce chapitre, l'utilisation des codes CSO^{2}C-WS semble particulièrement appropriées aux situations où le rapport E_b/N_o est relativement élevé. En effet, les comparaisons entre les différentes techniques de codage et de décodage itératif montrées dans ce chapitre font ressortir cette particularité des codes CSO^{2}C-WS. Pour de plus faibles rapports E_b/N_o, les codes CSO^{2}C-SS procurent des performances d'erreur proches de celles que procure un code CPCCSR où les deux techniques de décodage nécessitent environ le même nombre d'opérations à effectuer. Toutefois, la latence introduite par le décodage itératif des codes CSO^{2}C-SS devient importante devant celle que nécessite le décodage Turbo d'un code CPCCSR. Finalement, les codes R-CSO^{2}C-SS offrent de très bonnes performances d'erreur et sont comparables à des codes CPCCSR dont l'entrelaceur est de grande taille. Même si un grand nombre d'opérations et d'itérations est requis par le décodage itératif des codes R-CSO^{2}C-SS, la latence totale engendrée par le décodage de ces codes est de beaucoup inférieure à celle introduite par le

décodage Turbo des codes CPCCSR. Cette caractéristique devient un avantage lorsqu'un tel système de codage correcteur d'erreurs est utilisé dans des conditions de transmission où le délai devient un paramètre important à minimiser. Finalement, il importe de mentionner que ces comparaisons demeurent à des niveaux théoriques et donc les résultats obtenus dans ce chapitre peuvent varier lors d'une réalisation matérielle.

CHAPITRE 7: CONCLUSION

Cette thèse a pour but d'explorer des nouvelles techniques de codage et de décodage itératif efficaces afin de proposer des solutions simples aux problèmes de complexité et de latence du codage et du décodage Turbo. Ces solutions doivent satisfaire les contraintes d'indépendance entre les observables utilisés dans le processus de décodage itératif. Les solutions retenues sont donc basées essentiellement sur l'utilisation de l'algorithme de décodage à seuil itératif sans entrelacement et sur l'utilisation d'une nouvelle classe de codes : les codes convolutionnels doublement orthogonaux.

Ce but a été atteint en grande partie. En effet, les solutions proposées dans cette thèse permettent, dans plusieurs cas, des réductions substantielles de la complexité et de la latence des techniques conventionnelles de codage et de décodage itératif et ce, tout en maintenant des performances d'erreur acceptables. Bien sûr, nous ne pouvons pas prétendre que, pour toutes les conditions possibles de transmission, ces solutions proposées soient optimales en termes de complexité, de latence et de probabilité d'erreur. Toutefois, pour des canaux où le bruit est blanc, gaussien et additif, dans certaines plages de variations du rapport signal à bruit nous pouvons affirmer que les techniques de correction d'erreurs proposées dans cette thèse offrent de nombreux avantages comparativement aux techniques conventionnelles de codage et de décodage Turbo.

Le décodage Turbo conventionnel nécessite un algorithme de décodage symbole par symbole. Généralement, l'algorithme de décodage de Bahl est utilisé dans le processus de décodage itératif Turbo. D'autres algorithmes de décodage symbole par symbole peuvent

être utilisés. Parmi ceux-ci, le décodage à seuil offre une faible complexité. Cette dernière constatation, nous conduit au choix du décodage à seuil comme alternative aux autres techniques utilisées dans les processus conventionnels de décodage itératif Turbo. Avec la technique de décodage à seuil, les codes convolutionnels doivent être orthogonaux et systématiques.

L'entrelacement joue un rôle clé dans le décodage itératif Turbo conventionnel. Il assure une réduction de la corrélation des observables utilisés à chaque itération du processus de décodage. Cependant, la présence de l'entrelacement au codage et au décodage introduit une latence pouvant, dans certains cas, devenir importante. Dans le but de réduire la latence globale du processus de décodage itératif ainsi que la quantité de mémoire requise à la réalisation matérielle, la solution que nous proposons consiste à éliminer les entrelaceurs au codage et au décodage. Toutefois, le retrait de l'entrelacement doit se faire en respectant les conditions nécessaires qui assurent l'indépendance des observables utilisés dans le processus de décodage. Pour y arriver, des conditions d'orthogonalité supplémentaires sont alors imposées au code convolutionnel orthogonal traditionnel. En considérant deux itérations successives ces conditions ont été déterminées. L'analyse sur deux itérations consécutives du processus de décodage des codes convolutionnels orthogonaux, a conduit à la définition d'une nouvelle classe de codes convolutionnels pouvant être décodés itérativement par un décodage à seuil. Les codes convolutionnels alors obtenus ont été nommés «Code Convolutionnel Doublement Orthogonal» ou en anglais, Convolutional Self-Doubly Orthogonal Codes, CSO^2C. Nous avons pu définir deux types de codes CSO^2C : les codes CSO^2C définis au sens large

(CSO^{2}C-WS) et les codes CSO^{2}C définis au sens strict (CSO^{2}C-SS). Ces deux définitions conduisent à des structures différentes du codeur. Pour la définition au sens large des codes CSO^{2}C, une structure classique de codage convolutionnel est utilisée. Pour obtenir une définition au sens strict, une structure de codage parallèle a été nécessaire.

Avec des codes CSO^{2}C-WS, les conditions pour la double orthogonalité sont difficiles à satisfaire sans qu'il y ait une propagation d'erreur au décodage. Trois structures de décodage des codes CSO^{2}C-WS ont alors été définies. Chaque structure est liée aux conditions pour que le décodage puisse s'effectuer sur des observables indépendants. La première structure de décodage n'impose aucune condition sur l'utilisation des observables et donc le décodage s'effectue sur un ensemble d'observables dépendants. La deuxième structure impose des conditions de sorte que l'ensemble des observables dépendants soit réduit. La troisième structure utilise toutes les conditions pour que le décodage soit effectué sur un ensemble d'observables indépendants. Malheureusement, cette dernière structure a tendance à propager les erreurs. Des résultats de simulation ont montré que la deuxième structure de décodage apporte une amélioration marginale des performances d'erreur par rapport à la première structure de décodage. L'étude du décodage à seuil itératif des CSO^{2}C-WS a donc été basée sur la première stucture de décodage à seuil.

Pour améliorer davantage l'algorithme de décodage itératif à seuil des codes CSO^{2}C-WS, les équations d'inversion de parité de chaque itération sont pondérées par des coefficients. Ces coefficients sont déterminés de manière à minimiser la probabilité

d'erreur par bit à la dernière itération. Grâce à des résultats de simulation obtenus pour le décodage des codes CSO^{2}C-WS avec coefficients de pondération, il a été démontré que des améliorations substantielles des performances d'erreur par rapport à un décodage sans les coefficients de pondération sont possibles.

Les coefficients de pondération ont pour but de maintenir l'orthogonalité des équations d'inversion de parité dans tout le processus de décodage itératif. Ils servent aussi à conserver la fiabilité des observables à leur valeur maximale tout en assurant la convergence du processus de décodage itératif. Pour de forts rapports signal à bruit, les résultats de simulation ont montré qu'il n'est pas nécessaire d'utiliser des coefficients de pondération distincts pour obtenir des améliorations des performances d'erreur des codes CSO^{2}C-WS. Il suffit simplement d'assigner tous les coefficients de pondération utilisés dans tout le processus de décodage itératif à une valeur identique. Par la suite, cette valeur est déterminée de façon à minimiser la probabilité d'erreur à la dernière itération. Malheureusement, pour de faibles rapports signal à bruit, il n'est pas clair que l'utilisation de coefficients de pondération identiques soit le meilleur choix. Par conséquent, le problème de la détermination des coefficients de pondération pour de faibles rapports signal à bruit demeure encore un problème non-résolu.

À de forts rapports signal à bruit, une analyse de la sensibilité des performances d'erreur aux imprécisions des coefficients de pondération a été présentée. Dans cette analyse, les résultats obtenus par simulation montrent que le processus de décodage itératif à seuil appliqué à des codes CSO^{2}C-WS est peu sensible aux imprécisions des valeurs des coefficients de pondération. En effet, la dégradation des performances d'erreur

due à ces imprécisions est faible. Dans plusieurs cas analysés, cette dégradation est de l'ordre de 0.3 dB.

Une définition au sens strict des codes doublement orthogonaux a été obtenue en modifiant la structure du codeur. La structure parallèle du codeur permet d'éviter les répétitions indésirables d'observables pouvant se produire aux deux premières itérations. Ainsi, contrairement aux codes CSO^{2}C-WS, les codes CSO^{2}C-SS ne nécessitent pas de coefficients de pondération. De plus, à de forts rapports signal à bruit, seulement trois à quatre itérations sont requises pour atteindre la convergence du processus itératif. À de faibles rapports signal à bruit, les codes CSO^{2}C-SS, procurent de meilleures performances d'erreur que les codes CSO^{2}C-WS mais, au prix d'une augmentation de la latence.

Ainsi, selon une analyse des comparaisons entre les performances d'erreur des codes CSO^{2}C-WS et CSO^{2}C-SS, il ressort qu'en présence d'un canal de transmission très bruité, il est préférable d'utiliser un code CSO^{2}C-SS. D'autre part, pour un rapport signal à bruit fort, un code CSO^{2}C-WS procure de très bonnes performances d'erreur compte tenu de ses faibles complexité et latence.

Une analyse des performances d'erreur du décodage à seuil sans entrelacement a montré que, pour de forts rapports signal à bruit, une définition au sens strict de l'orthogonalité d'ordre deux suffit pour atteindre la performance limite du code donnée par le gain asymptotique de codage.

Une autre technique de codage correcteur d'erreur faisant appel à une définition de codes convolutionnels doublement orthogonaux récursifs, R-CSO^{2}C a été présentée. Ces

codes peuvent aussi être définis au sens large ainsi qu'au sens strict. Une caractéristique importante de ces codes est le fait que non seulement la fiabilité des symboles d'information peut être améliorée à chaque itération mais aussi, qu'il est possible d'obtenir à chaque itération, une amélioration de la fiabilité des symboles de parité. Selon les résultats de simulation obtenus dans cette thèse, les codes R-CSO^{2}C procurent de très bonnes performances d'erreur et ce, tout particulièrement avec des codes R-CSO^{2}C définis au sens strict (R-CSO^{2}C-SS). Ces performances d'erreur sont comparables à celles obtenues avec la technique de codage et de décodage Turbo conventionnel utilisant une grande taille d'entrelacement.

Finalement, quelques comparaisons entre la technique de décodage Turbo et les techniques de décodage itératif sans entrelacement ont été présentées en termes de leur complexité, de leur latence et de leurs performances d'erreur. Ces comparaisons révèlent que dans certaines plages des valeurs du rapport signal à bruit, le décodage itératif à seuil des codes CSO^{2}C-WS procurent des performances d'erreur semblables à celles obtenues avec le décodage Turbo des codes utilisant un concaténation parallèle de codeurs convolutionnels systématiques et récursifs (CPCCSR) ayant un entrelaceur de petite taille. Par exemple, pour une probabilité d'erreur de l'ordre de 10^{-6} et un rapport signal à bruit $E_b/N_o \geq 3.5$ dB, le décodage itératif d'un code CSO^{2}C-WS, J=9, introduit une latence environ 2.8 fois plus petite que celle introduite par le décodage Turbo d'un code CPCCSR de mémoire m=4 et utilisant un entrelaceur de taille N=1024. L'intérêt pour le décodage itératif à seuil des codes CSO^{2}C-WS réside surtout dans leur faible complexité et leur

faible latence comparativement à la complexité et à la latence du décodage Turbo des codes CPCCSR. Ces comparaisons montrent aussi que la latence des codes CSO^{2}C-SS est généralement supérieure à celle des codes Turbo. Par contre, à de faibles rapports signal à bruit, la complexité des codes CSO^{2}C-SS est nettement inférieure à celle des codes Turbo. La comparaison entre le décodage Turbo des codes CPCCSR et celui des codes R-CSO^{2}C-SS a montré que, même si un grand nombre d'opérations et d'itérations était requis par le décodage itératif des R-CSO^{2}C-SS, la latence totale engendrée par le décodage des codes R-CSO^{2}C-SS demeure inférieure à celle introduite par le décodage Turbo des codes CPCCSR. Cette caractéristique devient un atout important lorsqu'un tel système de codage correcteur d'erreurs est utilisé dans des conditions de transmission où le délai devient un paramètre important à minimiser.

En somme, ces travaux ont permis d'ouvrir de nouveaux sentiers qui n'avaient pas été explorés jusqu'à présent dans le domaine du codage et du décodage itératif. Plusieurs problèmes qui sont demeurés sans solution pourront être explorés dans des travaux de recherche futurs.

7.1 Recommandations pour recherches futures

Cette thèse porte sur une nouvelle technique de codage et de décodage itératif sans entrelacement. Les recherches présentées dans cette thèse ont permis entre autres choses, d'identifier un ensemble de problèmes qui restent encore à résoudre. D'autre part, des extensions de cette technique de codage et de décodage itératif sont toujours possibles. Les paragraphes suivants décrivent quelques-uns des problèmes et extensions qui

paraissent les plus importants.

Parmi les problèmes identifiés dans cette thèse, celui de l'ajustement des coefficients de pondération lorsque le canal de transmission est très bruité devrait retenir une attention particulière. Pour cela, il est nécessaire d'accroître la compréhension du rôle que jouent les coefficients de pondération dans le processus de décodage itératif à seuil. Il est utile de noter que, même pour des codes convolutionnels doublement orthogonaux définis au sens strict, un ajustement approprié des coefficients de pondération peut améliorer les performances d'erreur à de faibles rapports signal à bruit. Quoique la méthode d'ajustement des coefficients de pondération basée sur un réseau de neurones pourrait être améliorée, il est aussi probable, qu'une analyse théorique du processus de décodage itératif des codes convolutionnels doublement orthogonaux au sens large tenant compte de la corrélation entre les observables du processus soit nécessaire. Cette analyse théorique pourrait s'inspirer de celle proposée dans cette thèse pour des codes convolutionnels multiplement orthogonaux au sens strict. Notons aussi que l'ajustement approprié des coefficients de pondération est particulièrement profitable au décodage itératif des codes doublement orthogonaux récursifs définis au sens strict lorsque le rapport signal à bruit est faible. Une autre approche basée sur l'utilisation des graphes de Tanner pourrait conduire à une analyse précise des codes CSO^2C définis au sens large et au sens strict et de leur décodage itératif. Ces analyses permettraient de raffiner la procédure de décodage itératif proposée dans cette thèse.

Un autre problème d'intérêt consiste certainement à déterminer d'autres codes doublement orthogonaux récursifs au sens strict. Ces codes doivent avoir de bonnes

propriétés en ce qui concerne la partie récursive du code. En effet, une réduction de la densité de connexions provenant de la partie récursive de ces codes a permis d'obtenir des améliorations substantielles des performances d'erreur. Le problème qui se pose est alors de comprendre cet effet et de déduire les règles de génération de ces codes. Bien sûr, une analyse des performances devrait aussi être très utile afin de prévoir en fonction du rapport signal à bruit, le comportement de ces codes.

Une étude pourrait être entreprise afin d'identifier les applications qui pourraient tirer avantage des caractéristiques particulières de cette technique de codage et de décodage itératif. En outre, elle permetterait de déterminer les plages des valeurs de E_b/N_o et de la probabilité d'erreur où ces codes et ces techniques de décodage seraient les plus prometteurs. Cette étude demande d'établir un nombre important de comparaisons entre les différentes techniques de codage disponibles incluant celle présentée dans cette thèse. Par la suite, l'identification des applications serait immédiate.

Le codage allié à la modulation est généralement un moyen privilégié pour des transmissions où une grande efficacité spectrale est nécessaire. Il peut donc être intéressant d'explorer des méthodes permettant de combiner de façons astucieuses la technique de codage utilisant des codes convolutionnels doublement orthogonaux à la modulation numérique de grande efficacité spectrale. Le problème de déterminer les codes les plus appropriés ainsi que celui du décodage à seuil itératif lorsqu'une modulation codée est utilisée sont probablement les plus critiques.

Finalement, plusieurs autres problèmes pourraient être explorés. Par exemple, l'évaluation des performances d'erreur des codes convolutionnels doublement

orthogonaux utilisant un décodage itératif à seuil pour des canaux à évanouissements est aussi d'un grand intérêt pour la plupart des applications et pourrait faire l'objet de recherches futures.

RÉFÉRENCES

[1] SHANNON, C. E., "A Mathematical Theory of Communication", Bell System Technical Journal, Vol. 27, pp. 379-423, 1948.

[2] LIN, S., COSTELLO, D. J. JR., "Error-Control Coding: Fundamentals and Applications", Englewood Cliffs, NJ: Prentice-Hall, 1983.

[3] FORNEY, G. D. JR., "Concatenated Codes", MIT Press Cambridge, Mass., 1966.

[4] BERROU, C., GLAVIEUX, A., THITIMAJSHIMA, P., "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes", ICC 1993 Symposium, Geneva, pp. 1064-1070, mai 1993.

[5] BAHL, L. R., COCKE, J., JELINEK, F., RAVIV, J., "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate", IEEE Transaction on Information Theory, Vol. IT-20, No. 2, pp. 284-287, mars 1974.

[6] PIETROBON, S. S., "Implementation and Performance of a Serial MAP Decoder for use in an iterative turbo decoder", IEEE International Symposium on Information Theory, Whisler, Canada, p. 471, septembre 1995.

[7] PIETROBON, S. S., "Efficient Implementation of Continuous MAP Decoders and a Synchronisation Technique for Turbo Decoders", IEEE International Symposium of Information Theory and its Applications (ISITA 96), Victoria, British Colombia, Canada, Vol. 2 pp. 586-589, septembre 1996.

[8] FREY, B. J., KSCHISCHANG, F. R., "Early Detection and Trellis Splicing: Reduced-Complexity Iterative Decoding", IEEE Journal on Selected Areas in Communications, Vol. 16, pp. 153-159, février 1998.

[9] FRANZ V., ANDERSON, J. B., "Concatenated Decoding with a Reduced-Search BCJR algorithm", IEEE Journal on Selected Areas in Communications, Vol. 16, pp. 186-195, février 1998.

[10] HAGENAUER, J., PAPKE, L., "Decoding "Turbo"- Codes with the soft Output Viterbi Algorithm (SOVA)", IEEE International Symposium on Information Theory, Trondheim, Norway, p. 164, juillet 1994.

[11] BERROU, C., ADDE, P., ANGUI, E., FAUDEIL, S., "A Low Complexity Soft-Output Viterbi Decoder Architecture", ICC 1993, Geneva, Switzerland, pp. 737-740, mai 1993.

[12] RIEDEL, S., SVIRID, Y. V., "Iterative ("Turbo") Decoding of Threshold Decodable Codes", European Transactions on Telecomunications, Vol. 6, pp. 527-534, septembre - octobre 1995.

[13] SVRIRID, Y. V., RIEDEL, S., "Threshold Decoding of Turbo-Codes", Proceedings 1995 IEEE International Symposium on Information Theory, Whistler Conference Centre, Whistler, British Colombia, Canada, p. 39, september 1995.

[14] MASSEY, J. L., Threshold Decoding, Cambridge, Mass., MIT Press 1963.

[15] RHEE, M. Y., "Error Correcting Theory", New York, McGraw-Hill, 1989.

[16] BENEDETTO, S., MONTORSI, G., "Unveiling Turbo Codes: Some Results on Parallel Concatenated coding schemes", Transactions on Information Theory, Vol. IT-42, No. 2, pp. 429-445, mars 1996.

[17] BENEDETTO, S., MONTORSI, G., "Design of Parallel Convolutional Codes", IEEE Transactions on Communications, Vol. 44, No. 5, pp. 591-600, mai 1996.

[18] LAVOIE, P., HACCOUN, D., SAVARIA, Y., "New VLSI Architectures for Fast Soft-Decision Threshold Decoders", IEEE Transactions on Communications, Vol. 39, No. 2, pp. 200-207, février 1991.

[19] GAGNON, F., BATANI, N., DAM, T. Q., "Simplified Designs for AAPP Soft Decision Threshold Decoders", IEEE Transactions on Communications. Vol. 43, No. 2/3/4 pp. 743-750, février/mars/avril 1995.

[20] GALLAGER, R. G. "Low-Density Parity-Check Codes", IRE Transactions on Information Theory, Vol. IT-8, No. 1, pp. 21-28, janvier 1962.

[21] BAECHLER, B., "Génération de codes convolutionnels doublement orthogonaux", Mémoire de maîtrise, École Polytechnique de Montréal, avril 2000.

[22] BAECHLER, B., HACCOUN, D., GAGNON, F., "On the Search for Self-Doubly Orthogonal Codes", IEEE International Symposium on Information Theory, Sorrento Palace Hotel, Sorrento, Italie, p. 292, juin 2000.

[23] BHARGAVA, V., HACCOUN, D., MATYAS, R., NUSPL, P., "Digital Communications by Satellite: Multiple Access, Modulation and Coding", New York, John Wiley & Sons, Inc, 1981.

[24] WU, W. W., "New Convolutional Codes - Part I", IEEE Transactions on Communications, Vol. 23, No. 9, pp. 942-956, septembre. 1975.

[25] WU, W. W., "New Convolutional Codes - Part II", IEEE Transactions on Communications, Vol. 24, No. 9, pp. 946-955, septembre 1976.

[26] TANAKA, H., FURUSAWA, K., KANEKU, S., "A Novel Approach to Soft Decision Decoding of Threshold Decodable Codes", IEEE Transactions on Information Theory, Vol. IT-26, No. 2, pp. 244-246, mars 1980.

[27] HAGENAUER, J., "Soft is Better than Hard", Printed In Communication and Cryptography - Two Sides of ones Tapestry Edited by R. E. Blahut, D. J. Costello, jr., U. Maurer, T. Mittelholser, Kluver Academic Publishers, pp. 155-171, 1994.

[28] HAGENAUER, J., OFFER, E., PAPKE, L.,"Iterative Decoding of Binary Block and Convolutional Codes", Transactions on Information Theory, Vol. IT-42, No. 2, pp. 429-445, mars 1996.

[29] CLARK, G. C. JR., CAIN, J. B., "Error Correction Coding for Digital Communications", New York, Plenum Press, 1981.

[30] VITERBI, A. J., "An Intuitive Justification and a Simplified Implementation of the MAP Decoder for Convolutional Codes", IEEE Journal on Selected Areas in Communications, Vol. 16, pp. 260-264, février 1998.

[31] ROBERTSON, P., "Illuminating the Structure of Code and Decoder of Parallel Concatenated Recursive Systematic (Turbo) Codes", IEEE Globecom, pp. 1298-1303, 1994.

[32] BARBULESCU, S. A., "Iterative Decoding of Turbo Codes and Other Concatenated Codes", Ph.D. Dissertation, School of Electronic Engineering, Faculty of Engineering, University of South Australia, février 1996.

[33] GAGNON, F., HACCOUN, D., BATANI, N., CARDINAL, C., "Apparatus for Convolutional Self Doubly Orthogonal Encoding and Decoding", US Patent No. 6,167,552, 26 décembre 2000, et European Patent No. EP 0 907 256 A2, 2 avril 1999.

[34] CARDINAL, C., HACCOUN, D., GAGNON, F., BATANI, N.,"Convolutional Self Doubly Orthogonal Codes for Iterative Decoding Without Interleaving", Proceedings 1998 IEEE International Symposium on Information Theory, MIT, Cambridge, Mass. USA, p. 280, août 1998.

[35] CARDINAL, C., HACCOUN, D., GAGNON, F., BATANI, N., "Turbo Decoding Using Convolutional Self Doubly Orthogonal Codes", International Conference on Communication, ICC'99, Vancouver, British Colombia, Canada, pp. 113-117, juin 1999.

[36] CARDINAL, C., HACCOUN, D., GAGNON, F., BATANI, N., "Iterative Turbo Decoding Without Interleaving", Biennial Symposium on Communications, Proceedings, Kingston, Ontario, pp. 75-78, juin 1998.

[37] GAGNON, F., HACCOUN, D., "On the Construction of Codes for Iterative Decoding Based on an Extension of Difference Sets Principles", 13^{th} Applied Algebra, Algebraic Algorithms and Error Correcting Codes Symposium, 1999 AAECC-13, Honolulu, Hawaii, p. 17, novembre 1999.

[38] GAGNON, F., HACCOUN, D., "Convolutional Codes with Extended Self-Orthogonality and New Low-Density Parity-Check Block Codes", soumis à IEEE Transactions on Information Theory, juin 1999.

[39] SINGER, J., "A Theorem in Finite Projective Geometry and Some Applications to Number Theory", American Mathematic Society Transactions, Vol. 43, pp 377-385, 1938.

[40] ZURIDA, J. M., "Artificial Neural Nets", West, St-Paul, MN, 1992.

[41] HERAULT J., JUTTEN, C., "Réseaux neuronaux et traitement du signal", Traité des nouvelles technologies, série Traitement de signal, Hermès, 1994.

[42] MORRISEY, T. N. JR., "Analysis of Decoders for Convolutionnal Codes by Stochastic Sequential Machine Methods", IEEE Transactions on Information Theory, Vol. IT-16, No. 4, pp. 460-469, juillet 1970.

[43] FORNEY, G. D., "Convolutional Codes I: Algebraic Structure". IEEE Transactions on Informations Theory, Vol. IT-16, No. 6, pp. 720-738, novembre 1970.

[44] ROBINSON, J. P., "Definite decoding and error propagation of convolutionnal codes", IEEE Transactions on Information Theory, Vol. IT-14, No. 1, pp. 121-128, janvier 1968.

[45] MACKAY, D. J. C., NEAL, R. M., "Near Shannon limit performance of low density parity check codes", Electronic Letters, Vol. 32, pp. 1645-1646, août 1996, et Vol. 33, pp. 457-458, mars 1997.

[46] FOSSORIER, M. P. C., MIHALJEVIC, M., IMAI, H., "Reduced Complexity Iterative Decoding of Low Density Parity Check Codes Based on Belief Propagation", IEEE Transactions on Information Theory, Vol. IT-47, No. 3, pp. 673-680, mai 1999.

[47] LUCAS, R., FOSSORIER, M. P. C., KOU, Y., LIN, S., "Iterative Decoding of One-Step Majority Logic Decodable Codes Based on Belief Propagation", IEEE, Transactions on Communications, Vol. 48, No. 6, pp. 931-937, juin 2000.

[48] MCELIECE, R. J., MACKAY D. J. C., CHENG, J.-F., "Turbo Decoding as an Instance of Pearl's "Belief Propagation" Algorithm", IEEE, Journal on Selected Areas in Communications, Vol. 16, pp. 140-152, février 1998.

[49] KSCHISCHANG, F., FREY, B. J., "Iterative Decoding of Compound Codes by Probability Propagation in Graphical Models", IEEE, Journal on Selected Areas in Communications, Vol. 16, pp. 219-230, février 1998.

[50] MACKAY, D. J. C., "Good Error-Correcting Codes Based on Very Sparse Matrices", IEEE, Transactions on Information Theory, Vol. IT-45, No. 2, pp. 399-431, mars 1999.

[51] RICHARDSON, T., SHOKROLLAHI, A., URBANKE, R., "Design of Provably Good Low-Density Parity Check Codes", Proceedings 2000 IEEE International Symposium on Information Theory, Sorrento Palace Hotel, Sorrento, Italie, p. 199, juin 2000.

[52] KOORAPATY, H., WANG, Y.-P. E., BALACHANDRAN, K., "Performance of Turbo Codes with ShortFrame Sizes" IEEE Vehicular Technology Conference Proceeding of the 1997, 47th IEEE,Vehicular Technology Conference partie 1 de 3, Phoenix, AZ, USA, pp. 329-333, mai 1997.

[53] JUNG, P., "Comparison of Turbo-Code Decoders Applied to Short Frame Transmission Systems", IEEE Journal on Selected Areas in Communications, Vol. 14, pp. 530-537, avril, 1996.

[54] OSSEIRAN, A. H., "Sur le décodage des Codes Turbo", mémoire de maîtrise, École Polytechnique de Montréal, octobre 1999.

[55] LEANDERSON, C. F., EDFORS, O., MASENG, T., OTTOSSON, T., "On the Performance of Turbo Codes and Convolutional Codes of Low Rate", IEEE Vehicular Technology Conference, pp. 1560-1564, 1999.

[56] http://www2.elen.utah.edu/~schlegel/seminars/pearl.pdf

[57] CARDINAL, C., HACCOUN, D., GAGNON, F., "An Error Performance Analysis of Iterative Threshold Decoding", Proceedings 2000 IEEE International Symposium on Information Theory, Sorrento Palace Hotel, Sorrento, Italie, p. 187, juin 2000.

ANNEXE I : LOGARITHME DU RAPPORT DE VRAISEMBLANCE (LRV) DE L'ERREUR DE DÉCISION

Dans cette annexe, on montre que:

$$L(e_l^u \oplus \hat{e}_l^u) = -\left|L(u_l|\{B_{j,l}\})\right| \tag{I.1}$$

Par définition, on a que:

$$L(e_l^u \oplus \hat{e}_l^u) = \ln\left(\frac{P(e_l^u \oplus \hat{e}_l^u = 1)}{P(e_l^u \oplus \hat{e}_l^u = 0)}\right) = \ln\left(\frac{P(e_d = 1)}{P(e_d = 0)}\right) \tag{I.2}$$

où e_d représente l'erreur de décodage. On a aussi

$$\lambda_l = L(u_l|\{B_{j,l}\}) = \ln\left(\frac{P(u_l = 1|\{B_{j,l}\})}{P(u_l = 0|\{B_{j,l}\})}\right) \tag{I.3}$$

Une erreur de décodage ($e_d = 1$) se produit si $u_l \oplus \hat{u}_l = 1$, c'est-à-dire si $u_l \neq \hat{u}_l$. De la même manière, il n'y a pas d'erreur de décodage ($e_d = 0$) si $u_l \oplus \hat{u}_l = 0$, *i.e.* $u_l = \hat{u}_l$. Par conséquent, on peut écrire la relation suivante:

$$P(e_d = 1) = \begin{cases} P(u_l = 1|\{B_{j,l}\}), & \lambda_l \leq 0 \\ P(u_l = 0|\{B_{j,l}\}), & \lambda_l > 0 \end{cases} \tag{I.4}$$

Pour $\lambda_l \leq 0$ et à partir de la définition donnée par (I.3), on a que

$$P(u_l = 1|\{B_{j,l}\}) = e^{\lambda_l} P(u_l = 0|\{B_{j,l}\}) = e^{\lambda_l}(1 - P(u_l = 1|\{B_{j,l}\})) \tag{I.5}$$

$$= \frac{e^{\lambda_l}}{1 + e^{\lambda_l}} = P(e_d = 1)$$

Pour $\lambda_l > 0$ et toujours à partir de la définition donnée par (I.3), on trouve que:

$$e^{-\lambda_l} = \frac{P(u_l = 0 | \{B_{j,l}\})}{P(u_l = 1 | \{B_{j,l}\})} \tag{I.6}$$

d'où

$$P(u_l = 0 | \{B_{j,l}\}) = \frac{e^{-\lambda_l}}{1 + e^{-\lambda_l}} = P(e_d = 1) \tag{I.7}$$

Donc, en combinant I.5 avec I.7, on déduit que:

$$P(e_d = 1) = \frac{e^{-|\lambda_l|}}{1 + e^{-|\lambda_l|}} \tag{I.8}$$

La probabilité de ne pas avoir d'erreur de décodage $(e_d = 0)$ est:

$$P(e_d = 0) = 1 - \frac{e^{-|\lambda_l|}}{1 + e^{-|\lambda_l|}} = \frac{1}{1 + e^{-|\lambda_l|}} \tag{I.9}$$

On peut alors calculer:

$$\frac{P(e_d = 1)}{P(e_d = 0)} = \left(\frac{e^{-|\lambda_l|}}{1 + e^{-|\lambda_l|}}\right)(1 + e^{-|\lambda_l|}) \tag{I.10}$$

En prenant le logarithme de (I.10), on obtient:

$$\ln\left(\frac{P(e_d = 1)}{P(e_d = 0)}\right) = -|\lambda_l| = -|L(u_l | \{B_{j,l}\})| \tag{I.11}$$

C.Q.F.D.

ANNEXE II : DÉRIVATION DE L'OPÉRATEUR ADD-MIN

Dans cette annexe, on montre la dérivation de l'opérateur add-min à partir du calcul du logarithme du rapport de vraisemblance (LRV) de la somme modulo-2 de deux variables aléatoires binaires indépendantes ξ_1 et ξ_2. Le LRV de cette somme modulo-2 est donné par :

$$-L(\xi_1 \oplus \xi_2) = -\ln\left(\frac{P\{(\xi_1 \oplus \xi_2) = 1\}}{P\{(\xi_1 \oplus \xi_2) = 0\}}\right) = 2\,\text{atanh}\left(\tanh\left(\frac{-L(\xi_1)}{2}\right)\tanh\left(\frac{-L(\xi_2)}{2}\right)\right) \quad \text{(II.1)}$$

Cette démonstration se fait en deux étapes. Il sagit tout d'abord de déterminer le signe de la fonction $-L(\xi_1 \oplus \xi_2)$ et par la suite, de déterminer la valeur absolue $|L(\xi_1 \oplus \xi_2)|$, c'est-à-dire :

$$-L(\xi_1 \oplus \xi_2) = sign\{-L(\xi_1 \oplus \xi_2)\}|L(\xi_1 \oplus \xi_2)| \quad \text{(II.2)}$$

Puisque les fonctions $\text{atanh}(x)$ et $\tanh(y)$ sont des fonctions de symétrie impaire ($\text{atanh}(-x) = -\text{atanh}(x)$ et $\tanh(-y) = -\tanh(y)$), on a que :

$$sign\left\{2\,\text{atanh}\left(\tanh\left(\frac{-L(\xi_1)}{2}\right)\tanh\left(\frac{-L(\xi_2)}{2}\right)\right)\right\} = sign\left\{\tanh\left(\frac{-L(\xi_1)}{2}\right)\tanh\left(\frac{-L(\xi_2)}{2}\right)\right\}$$

$$= sign\{L(\xi_1)\}sign\{L(\xi_2)\} \quad \text{(II.3)}$$

D'autre part, on a que $|L(\xi_1 \oplus \xi_2)| = 2\left|\text{atanh}\left(\tanh\left(\frac{-L(\xi_1)}{2}\right)\tanh\left(\frac{-L(\xi_2)}{2}\right)\right)\right|$ et du fait de la symétrie impaire des fonctions $\text{atanh}(x)$ et $\tanh(y)$, on écrit :

$$2\left|\text{atanh}\left(\tanh\left(\frac{-L(\xi_1)}{2}\right)\tanh\left(\frac{-L(\xi_2)}{2}\right)\right)\right| = 2\,\text{atanh}\left(\tanh\left(\frac{|L(\xi_1)|}{2}\right)\tanh\left(\frac{|L(\xi_2)|}{2}\right)\right)$$

$$= \ln\left(\frac{1+\tanh\left(\frac{|L(\xi_1)|}{2}\right)\tanh\left(\frac{|L(\xi_2)|}{2}\right)}{1-\tanh\left(\frac{|L(\xi_1)|}{2}\right)\tanh\left(\frac{|L(\xi_2)|}{2}\right)}\right) \qquad \text{(II.4)}$$

Puisque $\tanh(z/2) = (e^{z}-1)/(e^{z}+1)$, on peut écrire pour (II.4), la relation suivante :

$$\ln\left(\frac{1+\frac{(e^{|L(\xi_1)|}-1)}{(e^{|L(\xi_1)|}+1)}\frac{(e^{|L(\xi_2)|}-1)}{(e^{|L(\xi_2)|}+1)}}{1-\frac{(e^{|L(\xi_1)|}-1)}{(e^{|L(\xi_1)|}+1)}\frac{(e^{|L(\xi_2)|}-1)}{(e^{|L(\xi_2)|}+1)}}\right) = \ln\left(\frac{1+e^{|L(\xi_1)|}e^{|L(\xi_2)|}}{e^{|L(\xi_1)|}+e^{|L(\xi_2)|}}\right)$$

$$= \ln\left(\frac{1+e^{-|L(\xi_1)|}e^{-|L(\xi_2)|}}{e^{-|L(\xi_1)|}+e^{-|L(\xi_2)|}}\right) \qquad \text{(II.5)}$$

En admettant l'hypothèse qu'à de forts rapports signal à bruit, les valeurs de $|L(\xi_1)|$ et $|L(\xi_2)|$ sont grandes, on a que $e^{-|L(\xi_1)|}e^{-|L(\xi_2)|}$ devient négligeable devant la valeur 1, ce qui permet d'écrire :

$$\ln\left(\frac{1+e^{-|L(\xi_1)|}e^{-|L(\xi_2)|}}{e^{-|L(\xi_1)|}+e^{-|L(\xi_2)|}}\right) \approx \ln\left(\frac{1}{e^{-|L(\xi_1)|}+e^{-|L(\xi_2)|}}\right) = -\ln(e^{-|L(\xi_1)|}+e^{-|L(\xi_2)|}) \qquad \text{(II.6)}$$

De plus, si $|L(\xi_1)| > |L(\xi_2)|$, une approximation de la somme des deux fonctions exponentielles est donnée par :

$$e^{-|L(\xi_1)|}+e^{-|L(\xi_2)|} \approx e^{-|L(\xi_1)|} \qquad \text{(II.7)}$$

et de la même manière, si $|L(\xi_2)| > |L(\xi_1)|$, on peut écrire :

$$e^{-|L(\xi_1)|}+e^{-|L(\xi_2)|} \approx e^{-|L(\xi_2)|} \qquad \text{(II.8)}$$

de sorte que :

$$e^{-|L(\xi_1)|} + e^{-|L(\xi_2)|} \approx e^{-min\{|L(\xi_1)|, |L(\xi_2)|\}} \tag{II.9}$$

En combinant (II.9) à (II.6), on trouve :

$$\ln\left(\frac{1 + e^{-|L(\xi_1)|} e^{-|L(\xi_2)|}}{e^{-|L(\xi_1)|} + e^{-|L(\xi_2)|}}\right) \approx -\ln(e^{-|L(\xi_1)|} + e^{-|L(\xi_2)|}) \approx -\ln(e^{-min\{|L(\xi_1)|, |L(\xi_2)|\}})$$

$$\approx min\{|L(\xi_1)|, |L(\xi_2)|\} \tag{II.10}$$

Ainsi, on obtient la relation suivante :

$$|L(\xi_1 \oplus \xi_2)| = 2\left|\text{atanh}\left(\tanh\left(\frac{-L(\xi_1)}{2}\right)\tanh\left(\frac{-L(\xi_2)}{2}\right)\right)\right| \approx min\{|L(\xi_1)|, |L(\xi_2)|\} \tag{II.11}$$

de sorte qu'en combinant cette dernière relation avec (II.3) et (II.2), on détermine l'approximation suivante :

$$-L(\xi_1 \oplus \xi_2) = sign\{-L(\xi_1 \oplus \xi_2)\}|L(\xi_1 \oplus \xi_2)|$$

$$= sign\{L(\xi_1)\}sign\{L(\xi_2)\}min\{|L(\xi_1)|, |L(\xi_2)|\} \tag{II.12}$$

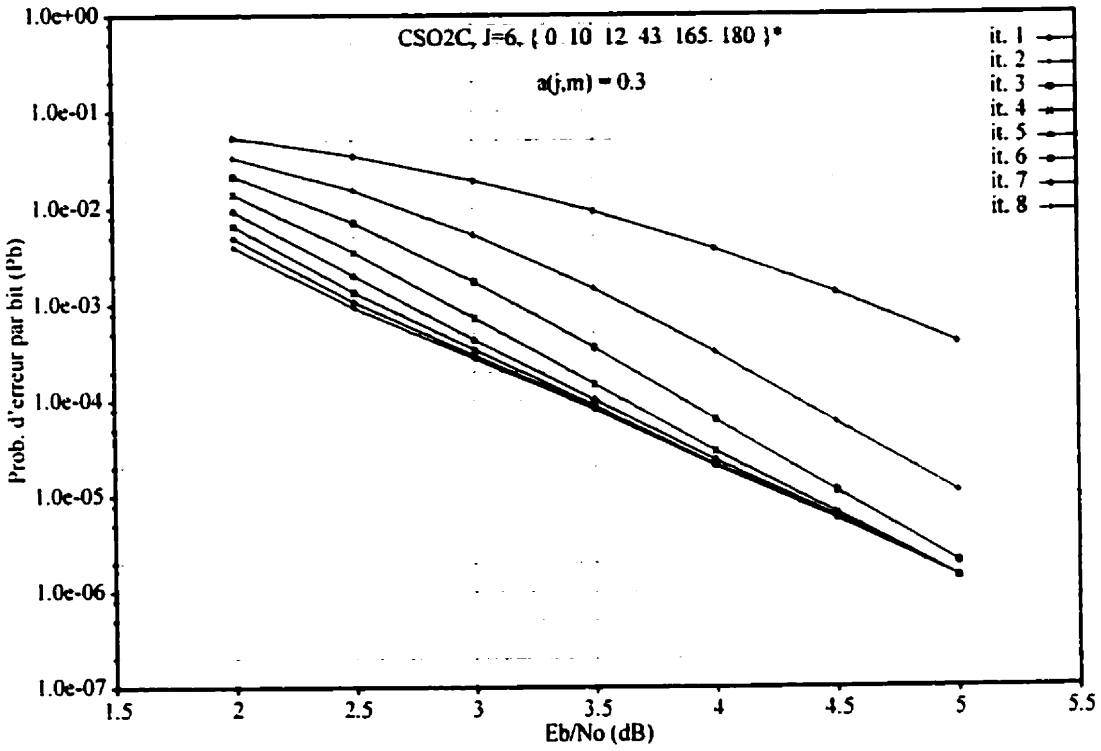
Finalement, on en déduit aussi la définition de l'opérateur addmin comme suit :

$$L(\xi_1 \oplus \xi_2) \approx L(\xi_1) \Diamond L(\xi_2) = -sign\{L(\xi_1)\}sign\{L(\xi_2)\}min\{|L(\xi_1)|, |L(\xi_2)|\}$$

(II.13)

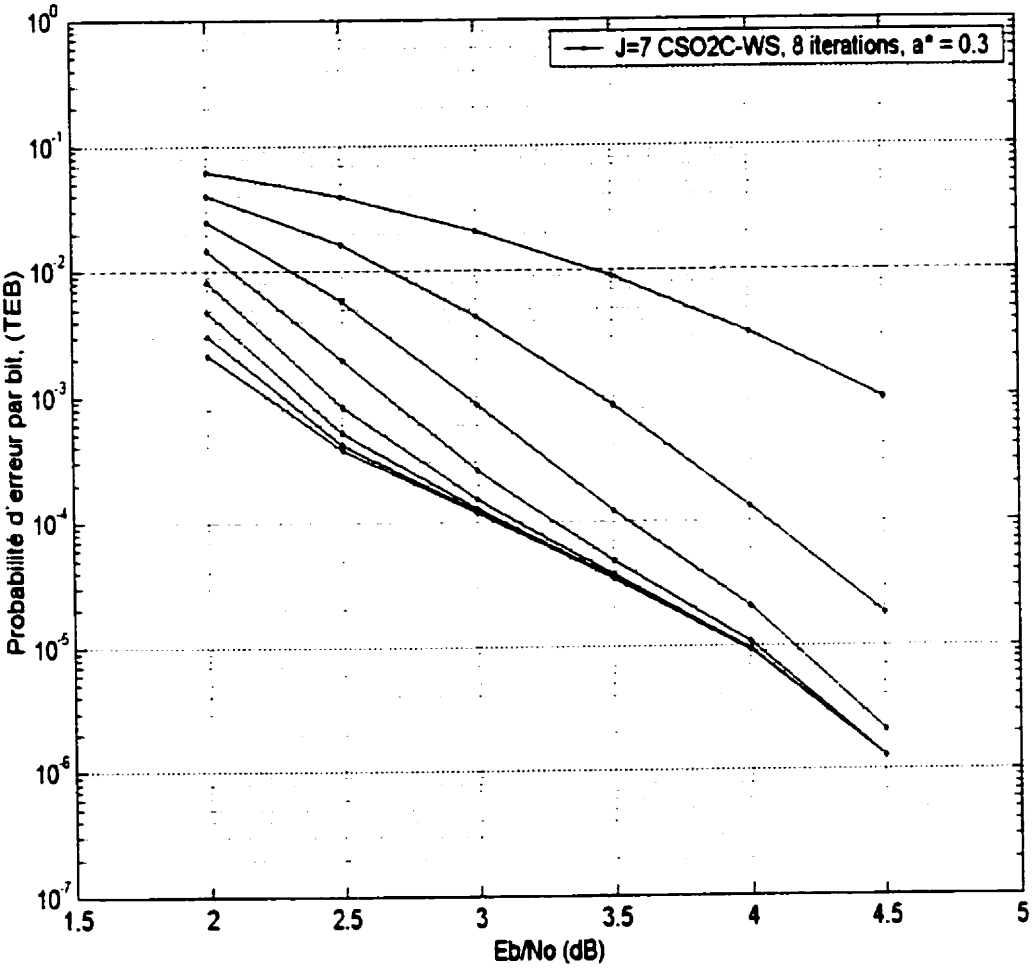
ANNEXE III : RÉSULTATS DE SIMULATION - MÉTHODE EXPÉRIMENTALE

$J = 6$, CSO^{2}C-WS, $r_c = 1/2$

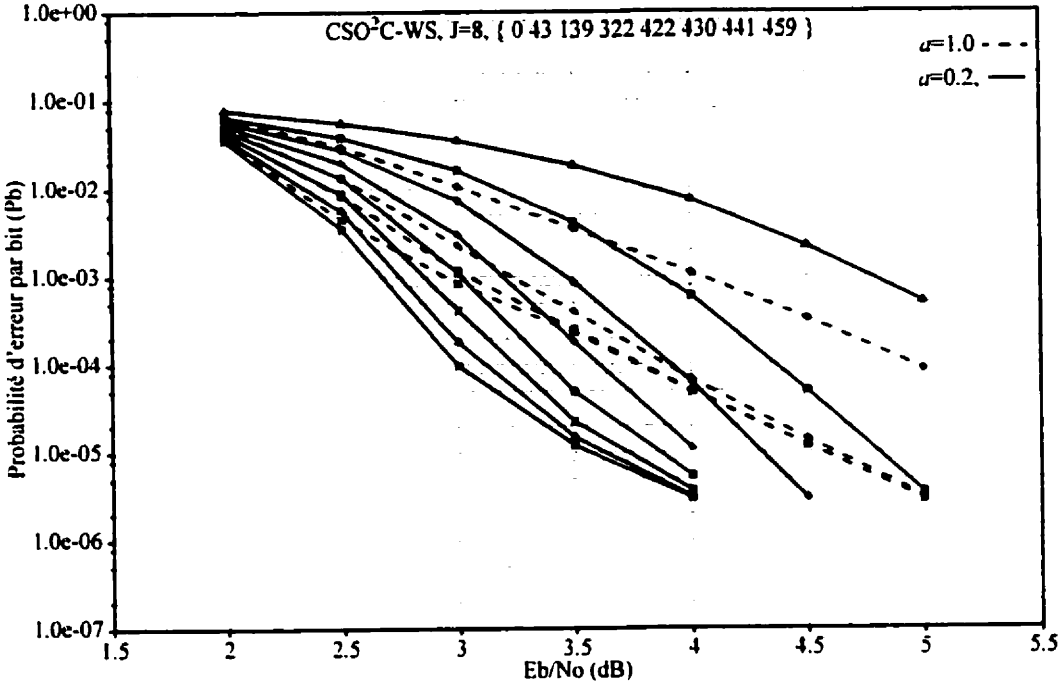


* Code non optimisé.

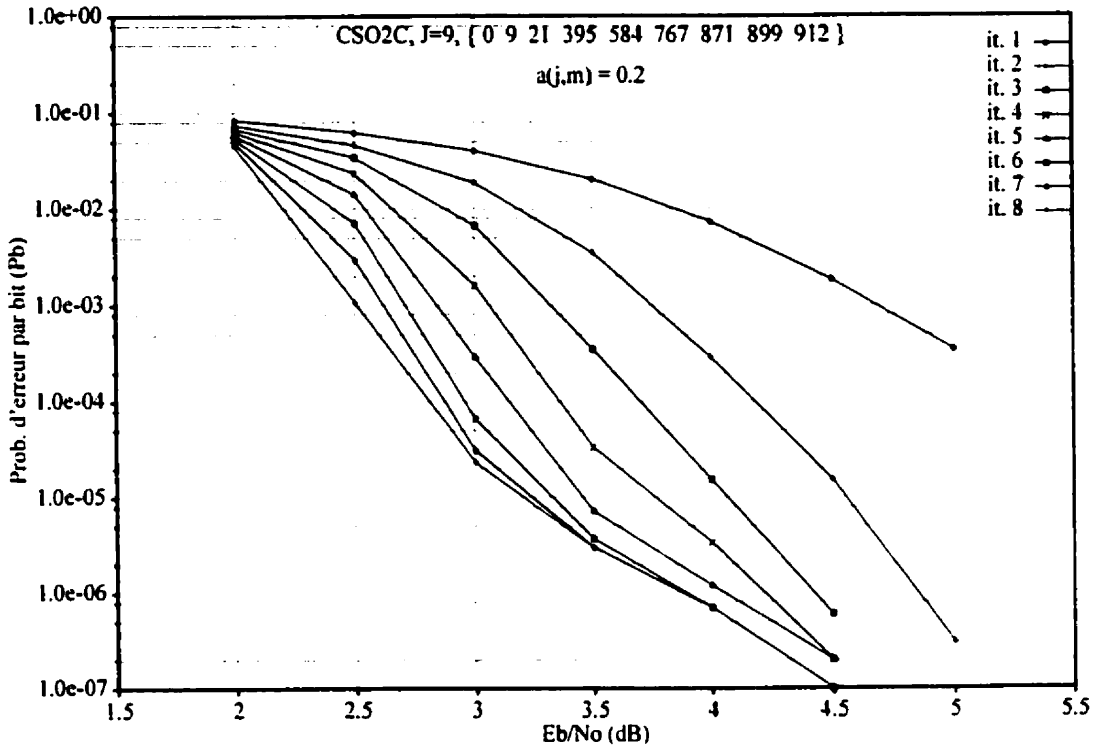
$J = 7$, CSO^{2}C-WS, $r_c = 1/2$, $a^* = 0.3$



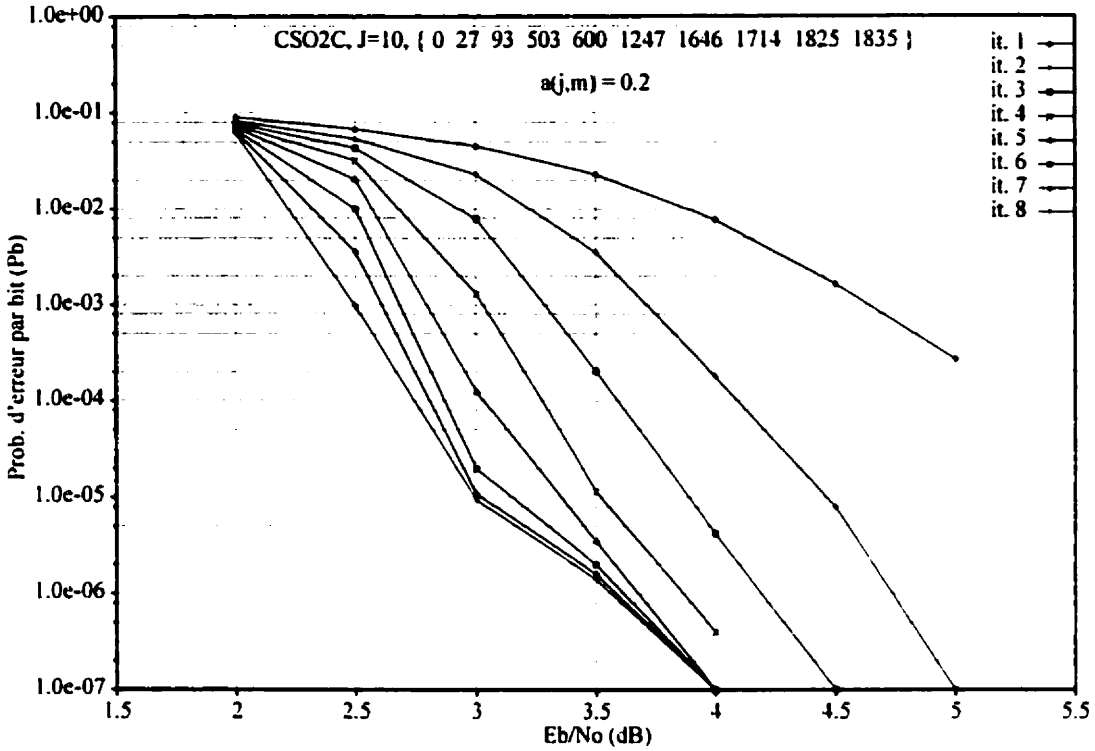
$J = 8$, CSO^{2}C-WS, $r_c = 1/2$



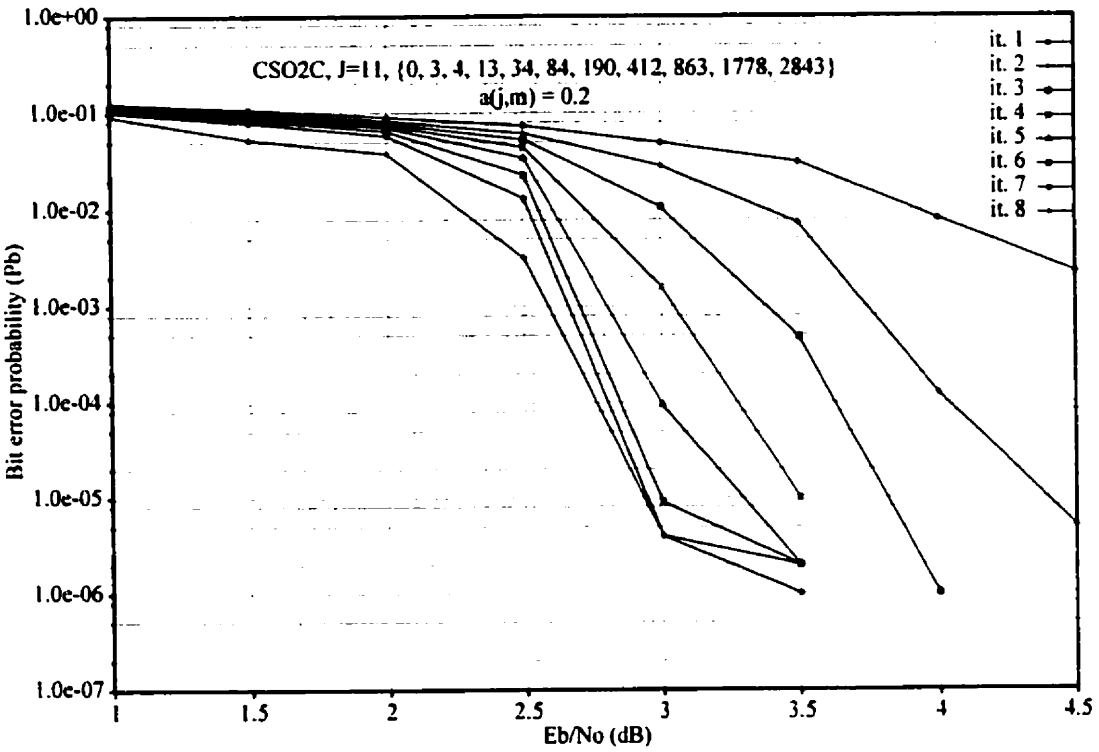
$J = 9$, CSO^{2}C-WS, $r_c = 1/2$



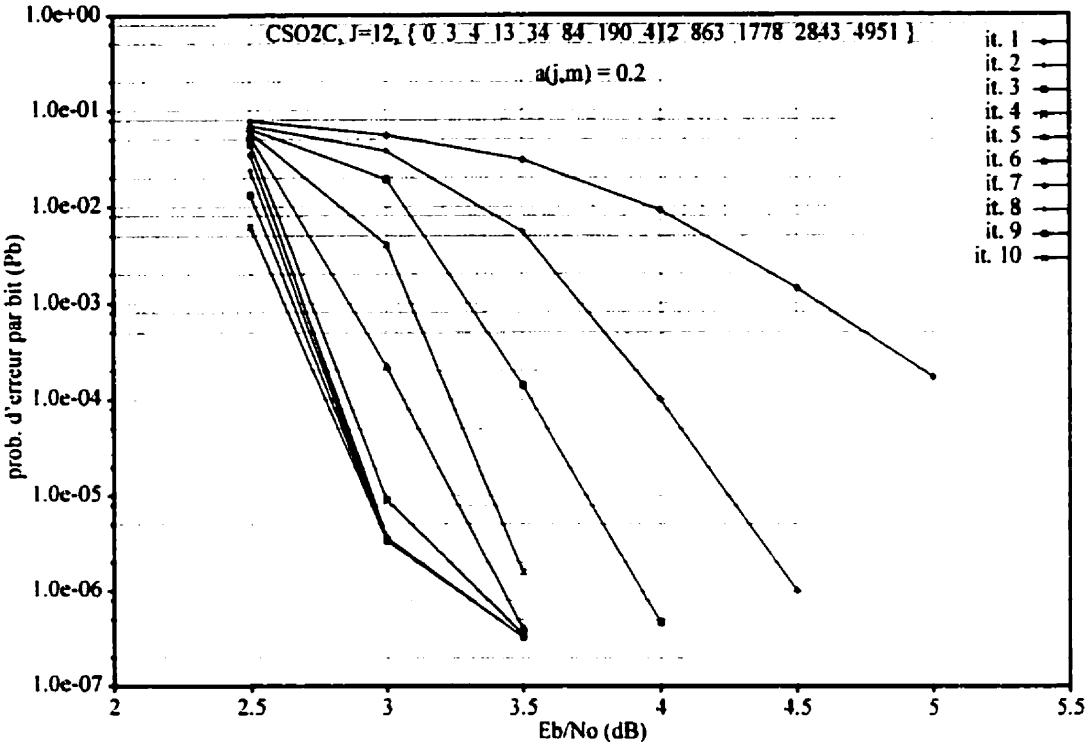
$J = 10$, CSO^{2}C-WS, $r_c = 1/2$



$J = 11$, CSO^{2}C-WS, $r_c = 1/2$



$J = 12$, CSO^{2}C-WS, $r_c = 1/2$

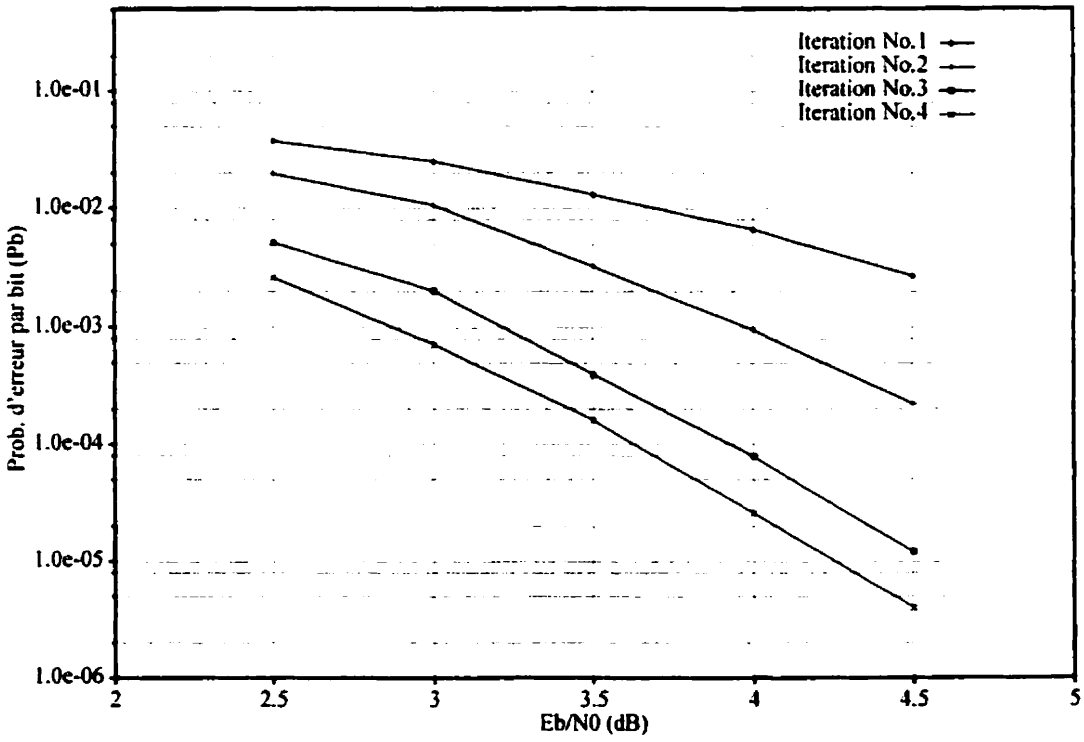


ANNEXE IV : RÉSULTATS DE SIMULATION - OPTIMISATION PAR RESEAUX DE NEURONES

J=6, CSO^{2}C-WS, r_c = 1/2, $\{\alpha_j\}$ = {0, 3, 4, 13, 34, 84}

Tableau III.1 : Coefficient de pondération opt. *J*=6, 4 itérations opt. à Eb/No=3.5 dB

	(information)	1	2	3	4	5	6
1	0.611658	0.297874	0.310055	0.359455	0.125683	0.279577	0.264237
2	0.454075	0.126263	0.203434	0.258132	0.401107	0.407700	0.423547
3	0.568132	0.597801	0.644026	0.550458	0.524248	0.410059	0.407618
4	0.107019 (1.0*)	0.271714 (1.0*)	0.257268 (1.0*)	0.238890 (1.0*)	0.223646 (1.0*)	0.199134 (1.0*)	0.12108 (1.0*)

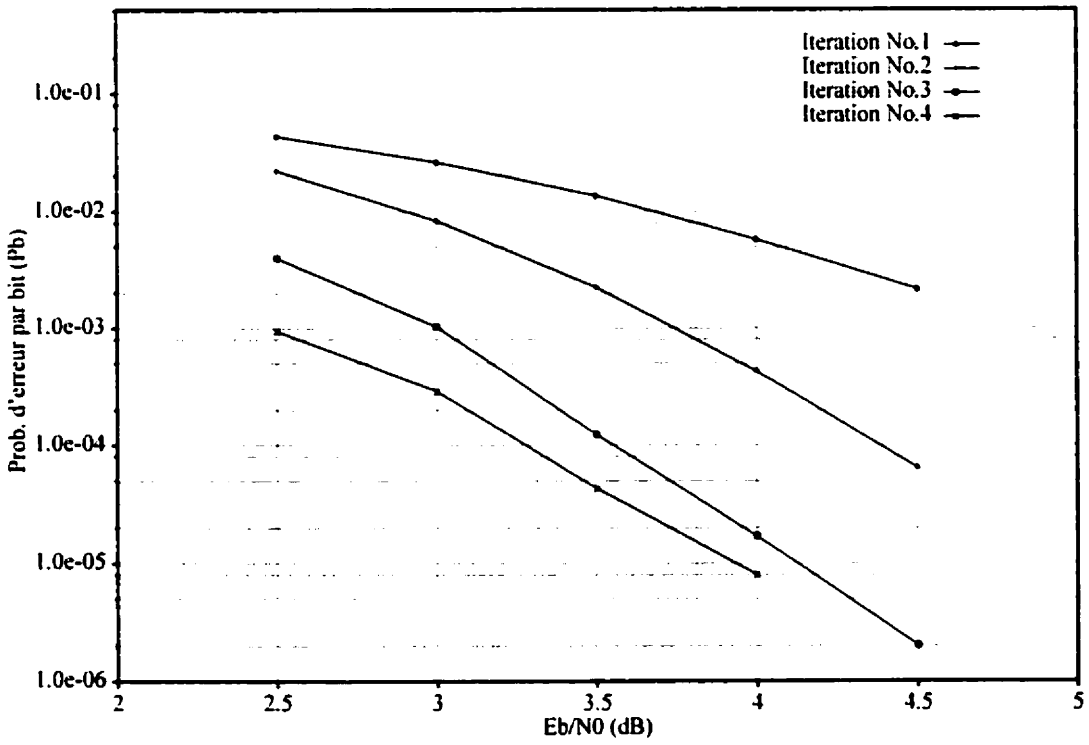


(*) Meilleurs résultats avec $a_j^{(4)} = 1.0$ pour $j = 0, ..., 6$

$J = 7$, CSO^{2}C-WS, $r_c = 1/2$, $\{\alpha_j\} = \{0, 3, 4, 13, 34, 84, 190\}$

Tableau III.2 : J=7, 4 itérations, opt. à Eb/No = 3.5 dB

	Information	1	2	3	4	5	6	7
1	0.600583	0.291193	0.303639	0.350103	0.129586	0.284449	0.253050	0.283001
2	0.451732	0.129399	0.186164	0.263750	0.390898	0.389496	0.430695	0.417326
3	0.590103	0.595138	0.636389	0.544530	0.517506	0.416599	0.395008	0.420053
4	0.089391 (1.0*)	0.230538 (1.0*)	0.224615 (1.0*)	0.213264 (1.0*)	0.193684 (1.0*)	0.179643 (1.0*)	0.154219 (1.0*)	0.088731 (1.0 *)

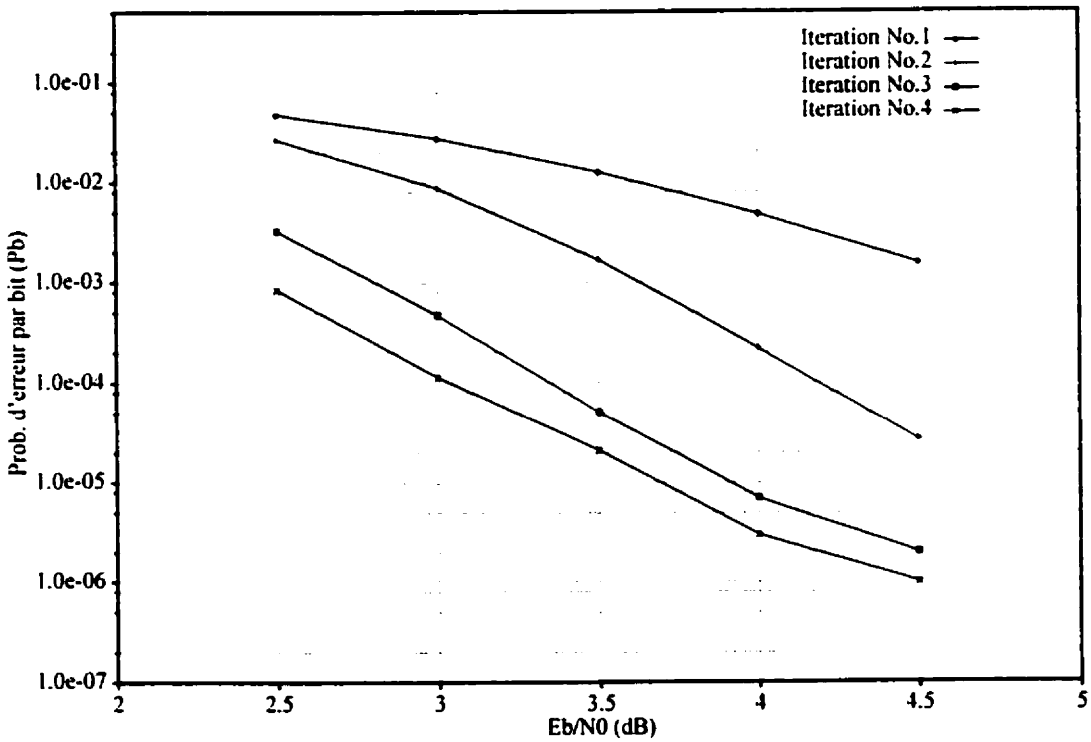


(*) Meilleurs résultats avec $a_j^{(4)} = 1.0$ pour $j = 0, ..., 7$

$J = 8$, CSO^{2}C-WS, $r_c = 1/2$, $\{\alpha_j\} = \{0, 3, 4, 13, 34, 84, 190, 412\}$

Tableau III.3 : J=8, 4 Itérations, opt. à Eb/No = 3.0 dB

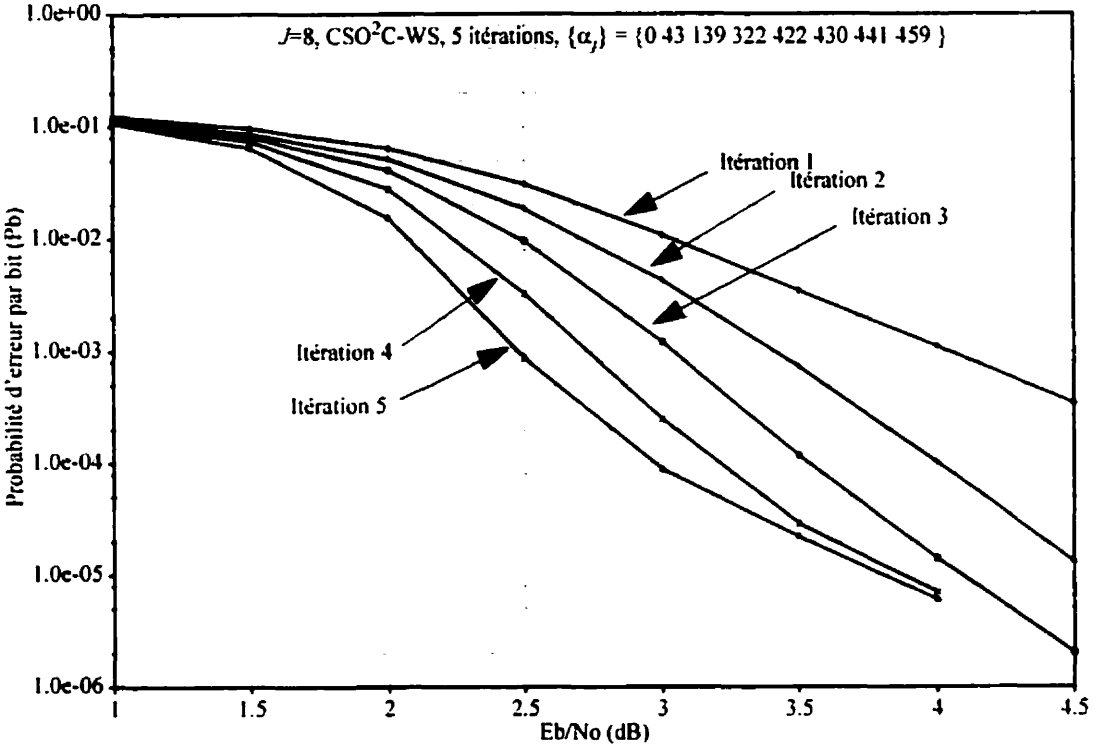
	Information	1	2	3	4	5	6	7	8
1	0.594187	0.292801	0.305418	0.354552	0.156604	0.298056	0.279119	0.292044	0.300087
2	0.318472	0.011823	0.108460	0.199745	0.362150	0.381525	0.428870	0.416052	0.424801
3	0.359054	0.667599	0.667656	0.560136	0.521223	0.398466	0.362635	0.375661	0.375341
4	0.076929 (1.0*)	0.203106 (1.0*)	0.202864 (1.0*)	0.193423 (1.0*)	0.185965 (1.0*)	0.178001 (1.0*)	0.165111 (1.0*)	0.148543 (1.0*)	0.085999 (1.0*)



(*) Meilleurs résultats avec $a_j^{(4)} = 1.0$ pour $j = 0, ..., 8$

$J = 8$, CSO^{2}C-WS, $r_c = 1/2$, $\{\alpha_j\} = \{0\ 43\ 139\ 322\ 422\ 430\ 441\ 459\}$

Note : pour ce code, l'optimisation par réseaux de neurones a été fait pour chaque itération montrée sur cette figure.



Coefficients à chaque itération :

Tableau III.4 : Coefficients de pondération pour deux itérations

	Information	1	2	3	4	5	6	7	8
1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
2	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Tableau III.5 : Coefficients de pondération pour trois itérations

	Information	1	2	3	4	5	6	7	8
1	0.20671	0.312530	0.279532	0.306356	0.110885	0.226763	0.222904	0.220284	0.216515
2	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
3	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

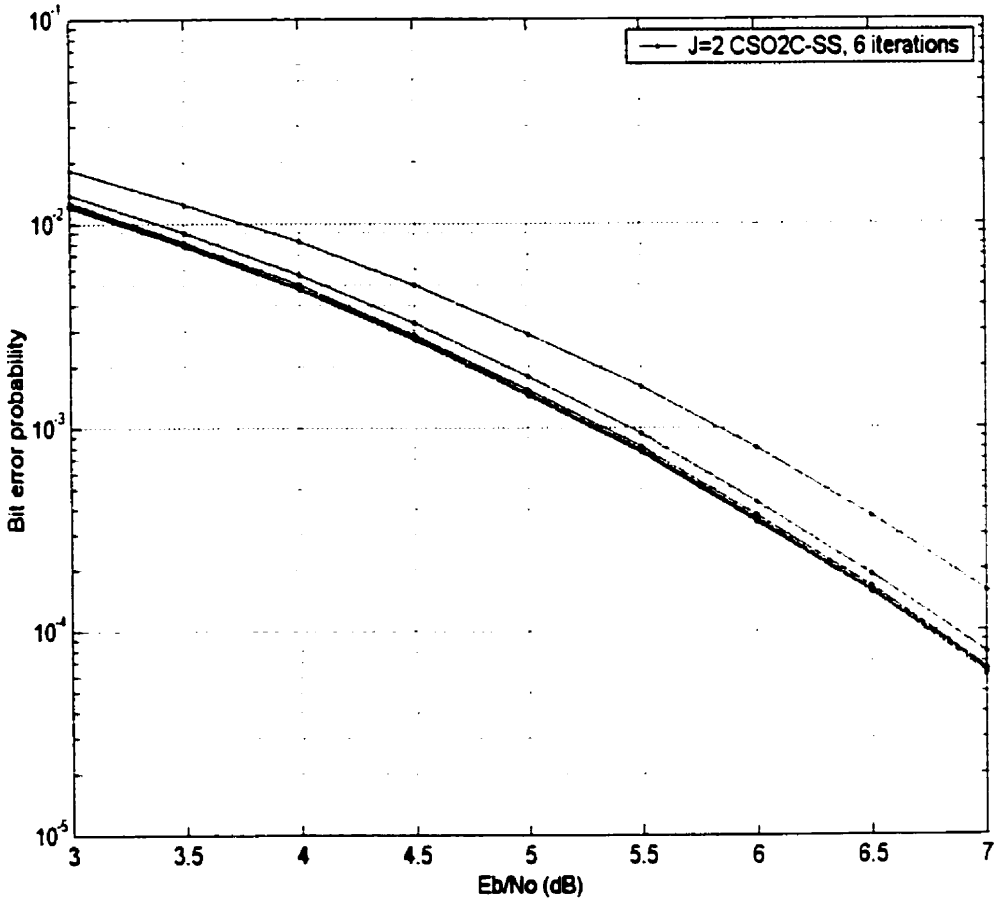
Tableau III.6 : Coefficients de pondération pour quatre itérations

	Information	1	2	3	4	5	6	7	8
1	0.338091	0.245692	0 226085	0 276203	0 048882	0 206264	0 209879	0 212754	0 216549
2	0.132884	0 286648	0 255909	0 233059	0 267659	0 267074	0 262635	0 263040	0 265196
3	1.0	1 0	1 0	1.0	1.0	1.0	1 0	1 0	1.0
4	1.0	1 0	1 0	1 0	1.0	1 0	1 0	1 0	1 0

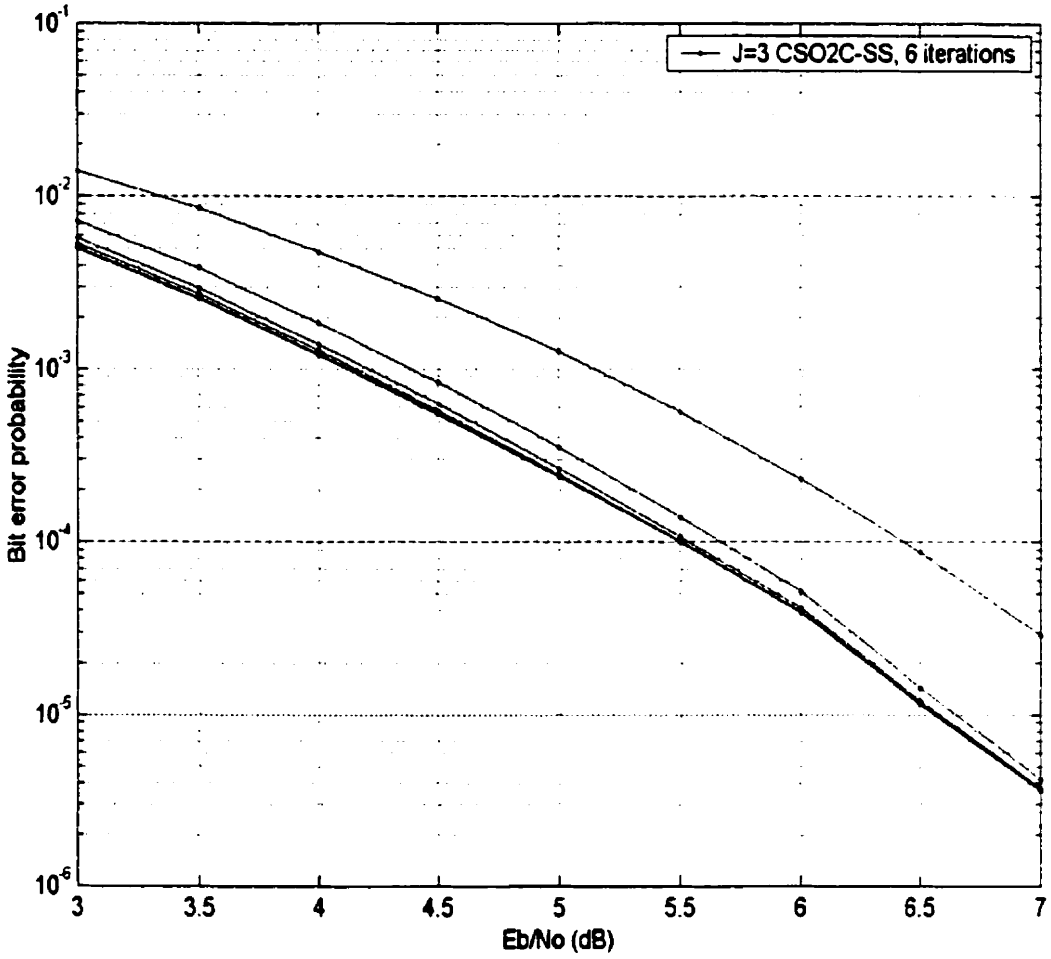
Tableau III.7 : Coefficients de pondération pour cinq itérations

	Information	1	2	3	4	5	6	7	8
1	0 398789	0.277387	0 247278	0 293361	0 055445	0 214385	0 216243	0 217340	0 218382
2	0 140025	0 274598	0 244374	0 228919	0 266262	0 276888	0 277108	0 278195	0 0 279358
3	0.122809	0.259570	0 226189	0 208575	0 245274	0 256188	0 256905	0 258776	0 260965
4	1 0	1 0	1 0	1 0	1 0	1 0	1 0	1 0	1 0
5	1 0	1 0	1 0	1 0	1 0	1 0	1 0	1 0	1 0

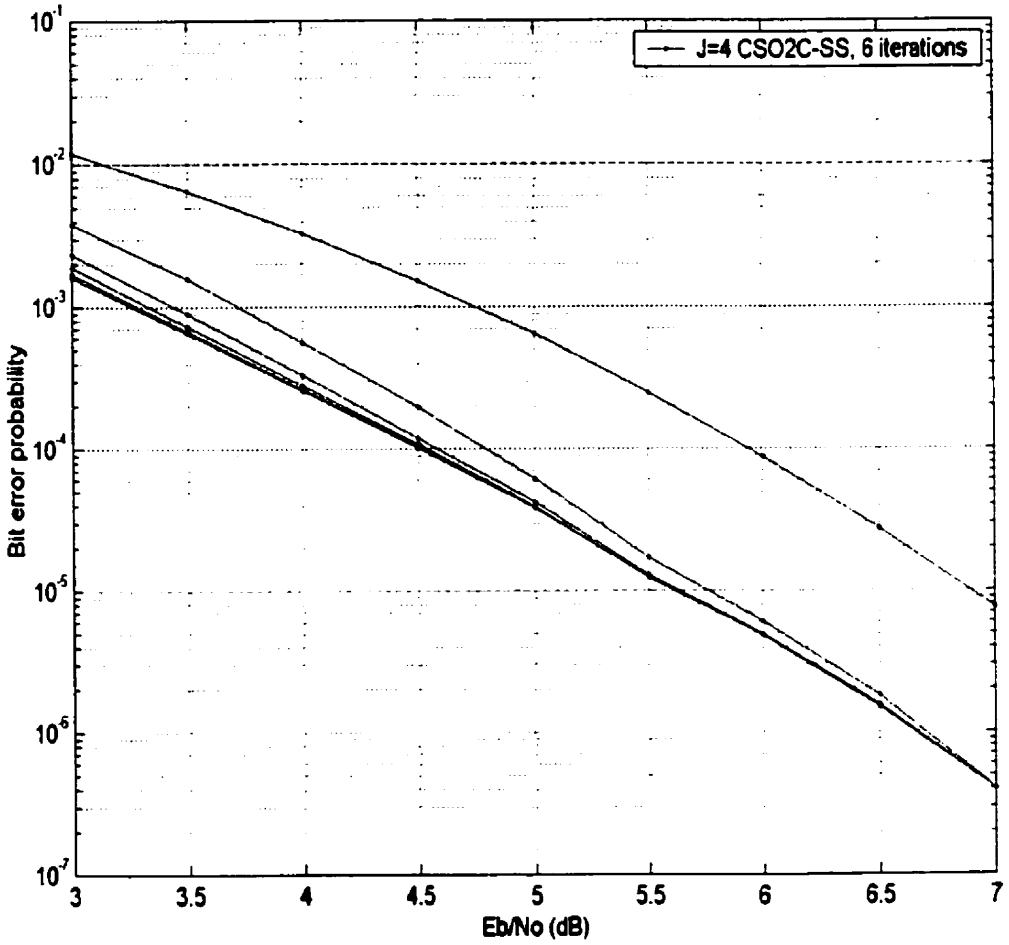
ANNEXE V : RÉSULTATS DE SIMULATION DES CODES CSO^{2}C-SS



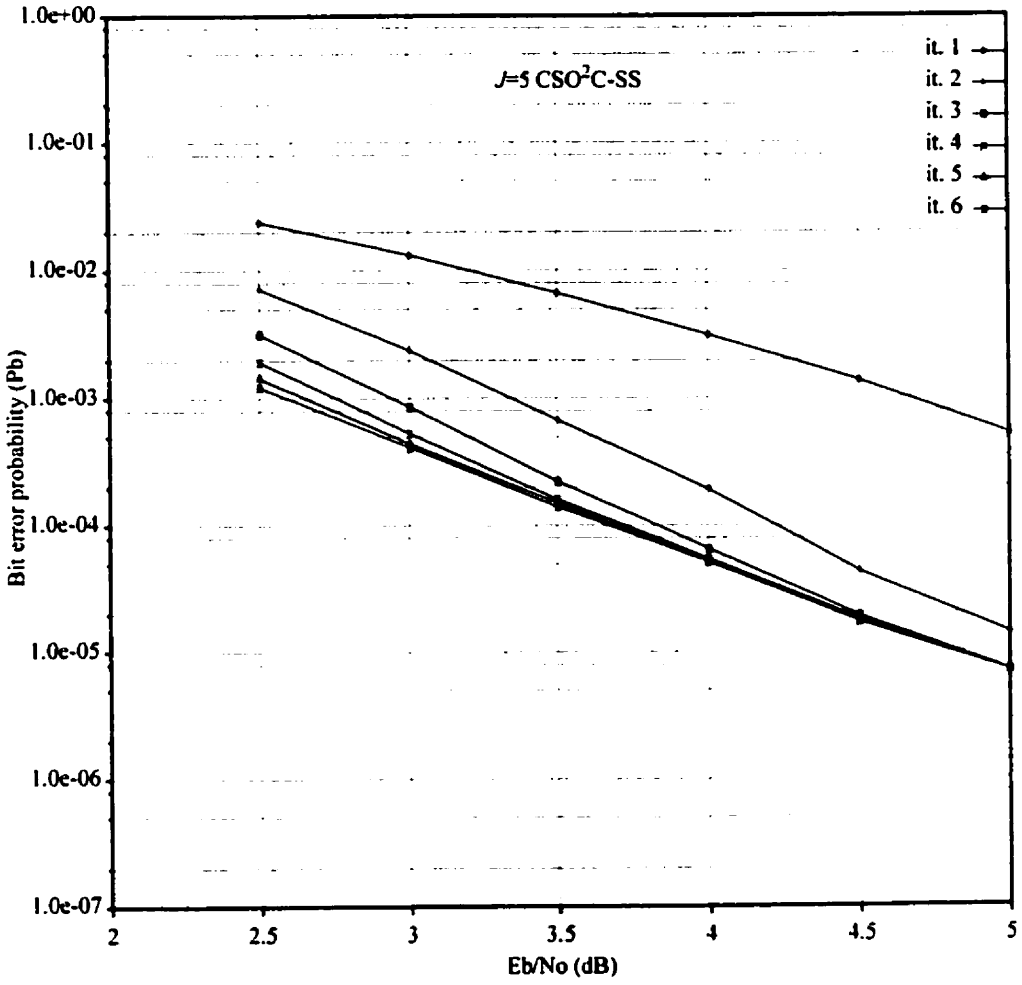
Matrice de codage : $[\alpha_{j,n}] = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$



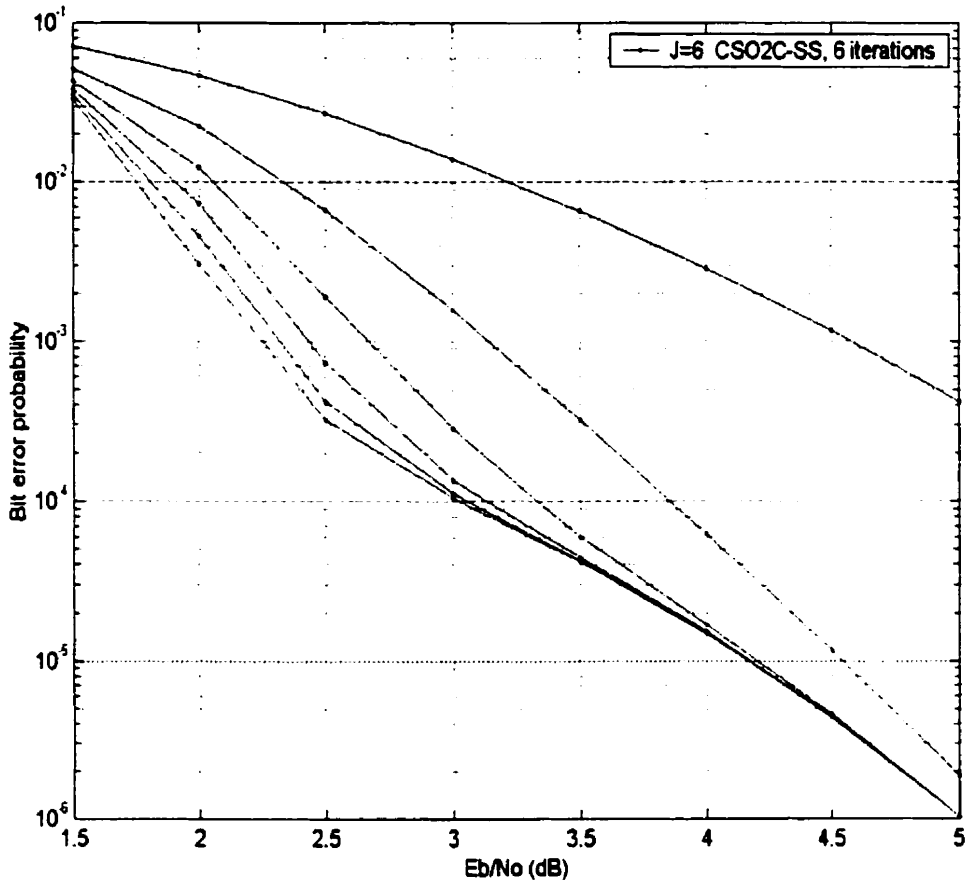
Matrice de codage : $[\alpha_{j,n}] = \begin{bmatrix} 0 & 0 & 5 \\ 1 & 3 & 0 \\ 5 & 2 & 0 \end{bmatrix}$



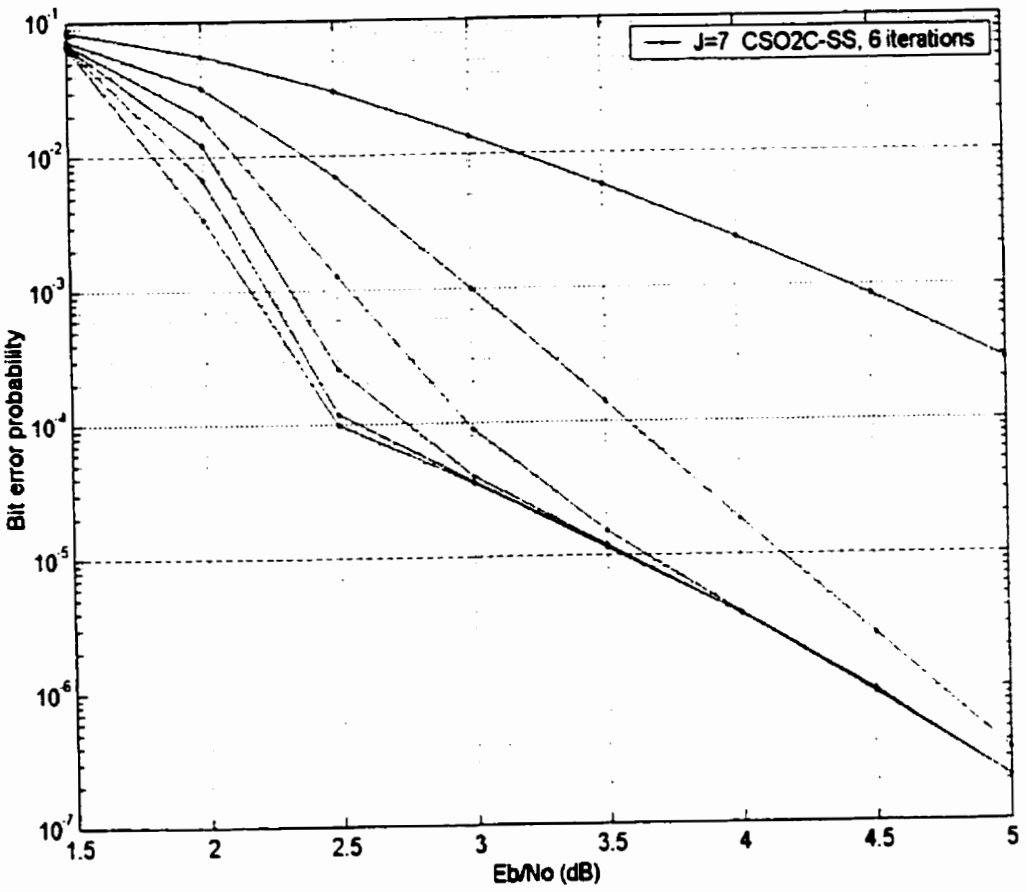
$$\text{Matrice de codage : } [\alpha_{j,n}] = \begin{bmatrix} 13 & 0 & 24 & 25 \\ 0 & 1 & 0 & 24 \\ 8 & 25 & 14 & 0 \\ 15 & 3 & 23 & 0 \end{bmatrix}$$



Matrice de codage : $[\alpha_{j,n}] = \begin{bmatrix} 28 & 26 & 0 & 14 & 0 \\ 41 & 21 & 103 & 58 & 0 \\ 92 & 33 & 0 & 44 & 104 \\ 0 & 0 & 103 & 0 & 61 \\ 29 & 103 & 0 & 24 & 4 \end{bmatrix}$

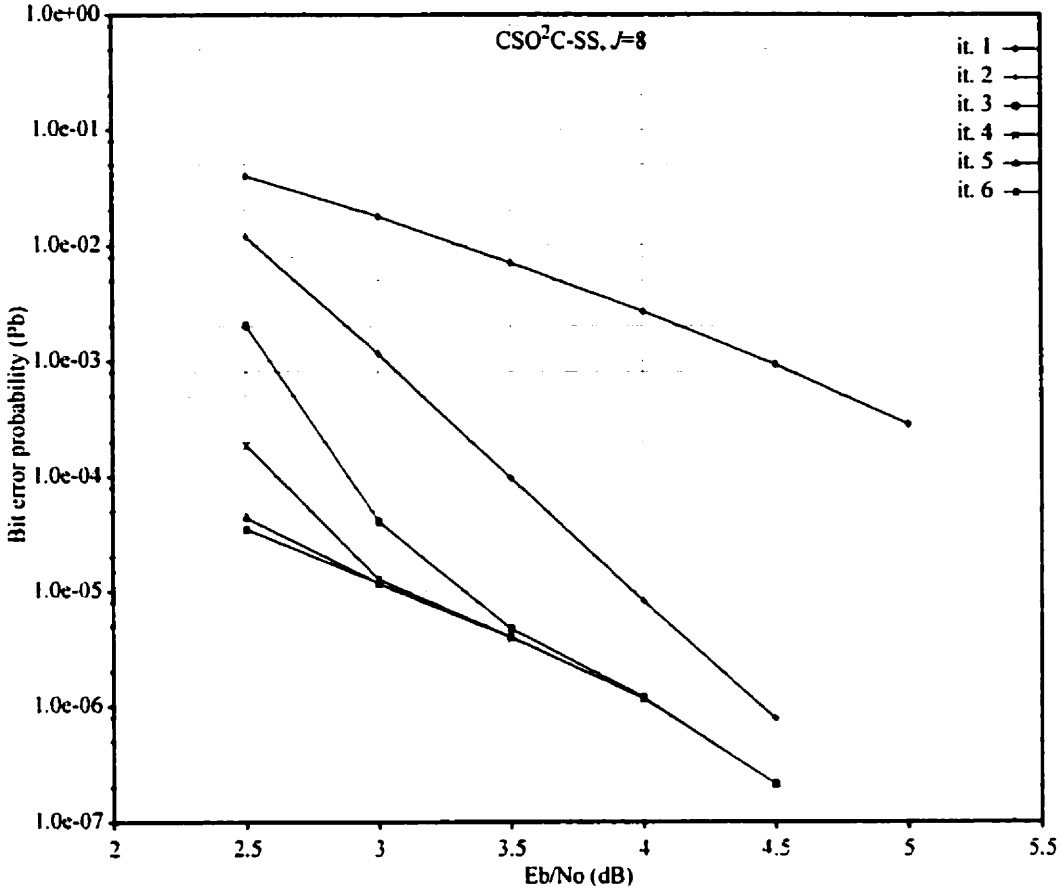


Matrice de codage : $[\alpha_{j,n}] = \begin{bmatrix} 238 & 308 & 289 & 0 & 240 & 302 \\ 120 & 0 & 313 & 232 & 0 & 111 \\ 339 & 154 & 243 & 0 & 121 & 240 \\ 228 & 317 & 156 & 0 & 27 & 158 \\ 0 & 209 & 0 & 363 & 314 & 0 \\ 271 & 165 & 363 & 0 & 187 & 272 \end{bmatrix}$



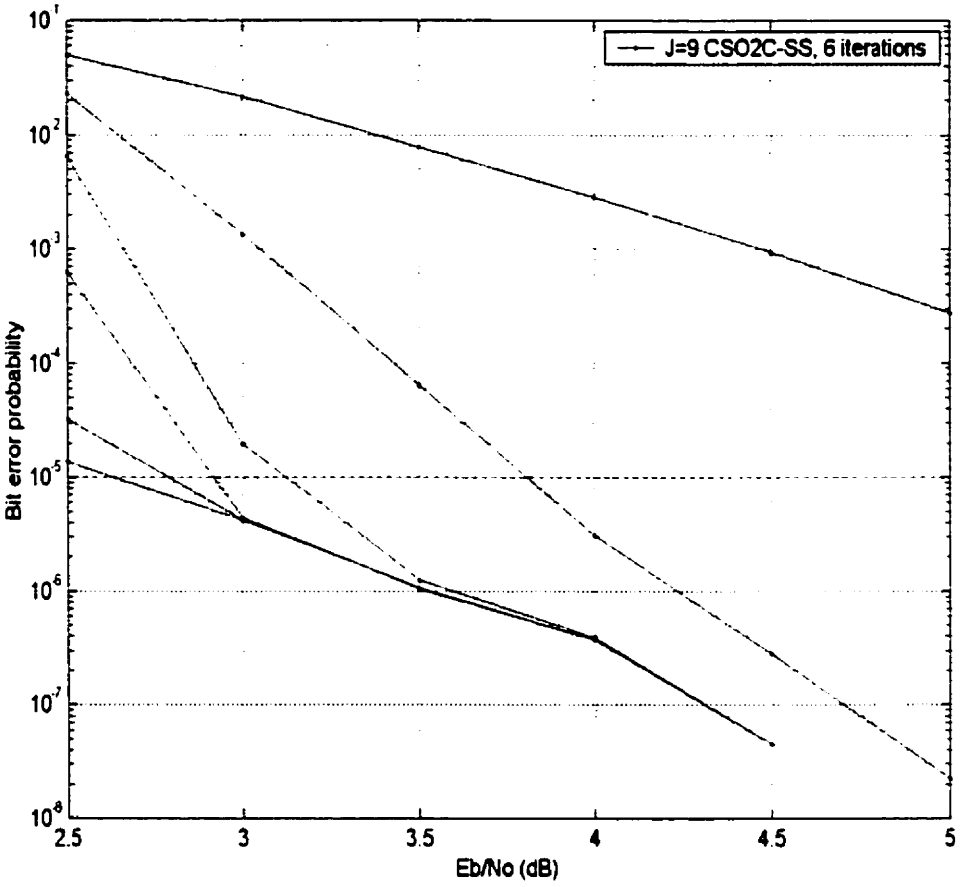
Matrice de codage : $[\alpha_{j,n}] =$

$$\begin{bmatrix} 0 & 98 & 273 & 970 & 406 & 202 & 224 \\ 0 & 37 & 198 & 199 & 12 & 131 & 189 \\ 1101 & 697 & 77 & 315 & 0 & 527 & 105 \\ 67 & 0 & 177 & 183 & 38 & 103 & 1043 \\ 0 & 167 & 0 & 0 & 109 & 1102 & 18 \\ 0 & 93 & 306 & 289 & 733 & 278 & 213 \\ 24 & 92 & 512 & 130 & 1101 & 0 & 0 \end{bmatrix}$$



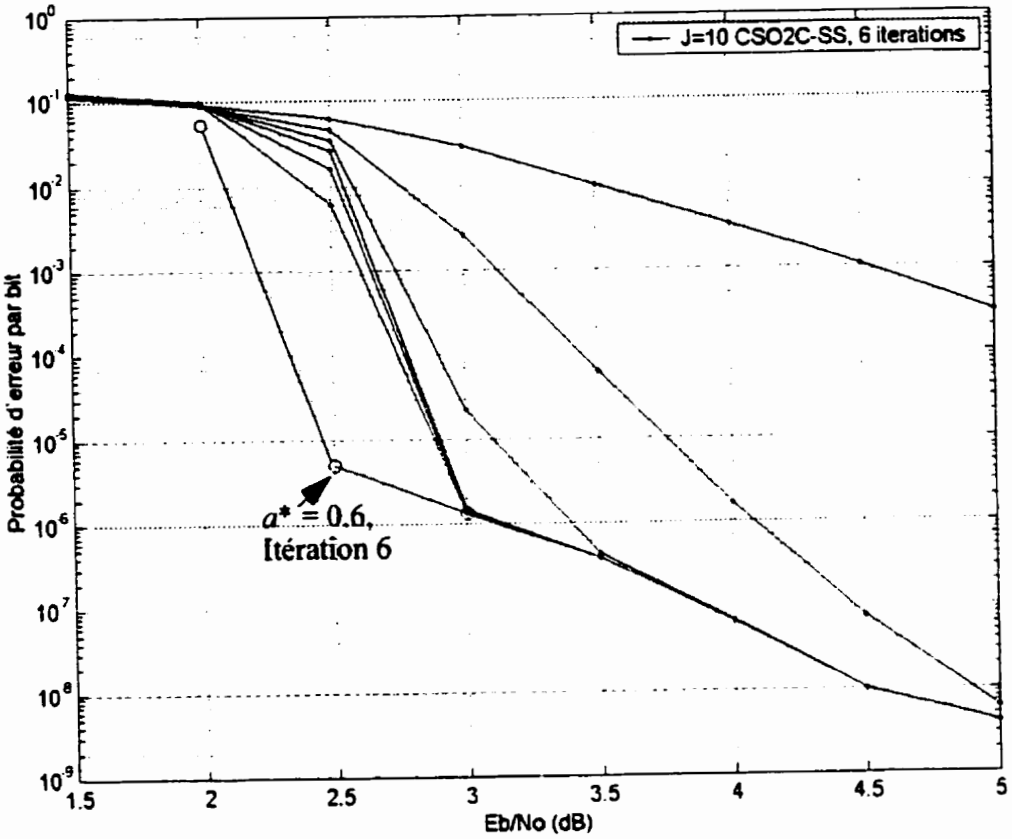
Matrice de codage : $[\alpha_{J,n}] =$

$$\begin{bmatrix} 151 & 58 & 65 & 2263 & 0 & 377 & 386 & 320 \\ 0 & 87 & 14 & 553 & 650 & 2179 & 207 & 505 \\ 2212 & 340 & 62 & 0 & 2660 & 0 & 0 & 0 \\ 0 & 2542 & 787 & 45 & 0 & 210 & 208 & 165 \\ 1104 & 82 & 104 & 185 & 0 & 537 & 497 & 2311 \\ 196 & 59 & 158 & 894 & 0 & 661 & 367 & 1413 \\ 88 & 0 & 0 & 124 & 0 & 1478 & 2659 & 288 \\ 1958 & 68 & 62 & 304 & 0 & 744 & 1900 & 925 \end{bmatrix}$$



Matrice de codage : $[\alpha_{j,n}] =$

$$\begin{bmatrix} 0 & 1082 & 380 & 5408 & 2908 & 226 & 589 & 966 & 671 \\ 0 & 350 & 187 & 3387 & 1434 & 559 & 581 & 607 & 5316 \\ 0 & 1606 & 50 & 482 & 203 & 830 & 447 & 471 & 23 \\ 3095 & 183 & 973 & 601 & 198 & 0 & 2896 & 479 & 23 \\ 1197 & 2840 & 92 & 438 & 163 & 0 & 419 & 1307 & 3027 \\ 0 & 5529 & 0 & 392 & 3887 & 230 & 379 & 950 & 0 \\ 0 & 1069 & 2256 & 609 & 361 & 145 & 1666 & 5707 & 304 \\ 0 & 454 & 128 & 580 & 5610 & 79 & 638 & 555 & 953 \\ 5707 & 0 & 65 & 0 & 0 & 1269 & 0 & 0 & 3338 \end{bmatrix}$$



Matrice de codage :

$$[\alpha_{j,n}] = \begin{bmatrix} 1567 & 5917 & 658 & 3551 & 9830 & 1714 & 0 & 3429 & 806 & 1787 \\ 10316 & 1317 & 656 & 1694 & 2128 & 1825 & 0 & 994 & 872 & 2492 \\ 3160 & 1305 & 1533 & 2878 & 1634 & 1689 & 0 & 962 & 1500 & 1685 \\ 1642 & 6999 & 820 & 2115 & 2927 & 1713 & 0 & 1276 & 854 & 4289 \\ 252 & 0 & 8826 & 373 & 660 & 419 & 1073 & 4121 & 0 & 648 \\ 8285 & 1320 & 5533 & 4892 & 3085 & 1908 & 0 & 2753 & 801 & 6252 \\ 1416 & 11352 & 0 & 9974 & 1009 & 1059 & 2550 & 373 & 170 & 2201 \\ 0 & 0 & 12426 & 0 & 0 & 0 & 0 & 5297 & 550 & 0 \\ 3424 & 798 & 0 & 4925 & 2839 & 1020 & 12426 & 291 & 119 & 11559 \\ 198 & 0 & 1195 & 2233 & 410 & 351 & 2725 & 0 & 6234 & 522 \end{bmatrix}$$

ANNEXE VI : PROBABILITÉ D'ERREUR DU PROCESSUS DE DÉCODAGE À SEUIL ITÉRATIF EN QUANTIFICATION FERME

Dans cette annexe, on évalue l'équation (3.11) permettant d'obtenir la probabilité d'erreur du processus de décodage itératif à seuil en quantification ferme. L'équation 3.11 à évaluer est donnée par :

$$P_b^{(\mu)} = P\left\{\sum_{n=1}^{J} s'^{(\mu)}_{i+\alpha_n} > T^{(\mu)} \middle| e_i^u = 0\right\}(1-\gamma_t) + P\left\{\sum_{n=1}^{J} s'^{(\mu)}_{i+\alpha_n} \le T^{(\mu)} \middle| e_i^u = 1\right\}\gamma_t \quad \text{(VI.1)}$$

où γ_t la probabilité de transition du canal $\gamma_t = P\left\{e_i^u = 1\right\} = Q(\sqrt{E_b/N_o})$. Cette probabilité de transition du canal γ_t représente aussi la probabilité qu'une erreur sur le symbole d'information u_i se produise, c'est-à-dire que :

$$P(e_i^u = 1) = 1 - P(e_i^u = 0) = \gamma_t \quad \text{(VI.2)}$$

La probabilité d'erreur à la première itération, (μ=1), peut s'écrire comme suit:

$$P_b^{(1)} = P\left\{\sum_{n=1}^{J} s'^{(1)}_{i+\alpha_n} > T^{(1)} \middle| e_i^u = 0\right\}(1-\gamma_t) + P\left\{\sum_{n=1}^{J} s'^{(1)}_{i+\alpha_n} \le T^{(1)} \middle| e_i^u = 1\right\}\gamma_t \quad \text{(VI.3)}$$

Afin d'évaluer VI.3, on doit tout d'abord calculer les probabilités suivantes [Massey, éq. 104]:

$$P\left\{s'^{(1)}_{i+\alpha_n} = 1 \middle| e_i^u = 0\right\} = \frac{1}{2}(1-(1-2\gamma_t)^n) = P^{(1)}_{0,n},\ n=1, \ldots, J \quad \text{(VI.4)}$$

et

$$P\left\{ s'^{(1)}_{i+\alpha_n} = 1 \middle| e_i^u = 1 \right\} = 1 - P^{(1)}_{0,n} = P^{(1)}_{1,n}, n=1, ..., J \qquad \text{(VI.5)}$$

La probabilité que la somme des équations de syndrome, $s'^{(1)}_{i+\alpha_n}$, soit égale à v sachant l'erreur sur le symbole d'information se calcule en utilisant les équations VI.4 et VI.5 et en construisant une fonction d'énumération. On obtient la fonction d'énumération $g_0^{(1)}(z)$ de la variable aléatoire $v = \sum_{n=1}^{J} s'^{(1)}_{i+\alpha_n}$ sachant que le bit d'erreur $e_i^u = 0$ comme suit :

$$g_0^{(1)}(z) = \prod_{n=1}^{J} (P^{(1)}_{0,n} z + P^{(1)}_{1,n}) = g^{(1)}_{0,0} z^0 + g^{(1)}_{0,1} z^1 + \ldots + g^{(1)}_{0,J} z^J \qquad \text{(VI.6)}$$

où z est une variable muette. De la même manière, on a que la fonction d'énumération $g_1^{(1)}(z)$ de la variable aléatoire $v = \sum_{n=1}^{J} s'^{(1)}_{i+\alpha_n}$ sachant que le bit d'erreur $e_i^u = 1$ est:

$$g_1^{(1)}(z) = \prod_{n=1}^{J} (P^{(1)}_{1,n} z + P^{(1)}_{0,n}) = g^{(1)}_{1,0} z^0 + g^{(1)}_{1,1} z^1 + \ldots + g^{(1)}_{1,J} z^J \qquad \text{(VI.7)}$$

On peut alors calculer les deux probabilités suivantes:

$$P\left\{ \sum_{n=1}^{J} s'^{(1)}_{i+\alpha_n} = h \middle| e_i^u = 0 \right\} = g^{(1)}_{o,h} \qquad \text{(VI.8)}$$

$$P\left\{ \sum_{n=1}^{J} s'^{(1)}_{i+\alpha_n} = h \middle| e_i^u = 1 \right\} = g^{(1)}_{1,h} \qquad \text{(VI.9)}$$

À partir de VI.8 et VI.9, on peut facilement déduire les probabilités:

$$P\left\{\sum_{n=1}^{J} s'^{(1)}_{i+\alpha_n} > T^{(1)} \middle| e_i^u = 0\right\} = \sum_{k=T^{(1)}+1}^{J} \frac{1}{k!}\frac{d^k}{dz^k} g_0^{(1)}(z)\Big|_{z=0} \qquad \text{(VI.10)}$$

$$P\left\{\sum_{n=1}^{J} s'^{(1)}_{i+\alpha_n} \le T^{(1)} \middle| e_i^u = 1\right\} = \sum_{k=T^{(1)}}^{J} \frac{1}{k!}\frac{d^k}{dz^k} g_1^{(1)}(z)\Big|_{z=0} \qquad \text{(VI.11)}$$

Le calcul de la probabilité des autres itérations se fait selon le même principe que pour la première itération mais, en tenant compte que la décision dépend des bits d'erreur $e^u_{i+\alpha_n-\alpha_j}$ et $\hat{e}^{(\mu)}_{i+\alpha_n-\alpha_j}$. Ainsi, en utilisant 3.10, on trouve, pour l'itération (μ>1) les relations suivantes :

$$P\left\{s'^{(\mu)}_{i+\alpha_n} = 1 \middle| e_i^u = 0\right\} = \frac{1}{2}\left(1-(1-2\gamma_t)(1-2P_b^{(\mu-1)})^{n-1}\right) = P^{(\mu)}_{0,n}. \qquad \text{(VI.12)}$$

$$P\left\{s'^{(\mu)}_{i+\alpha_n} = 1 \middle| e_i^u = 1\right\} = 1 - P^{(\mu)}_{0,n} = P^{(\mu)}_{1,n},\ n = 1,\ldots,J \qquad \text{(VI.13)}$$

En utilisant les équations VI.12 et VI.13, les deux fonctions d'énumération $g_0^{(\mu)}(z)$ et $g_1^{(\mu)}(z)$ donnent:

$$g_0^{(\mu)}(z) = \prod_{n=1}^{J}(P^{(\mu)}_{0,n}z + P^{(\mu)}_{1,n}) = g^{(\mu)}_{0,0}z^0 + g^{(\mu)}_{0,1}z^1 + \ldots + g^{(\mu)}_{0,J}z^J \qquad \text{(VI.14)}$$

$$\text{et } g_1^{(\mu)}(z) = \prod_{n=1}^{J}(P^{(\mu)}_{1,n}z + P^{(\mu)}_{0,n}) = g^{(\mu)}_{1,0}z^0 + g^{(\mu)}_{1,1}z^1 + \ldots + g^{(\mu)}_{1,J}z^J \qquad \text{(VI.15)}$$

De ces deux fonctions d'énumération, on calcule les deux probabilités suivantes:

$$P\left\{\sum_{n=1}^{J} s'^{(\mu)}_{i+\alpha_n} = h \middle| e_i^u = 0\right\} = g^{(\mu)}_{o,h} \tag{VI.16}$$

$$P\left\{\sum_{n=1}^{J} s'^{(\mu)}_{i+\alpha_n} = h \middle| e_i^u = 1\right\} = g^{(\mu)}_{1,h} \tag{VI.17}$$

À partir des équations VI.14 à VI.17, on trouve:

$$P\left\{\sum_{n=1}^{J} s'^{(\mu)}_{i+\alpha_n} > T^{(\mu)} \middle| e_i^u = 0\right\} = \sum_{k=T^{(\mu)}+1}^{J} \frac{1}{k!}\frac{d^k}{dz^k} g_0^{(\mu)}(z)\Big|_{z=0} \tag{VI.18}$$

$$P\left\{\sum_{n=1}^{J} s'^{(\mu)}_{i+\alpha_n} \leq T^{(\mu)} \middle| e_{I_o} = 1\right\} = \sum_{k=T^{(\mu)}}^{J} \frac{1}{k!}\frac{d^k}{dz^k} g_1^{(\mu)}(z)\Big|_{z=0} \tag{VI.19}$$

Ainsi, en suivant cette procédure de calcul, la probabilité $P_b^{(\mu)}$ de l'équation VI.1. se calcule comme suit:

$$P_b^{(\mu)} = \gamma_t \sum_{k=T^{(\mu)}+1}^{J} \frac{1}{k!}\frac{d^k}{dz^k} g_0^{(\mu)}(z)\Big|_{z=0} + (1-\gamma_t) \sum_{k=T^{(\mu)}}^{J} \frac{1}{k!}\frac{d^k}{dz^k} g_1^{(\mu)}(z)\Big|_{z=0} \tag{VI.20}$$

ANNEXE VII : ALGORITHME DE RÉTROPROPAGATION DE L'ERREUR ADAPTÉ AU DÉCODAGE À SEUIL ITÉRATIF

VI.A Dérivation de l'algorithme

Dans cette annexe, le développement mathématique menant à l'algorithme de rétropropagation de l'erreur adaptée au décodage itératif à seuil à sortie pondérée est donné. Il s'agit de minimiser l'erreur quadratique moyenne, $E^{(M)} = \overline{\left\{\lambda_i^{(M)} - x_i^u\right\}^2}$, calculée à la dernière itération M, par rapport aux coefficients de pondération $a_j^{(\mu)}$, $j = 0, \ldots, J$, $\mu = 1, \ldots, M$. Pour une longueur de séquence d'information L, $E^{(M)}$ s'écrit :

$$E^{(M)} = \frac{1}{L}\sum_{i=0}^{L-1}(\lambda_i^{(M)} - x_i^u)^2 \qquad \text{(VII.1)}$$

Posons l'erreur instantannée à l'itération M comme étant :

$$\delta_i^{(M)} = \lambda_i^{(M)} - x_i^u \qquad \text{(VII.2)}$$

Ainsi la dérivée de VII.1 par rapport à $a_0^{(M)}$ devient :

$$\frac{\partial E^{(M)}}{\partial a_0^{(M)}} = \frac{1}{L}\sum_{i=0}^{L-1}\frac{\partial\left|\delta_i^{(M)}\right|^2}{\partial a_0^{(M)}} = \frac{2}{L}\sum_{i=0}^{L-1}\delta_i^{(M)} y_i^u \qquad \text{(VII.3)}$$

En faisant appel à la règle delta, on détermine l'incrément associée à $a_0^{(M)}$:

$$\Delta a_0^{(M)} = -\frac{2}{L}\eta \sum_{i=0}^{L-1} \delta_i^{(M)} y_i^u \qquad \text{(VII.4)}$$

où η est le pas d'adaptation de l'algorithme. Les dérivées de VII.1 par rapport autres valeurs $a_j^{(M)}, j = 0, ..., J$ donnent :

$$\frac{\partial E^{(M)}}{\partial a_j^{(M)}} = \frac{1}{L}\sum_{i=0}^{L-1} \frac{\partial \left|\delta_i^{(M)}\right|^2}{\partial a_j^{(M)}} = \frac{2}{L}\sum_{i=0}^{L-1} \delta_i^{(M)} \psi_{j,i}^{(M)}, j = 0, ..., J \qquad \text{(VII.5)}$$

En faisant appel de nouveau à la règle delta, on détermine l'incrément associée à $a_j^{(M)}$:

$$\Delta a_j^{(M)} = -\frac{2}{L}\eta \sum_{i=0}^{L-1} \delta_i^{(M)} \psi_{j,i}^{(M)}, j = 0, ..., J \qquad \text{(VII.6)}$$

Pour l'itération précédente la dérivée de $E^{(M)}$ par rapport à $a_0^{(M-1)}$ donne :

$$\frac{\partial E^{(M)}}{\partial a_0^{(M-1)}} = \frac{1}{L}\sum_{i=0}^{L-1} \frac{\partial \left|\delta_i^{(M)}\right|^2}{\partial a_0^{(M-1)}} = \frac{2}{L}\sum_{i=0}^{L-1} \delta_i^{(M)} \frac{\partial \lambda_i^{(M)}}{\partial a_0^{(M-1)}} \qquad \text{(VII.7)}$$

Si l'on considère que $\lambda_i^{(M)}$ est une fonction des $\lambda_{i+k}^{(M-1)}$, k=1, ..., α_J provenant de l'itération précédente (M-1), en utilisant la règle en chaîne de la dérivée, VII.7 devient :

$$\frac{\partial E^{(M)}}{\partial a_0^{(M-1)}} = \frac{2}{L}\sum_{i=0}^{L-1} \delta_i^{(M)} \sum_{k=1}^{\alpha_J} \frac{\partial \lambda_i^{(M)}}{\partial \lambda_{i+k}^{(M-1)}} \frac{\partial \lambda_{i+k}^{(M-1)}}{\partial a_0^{(M-1)}} \qquad \text{(VII.8)}$$

On peut réduire VII.8 en posant ce qui suit :

$$\delta_{i+k}^{(M-1)} = \delta_i^{(M)} \frac{\partial \lambda_i^{(M)}}{\partial \lambda_{i+k}^{(M-1)}}, k=1, ..., \alpha_J \tag{VII.9}$$

où $\frac{\partial \lambda_i^{(M)}}{\partial \lambda_{i+k}^{(M-1)}} = 0$ lorsque $k \neq \alpha_j - \alpha_l, j$ et $l = 1, ..., J$. La quantité $\delta_{i+k}^{(M-1)}$ représente l'erreur propagée à l'itération (M-1). Selon VII.9, la relation VII.8 s'écrit :

$$\frac{\partial E^{(M)}}{\partial a_0^{(M-1)}} = \frac{2}{L} \sum_{i=0}^{L-1} \sum_{k=1}^{\alpha_J} \delta_{i+k}^{(M-1)} \frac{\partial \lambda_{i+k}^{(M-1)}}{\partial a_0^{(M-1)}} \tag{VII.10}$$

où la dérivée $\frac{\partial \lambda_{i+k}^{(M-1)}}{\partial a_0^{(M-1)}} = y_{i+k}^u$ de sorte que VII.10 devient :

$$\frac{\partial E^{(M)}}{\partial a_0^{(M-1)}} = \frac{2}{L} \sum_{i=0}^{L-1} \sum_{k=1}^{\alpha_J} \delta_{i+k}^{(M-1)} y_{i+k}^u \tag{VII.11}$$

De la même manière, les dérivées $\frac{\partial E^{(M)}}{\partial a_j^{(M-1)}}, j=1, ..., J$, donnent :

$$\frac{\partial E^{(M)}}{\partial a_j^{(M-1)}} = \frac{2}{L} \sum_{i=0}^{L-1} \sum_{k=1}^{\alpha_J} \delta_{i+k}^{(M-1)} \psi_{j,i+k}^{(M-1)}, j=1, ..., J \tag{VII.12}$$

L'application de la règle delta pour $a_j^{(M-1)}, j=0, ..., J$, donne les valeurs d'accroissement $\Delta a_j^{(M-1)}$ suivantes :

$$\Delta a_0^{(M-1)} = -\eta \frac{2}{L} \sum_{i=0}^{L-1} \sum_{k=1}^{\alpha_J} \delta_{i+k}^{(M-1)} y_{i+k}^u \tag{VII.13}$$

$$\Delta a_j^{(M-1)} = -\eta \frac{2}{L} \sum_{i=0}^{L-1} \sum_{k=1}^{\alpha_J} \delta_{i+k}^{(M-1)} \psi_{j,i+k}^{(M-1)}, j=1, ..., J \qquad \text{(VII.14)}$$

Cette même procédure peut s'étendre à l'itération (M-2). Par exemple, la dérivée $\frac{\partial E^{(M)}}{\partial a_0^{(M-2)}}$ donne :

$$\frac{\partial E^{(M)}}{\partial a_0^{(M-2)}} = \frac{2}{L} \sum_{i=0}^{L-1} \delta_i^{(M)} \frac{\partial \lambda_i^{(M)}}{\partial a_0^{(M-2)}} = \frac{2}{L} \sum_{i=0}^{L-1} \delta_i^{(M)} \sum_{k=1}^{\alpha_J} \frac{\partial \lambda_i^{(M)}}{\partial \lambda_{i+k}^{(M-1)}} \sum_{l=1}^{\alpha_J} \frac{\partial \lambda_i^{(M-1)} \partial \lambda_{i+k+l}^{(M-2)}}{\partial \lambda_{i+k+l}^{(M-2)} \partial a_0^{(M-2)}} \qquad \text{(VII.15)}$$

En insérant VII.9 dans , on obtient :

$$\frac{\partial E^{(M)}}{\partial a_0^{(M-2)}} = \frac{2}{L} \sum_{i=0}^{L-1} \sum_{k=1}^{\alpha_J} \delta_{i+k}^{(M-1)} \sum_{l=1}^{\alpha_J} \frac{\partial \lambda_i^{(M-1)} \partial \lambda_{i+k+l}^{(M-2)}}{\partial \lambda_{i+k+l}^{(M-2)} \partial a_0^{(M-2)}} \qquad \text{(VII.16)}$$

En définissant l'erreur propagée à l'itération (M-2) comme suit :

$$\delta_{i+l}^{(M-2)} = \sum_{k=1}^{\alpha_J} \delta_{i+k}^{(M-1)} \frac{\partial \lambda_i^{(M-1)}}{\partial \lambda_{i+k+l}^{(M-2)}}, l = 1, ..., \alpha_J \qquad \text{(VII.17)}$$

ou comme suit :

$$\delta_{i+l'}^{(M-2)} = \sum_{k=1}^{\alpha_J} \delta_{i+k}^{(M-1)} \frac{\partial \lambda_{i+k}^{(M-1)}}{\partial \lambda_{i+l'}^{(M-2)}}, l' = 1, ..., 2\alpha_J \qquad \text{(VII.18)}$$

avec $\frac{\partial \lambda_{i+k}^{(M-1)}}{\partial \lambda_{i+l'}^{(M-2)}} = 0$, $l' \le k$ et $(l'-k) \ne (\alpha_j - \alpha_n)$ pour j et n = 1, ..., J, (VII.16) peut s'écrit de la manière suivante :

$$\frac{\partial E^{(M)}}{\partial a_0^{(M-2)}} = \frac{2}{L}\sum_{i=0}^{L-1}\sum_{l=1}^{2\alpha_J}\delta_{i+l}^{(M-2)}\frac{\partial\lambda_{i+l}^{(M-2)}}{\partial a_0^{(M-2)}} = \frac{2}{L}\sum_{i=0}^{L-1}\sum_{l=1}^{2\alpha_J}\delta_{i+l}^{(M-2)}y_{i+l}^{u} \qquad \text{(VII.19)}$$

En suivant cette même procédure mais, en remplacant $a_0^{(M-2)}$ par $a_j^{(M-2)}$, j=1, ..., J, dans (VII.19), on obtient :

$$\frac{\partial E^{(M)}}{\partial a_j^{(M-2)}} = \frac{2}{L}\sum_{i=0}^{L-1}\sum_{l=1}^{2\alpha_J}\delta_{i+l}^{(M-2)}\psi_{j,i+l}^{(M-2)},\ j=1,\ldots,J \qquad \text{(VII.20)}$$

Encore une fois l'application de la règle delta donne :

$$\Delta a_0^{(M-2)} = -\eta\frac{2}{L}\sum_{i=0}^{L-1}\sum_{l=1}^{2\alpha_J}\delta_{i+l}^{(M-2)}y_{i+l}^{u} \qquad \text{(VII.21)}$$

$$\Delta a_j^{(M-2)} = -\eta\frac{2}{L}\sum_{i=0}^{L-1}\sum_{l=1}^{2\alpha_J}\delta_{i+l}^{(M-2)}\psi_{j,i+l}^{(M-2)},\ j=1,\ldots,J \qquad \text{(VII.22)}$$

De façon générale, on obtient à l'itération (μ) les relations suivantes :

$$\frac{\partial E^{(M)}}{\partial a_j^{(\mu)}} = \frac{2}{L}\sum_{i=0}^{L-1}\sum_{l=0}^{(M-\mu)\alpha_J}\delta_{i+l}^{(\mu)}\begin{cases} y_{i+l}^{u}, & j=0 \\ \psi_{j,i+l}^{(M-2)}, & j=1,\ldots,J \end{cases},\ \mu=1,\ldots,M \qquad \text{(VII.23)}$$

où $\delta_{i+l}^{(\mu)} = \sum_{k=0}^{(M-\mu-1)\alpha_J}\delta_{i+k}^{(\mu+1)}\frac{\partial\lambda_{i+k}^{(\mu+1)}}{\partial\lambda_{i+l}^{(\mu)}}$, l = 1, ..., (M-μ)α_J, $l > k$, μ=1, ..., (M-l) et où la règle delta donne :

$$\Delta a_j^{(\mu)} = -\eta \frac{2}{L} \sum_{i=0}^{L-1} \sum_{l=0}^{(M-\mu)\alpha_J} \delta_{i+l}^{(\mu)} \begin{cases} y_{i+l}^{u}, & j = 0 \\ \psi_{j,i+l}^{(M-2)}, & j = 1, \ldots, J \end{cases} \qquad \text{(VII.24)}$$

VI.B L'algorithme de rétropropagation de l'erreur

L'algorithme de rétropropagation de l'erreur s'écrit alors de la façon suivante :

1. $\delta_i^{(M)} = \lambda_i^{(M)} - x_i^u$, $i = 0, \ldots, (L\text{-}1)$

2. (a) $\Delta a_j^{(M)} = -\frac{2}{L}\eta \sum_{i=0}^{L-1} \delta_i^{(M)} \begin{cases} y_i^u, & j = 0 \\ \psi_{j,i}^{(M)}, & j = 1, \ldots, J \end{cases}$ $, j = 0, \ldots, J$

(b) $a_j^{(M)} \leftarrow a_j^{(M)} + \Delta a_j^{(M)}$

3. $l = 1, \ldots, (M\text{-}\mu)\alpha_J$, $l > k$, $\mu = 1, \ldots, (M\text{-}l)$:

$$\delta_{i+l}^{(\mu)} = \sum_{k=0}^{(M-\mu-1)\alpha_J} \delta_{i+k}^{(\mu+1)} \frac{\partial \lambda_{i+k}^{(\mu+1)}}{\partial \lambda_{i+l}^{(\mu)}}$$

4. (a) $\Delta a_j^{(\mu)} = -\eta \frac{2}{L} \sum_{i=0}^{L-1} \sum_{l=0}^{(M-\mu)\alpha_J} \delta_{i+l}^{(\mu)} \begin{cases} y_{i+l}^{u}, & j = 0 \\ \psi_{j,i+l}^{(m)}, & j = 1, \ldots, J \end{cases}$ $, j = 0, \ldots, J$

(b) $a_j^{(\mu)} \leftarrow a_j^{(\mu)} + \Delta a_j^{(\mu)}$

VI.C Évaluation de la dérivée $\partial\lambda_{i+k}^{(\mu+1)}/\partial\lambda_{i+l}^{(\mu)}$

Dans cette section de l'annexe, on évalue la dérivée $\dfrac{\partial\lambda_{i+k}^{(\mu+1)}}{\partial\lambda_{i+l}^{(\mu)}}$ nécessaire à l'équation de l'erreur propagée $\delta_{i+l}^{(\mu)}$ de l'étape 3 de l'algorithme présenté à la section VI.B.

La fonction $\lambda_{i+k}^{(\mu+1)}$ s'écrit de la façon suivante :

$$\lambda_{i+k}^{(\mu+1)} = a_0^{(\mu+1)} y_{i+k}^{u} + \sum_{j=1}^{J} a_j^{(\mu+1)} \psi_{j,i+k}^{(\mu+1)} \qquad \text{(VII.25)}$$

et selon 3.2, la valeur $\psi_{j,i+k}^{(\mu+1)}$ s'écrit comme suit :

$$\psi_{j,i+k}^{(\mu+1)} = y_{i+k+\alpha_j}^{p} \Diamond \sum_{h=1}^{j-1}\Diamond \lambda_{i+k+\alpha_j-\alpha_h}^{(\mu)} \Diamond \sum_{h=j+1}^{J}\Diamond \lambda_{i+k+\alpha_j-\alpha_h}^{(\mu+1)} \qquad \text{(VII.26)}$$

Il est clair que selon VII.25, on obtient que

$$\frac{\partial\lambda_{i+k}^{(\mu+1)}}{\partial\lambda_{i+l}^{(\mu)}} = \frac{\partial}{\partial\lambda_{i+l}^{(\mu)}}\left(a_0^{(\mu+1)} y_{i+k}^{u} + \sum_{j=1}^{J} a_j^{(\mu+1)} \psi_{j,i+k}^{(\mu+1)}\right) = \sum_{j=1}^{J} a_j^{(\mu+1)} \frac{\partial\psi_{j,i+k}^{(\mu+1)}}{\partial\lambda_{i+l}^{(\mu)}} \qquad \text{(VII.27)}$$

Les J dérivées $\dfrac{\partial\psi_{j,i+k}^{(\mu+1)}}{\partial\lambda_{i+l}^{(\mu)}}$ s'obtiennent à partir de VII.26 comme suit :

$$\frac{\partial\psi_{j,i+k}^{(\mu+1)}}{\partial\lambda_{i+l}^{(\mu)}} = \frac{\partial}{\partial\lambda_{i+l}^{(\mu)}}\left(y_{i+k+\alpha_j}^{p} \Diamond \sum_{h=1}^{j-1}\Diamond \lambda_{i+k+\alpha_j-\alpha_h}^{(\mu)} \Diamond \sum_{h=j+1}^{J}\Diamond \lambda_{i+k+\alpha_j-\alpha_h}^{(\mu+1)}\right) \qquad \text{(VII.28)}$$

En utilisant la définition de l'opérateur add-min donnée par 2.41, on obtient :

$$\frac{\partial \psi_{j,i+k}^{(\mu+1)}}{\partial \lambda_{i+l}^{(\mu)}} = \frac{\partial}{\partial \lambda_{i+l}^{(\mu)}} \Bigg(sign\left(y_{i+k+\alpha_j}^{p}\right) \prod_{h=1}^{j-1} sign\left(\lambda_{i+k+\alpha_j-\alpha_h}^{(\mu)}\right) \prod_{h=j+1}^{J} sign\left(\lambda_{i+k+\alpha_j-\alpha_h}^{(\mu+1)}\right) \cdot$$

$$min\Bigg(\left|y_{i+k+\alpha_j}^{p}\right|, \left|\lambda_{i+k+\alpha_j-\alpha_1}^{(\mu)}\right|, \ldots, \left|\lambda_{i+k+\alpha_j-\alpha_{j-1}}^{(\mu)}\right|, \cdots$$

$$\ldots, \left|\lambda_{i+k+\alpha_j-\alpha_{j+1}}^{(\mu+1)}\right|, \ldots, \left|\lambda_{i+k+\alpha_j-\alpha_J}^{(\mu+1)}\right| \Bigg) \Bigg)$$

$$= \begin{cases} -sign(\psi_{j,i+k}^{(\mu+1)}), & si\ (l-k) = (\alpha_j - \alpha_h)\ et\ \lambda_{i+l}^{(\mu)} < 0\ et\ \lambda_{i+l}^{(\mu)}\ minimum \\ sign(\psi_{j,i+k}^{(\mu+1)}), & si\ (l-k) = (\alpha_j - \alpha_h)\ et\ \lambda_{i+l}^{(\mu)} > 0\ et\ \lambda_{i+l}^{(\mu)}\ minimum \\ 0, & autrement \end{cases}$$

(VII.29)

ANNEXE VIII : ALGORITHME «BELIEF PROPAGATION» OU DE PEARL

Dans cette annexe, on décrit brièvement l'algorithme «Belief Propagation», BP aussi connu sous le nom de l'algorithme de Pearl. Cette description est basée sur la référence [48] de sorte que la notation de cette même référence est utilisée. Le lecteur pourra donc consulter les références [48] à [50] et [56] s'il désire obtenir plus détails sur le fonctionnement et les applications de cet algorithme.

L'algorithme BP permet de résoudre le problème général de l'inférence probabiliste que l'on retrouve notamment dans les problèmes de décodage. Il permet d'évaluer la probabilité conditionnelle qu'une variable aléatoire (ou dite cachée. dans la terminologie adoptée en intelligence artificielle) $X_i = a$ sachant un vecteur d'observables $\boldsymbol{Y}$. Plus précisément, le résultat final obtenu par l'utilisation de l'algorithme de Pearl correspond à la probabilité *a posteriori* $P(X_i = a|\boldsymbol{Y})$ que l'on dénote par $BEL_i(a)$:

$$BEL_i(a) = P(X_i = a|\boldsymbol{Y}) \qquad \text{(VIII.1)}$$

La règle de décision qui minimise la probabilité de produire un décision incorrecte est alors la suivante : *si* $BEL_i(a_o) > BEL_i(a)$, $a_o \neq a$ *alors* $X_i = a_o$

L'évaluation de $BEL_i(a)$ se fait à travers un réseau bayesien. La Figure VII.1 illustre deux exemples de réseaux bayesiens. Dans ces exemples, les réseaux bayesiens sont appropriés pour décrire les codes LDPC et les codes Turbo.

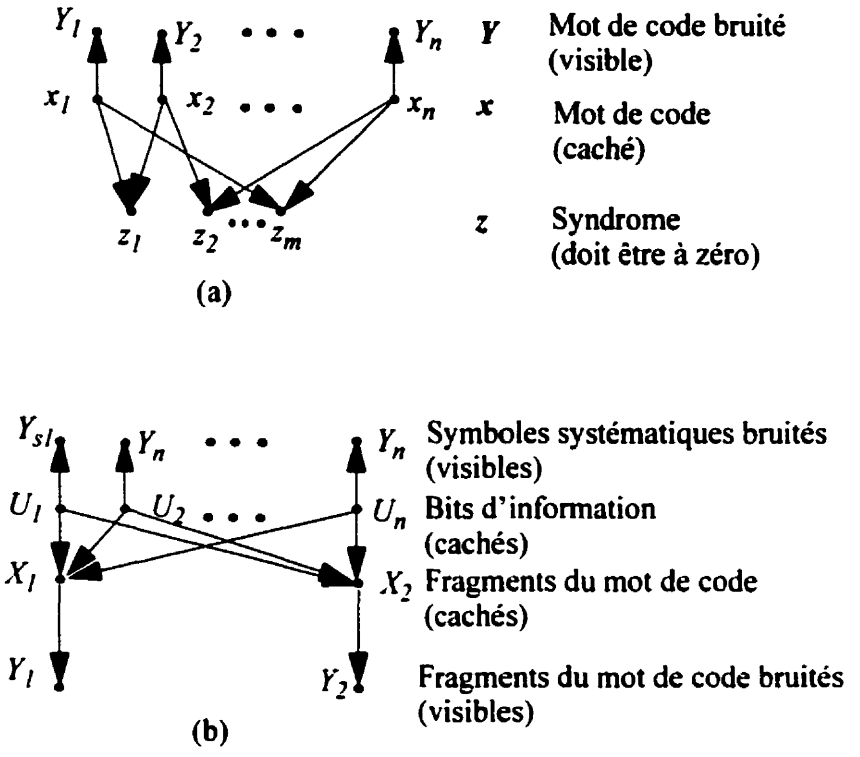


Figure VII.1 Réseaux bayesiens a) codes LDPC, b) codes Turbo

L'algorithme de Pearl est essentiellement basé sur un principe d'échange de «messages» entre les noeuds du réseau bayesien où chaque noeud représente une variable. À chaque noeud, un processeur est associé. Le réseau étant réprésenté par un graphe dirigé, chaque noeud peut être un «parent» d'un autre noeud et peut aussi être l'«enfant» d'un autre noeud.

Si $\boldsymbol{U} = (U_1, U_2, ..., U_n)$ sont les parents de X, alors le processeur du noeud X suppose connaître la valeur de la probabilité $p(x|\boldsymbol{u})=p(X = x|U_1 = u_1, U_2 = u_2, ..., U_n = u_n)$. La Figure VII.2 illustre ce principe.

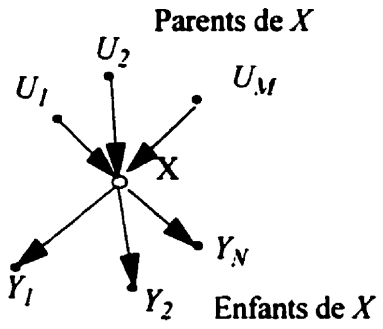


Figure VII.2 Section d'un réseau bayesien illustrant le principe de parent/enfant d'un noeud X

Lorsqu'un processeur est activé, il lit les messages provenant de chacun de ces noeuds parents et enfants, il met à jour sa valeur $BEL_X(x)$ en utilisant ces messages reçus et il envoie de nouveaux messages à ces noeuds parents et enfants. Lorsque le réseau bayesien contient des boucles (sans tenir compte de la direction des chemins dans le graphe), l'algorithme de Pearl nécessite une procédure itérative. C'est le cas des deux exemples montrés à la Figure VII.2. L'algorithme de Pearl peut se décrire en 5 étapes :

Phase de réception des messages :

Étape 1. Calculer la probabilité *a priori* du noeud X par une propagation de l'information *a priori* $\pi_{U_i,X}(u_i) = P_{U_i}(u_i)$ provenant du noeud U_i :

$$P_X(x) = \sum_{\boldsymbol{u}} P_{X|\boldsymbol{U}}(x|\boldsymbol{u}) \prod_{i=1}^{M} \pi_{U_i,X}(u_i) = \pi_X(x)$$

Étape 2. Calculer la probabilité *a posteriori* des noeuds observés $\boldsymbol{Y}$ par rétropropagation de la fonction de vraisemblance $\lambda_{Y_j,X}(x) = P_{Y_j|X}(y_j|x)$

allant des noeuds Y_j au noeud X :

$$P_{Y|U}(y|u) = \sum_{x} P_{X|U}(x|u) \prod_{j=1}^{N} \lambda_{Y_j X}(x) = \gamma_X(u)$$

La Figure VII.3a résume ces étapes de l'algorithme.

Étape 3. Calculer $\mathrm{BEL}_X(x)$: $\alpha \pi_X(x) \prod_{j=1}^{M} \lambda_{Y_j X}(x) = BEL_X(x)$, où le facteur α est une constante de normalisation.

Phase de transmission :

Étape 4. Calculer la valeur propagée :

$$P_{Y|U_i}(Y = y | U_i = u_i) = \lambda_{X, U_i}(u_i) = \sum_{u | U_i = u_i} \gamma_X(u) \prod_{l=1, l \neq i}^{M} \pi_{U_l, X}(u_l)$$

Étape 5. Calculer les valeurs propagées :

$$\pi_{X, Y_j} = P_{X|(Y_1, \ldots Y_{j-1}, Y_{j+1}, \ldots, Y_N)} = \alpha \pi_X(x) \prod_{i=1, i \neq j}^{N} \lambda_{Y_i, X}(x)$$

Ces échanges de messages sont résumés à la Figure VII.3b.

Figure VII.3 Résumé des échanges de message d'un noeud X vers les noeuds parents et les noeuds enfants

L'algorithme utilise aussi des règles d'initialisation qui sont données au tableau VIII.1.

Tableau VIII.1: Règles d'initialisation pour l'algorithme de Pearl

Quantité au noeud X	Initialisation
$\lambda_X(x)$	----------
$\pi_X(x)$	$p(x)$ * si X est un noeud source ------- sinon
$\gamma_X(u)$	1
$BEL_X(x)$	$p(x)$ si X est un noeud source ------- sinon
$\vec{\lambda}_{X,U}(u)$	1
$\pi_{X,Y_j}(x)$	$p(x)$ si X est un noeud source ------- sinon

(*) Une fois initialisée, cette quantité ne change pas au cours de l'algorithme.

ANNEXE IX : DÉCODAGE DES CODES À FAIBLE DENSITÉ DE PARITÉ

Dans cette annexe, on décrit l'algorithme de décodage des codes à faible densité de parité. Cet algorithme qui est aussi décrit dans [46], [50], peut être considéré comme une approximation de l'algorithme «*belief propagation*». Pour plus de facilité, la notation utilisée dans cette annexe est celle utilisée dans [50].

Le problème de ce décodage est de déterminer le vecteur des symboles codés $\boldsymbol{x}$ le plus probablement transmis tel que le produit $\boldsymbol{Hx} = \mathbf{z}$ (mod2) = 0, où $\boldsymbol{H}$ est est la matrice de contrôle de parité. Si la valeur binaire transmise pour chaque symbole x_n à l'instant n est $\pm a$ (signalisation antipodale) et que la valeur reçue est y_n (on suppose un canal à bruit blanc gaussien de variance σ^2), la vraisemblance de $\boldsymbol{x}$ est donnée par :

$$L(\boldsymbol{x}) = \prod_n f_n^{x_n} \tag{IX.1}$$

où f_n^1 , (x_n=1) s'obtient par :

$$f_n^1 = \frac{1}{1 + \exp\left(-2a\frac{y_n}{\sigma^2}\right)} \tag{IX.2}$$

et où $f_n^0 = 1 - f_n^1$. Les éléments x_n de $\boldsymbol{x}$ sont les bits et les rangés de $\boldsymbol{H}$ sont les symboles de contrôle de la parité z_m (voir Figure VII.1a). On dénote l'ensemble des bits n qui participent au symbole de contrôle de la parité m par $N(m) \equiv \{ n / H_{mn} = 1 \}$. De la même manière, on définit l'ensemble des symboles de contrôles de parité où le bit n participe

comme étant $M(n) \equiv \{m / H_{mn} = 1\}$. On dénote aussi par $N(m)\backslash n$, l'ensemble $N(m)$ excluant le bit n et par $M(n)\backslash m$ l'ensemble $M(n)$ excluant le symbole de contrôle de la parité m. L'algorithme est constitué de deux parties dans lesquelles les quantités q_{mn} et r_{mn} associées à chaque élément non-nul de $\boldsymbol{H}$ sont continuellement mises à jour. La quantité q_{mn}^{x} représente la probabilité que le bit n du vecteur x prenne la valeur x (0 ou 1) sachant l'information obtenue via l'ensemble des symboles de contrôle de la parité excluant le symbole m. La quantité r_{mn}^{x} représente la probabilité que le symbole de contrôle m soit satisfait (égal à zéro) si le bit n du vecteur x est considéré fixe à la valeur x et que les autres bits ont une distribution séparable donnée par $\{q_{mn'} / n' \in N(m)\backslash n\}$.

Initialisation : $q_{mn}^{0} = f_{n}^{0}$ et $q_{mn}^{1} = f_{n}^{1}$.

Étape «Horizontale» : On définit $\delta q_{mn} \equiv q_{mn}^{0} - q_{mn}^{1}$. Pour chaque m et n calculer:

$$\delta r_{mn} = \prod_{n' \in N(m)\backslash n} \delta q_{mn} \tag{IX.3}$$

$$r_{mn}^{0} = \frac{1}{2}(1 + \delta r_{mn}) \text{ et } r_{mn}^{1} = \frac{1}{2}(1 - \delta r_{mn}) \tag{IX.4}$$

Étape «Verticale» : Pour chaque m et n et pour $x = 0$ et 1, calculer :

$$q_{mn}^{x} = \alpha_{mn} f_{n}^{x} \prod_{m' \in M(n)\backslash m} r_{m'n}^{x} \tag{IX.5}$$

où α_{mn} est choisi de sorte que $q_{mn}^{0} + q_{mn}^{1} = 1$. On calcule aussi les valeurs des probabilités «pseudopostériori» q_n^0 et q_n^1 comme suit :

$$q_n^x = \alpha_n f_n^x \prod_{m \in M(n)} r_{mn}^x \qquad \text{(IX.6)}$$

où α_n est choisi pour que $q_n^0 + q_n^1 = 1$.

Décision : Si $q_n^1 > 0.5$ alors $\hat{x}_n = 1$ sinon $\hat{x}_n = 0$. Si $\boldsymbol{H}\hat{\boldsymbol{x}}(mod2) = 0$ alors la procédure de décodage est terminée sinon, retour à l'étape «Horizontale». Le décodage est déclaré «sans succès» si un maximum d'itérations est atteint (normalement ce nombre est fixé à 100).