Ouvrage écrit sous la direction de

Claude Berrou

Collection IRIS
dirigée par Nicolas Puech

Codes et turbocodes

Springer

Codes et turbocodes

Springer
Paris
Berlin
Heidelberg
New York
Hong Kong
Londres
Milan
Tokyo

Ouvrage écrit sous la direction de
Claude Berrou

Codes et turbocodes

Springer

Claude Berrou
École Nationale Supérieure des Télécommunications de Bretagne
CS 83818
29238 Brest Cedex 3

ISBN 13 : 978-2-287-32739-1 Springer Paris Berlin Heidelberg New York

Printed in France
Springer-Verlag France est membre du groupe Springer Science + Business Media

SPIN: 11682165

Maquette de couverture : Jean-François MONTMARCHÉ
Illustration de couverture : Jean-Noël JAFFRY

Liste des contributeurs

Cet ouvrage a été rédigé sous la direction de Claude Berrou avec la contribution de :

Karine Amis,
Matthieu Arzel,
Catherine Douillard,
Alain Glavieux †,
Frédéric Guilloud,
Michel Jézéquel,
Sylvie Kerouédan,
Charlotte Langlais,
Christophe Laot,
Raphaël Le Bidan,
Émeric Maury,
Samir Saoudi,
Yannick Saouter,
tous de l'École nationale supérieure des télécommunications de Bretagne,

Gérard Battail,
de l'École nationale supérieure des télécommunications,

Emmanuel Boutillon,
de l'Université de Bretagne Sud,

et avec le concours précieux de Mohamed Koubàa et de Nicolas Puech.

« Les deux mots les plus brefs et les plus anciens, oui et non, sont ceux qui exigent le plus de réflexion »

Pythagore, V^{e} siècle av. J.C.

À nos regrettés collègues et amis Alain Glavieux et Gérard Graton.

Avant-propos

C'est un double *big bang* qui a ouvert ce que l'on appelle communément aujourd'hui l'ère de l'information. Nous sommes en 1948 et les États-Unis d'Amérique continuent d'investir massivement dans la recherche de haute technologie, dont ils ont tiré les premiers bénéfices durant le conflit mondial. Dans les *Bell Telephone Laboratories*, installés dans le New Jersey, au sud de New York, plusieurs équipes se sont constituées autour de brillants chercheurs, pour beaucoup formés au MIT (*Massachusetts Institute of Technology*). Cette année-là se produisent deux découvertes exceptionnelles, l'une technologique, l'autre théorique, qui vont profondément marquer le $20^{ème}$ siècle. C'est en effet à quelques mois d'intervalle et dans le même établissement que John Bardeen, Walter Brattain et William Shockley inventent le transistor et que Claude Elwood Shannon établit la théorie de l'information et des communications numériques. Prodigieuse coïncidence qui fait naître comme presque jumeaux le composant semi-conducteur qui, suivant son état de conduction (ouvert ou fermé), est capable de représenter matériellement une information binaire (0 ou 1) et le *shannon* ou *bit* (contraction de *b*inary un*it*), unité de mesure de l'information.

On mesure bien aujourd'hui toute l'importance de ces deux inventions qui ont permis le formidable essor de l'informatique et des télécommunications, entre autres. Depuis 1948, les fulgurants progrès de l'électronique, puis de la micro-électronique, ont offert aux ingénieurs et chercheurs du monde des télécommunications le support de leurs innovations, pour accroître, toujours et toujours, les performances de leurs systèmes. Qui aurait pu imaginer, il y a peu, qu'un programme de télévision pourrait être transmis par une paire de fils téléphoniques ? En somme, Shockley et ses collègues, à travers la loi de Gordon Moore (doublement, tous les 18 mois, du nombre de transistors dans une même surface de silicium), ont apporté petit à petit les moyens de répondre aux défis lancés par Shannon, grâce à des algorithmes qui ne pouvaient être que de plus en plus complexes. Un exemple typique en est l'invention, plutôt tardive, des turbocodes et des traitement itératifs dans les récepteurs, qui ne purent être imaginés que parce que les dizaines ou centaines de milliers de transistors requis étaient disponibles.

Les experts de la micro-électronique prévoient comme butée finale de la technologie CMOS, vers 2015, autour de trois milliards de transistors par centimètre carré. C'est l'ordre de grandeur du nombre de neurones dans le cerveau humain (qui restera cependant incomparablement plus puissant, du fait de son extraordinaire réseau de connexions - plusieurs milliers de synapses par neurone). Des milliards de transistors dans une même puce, cela veut dire que les algorithmes les plus exigeants en ressources calculatoires, au moins parmi ceux qui sont connus aujourd'hui, y trouveront leur place sans jouer des coudes. Pour reprendre le slogan d'un fabricant de circuits intégrés : « la limitation n'est pas dans le silicium, elle est dans votre imagination ». Pour être honnête, précisons quand même que la conception et le test de ces fonctions complexes seront loin d'être aisés.

Cependant nous sommes déjà bien loin de l'époque où Andrew Viterbi, pour conclure la présentation de son fameux algorithme, en 1967, affichait un scepticisme à la mesure de sa modestie : « Bien que cet algorithme soit irréaliste du fait des contraintes excessives de mémorisation, il contribue à une compréhension générale des codes convolutifs, à travers sa simplicité opératoire et son analyse » [1]. Un décodeur de Viterbi, c'est aujourd'hui seulement un dixième de millimètre carré de silicium dans un téléphone portable.

Parmi les résultats présentés par Shannon dans la publication fondatrice [2], celui-ci est particulièrement étonnant : *dans une transmission numérique en présence de perturbation, si le niveau moyen de celle-ci ne dépasse pas un certain seuil de puissance et en utilisant un codage approprié, le récepteur peut identifier le message d'origine sans aucune erreur.* Par codage, on entend ici, comme dans la totalité du livre, codage correcteur d'erreurs, c'est-à-dire écriture redondante de l'information binaire. Le codage de source (compression numérique), le codage cryptographique, ou tout ce que le mot codage peut avoir comme autre signification, ne sont pas traités dans *Codes et turbocodes*.

Le résultat théorique établi par Shannon a constitué pour des milliers de chercheurs et d'ingénieurs un défi scientifique majeur car l'enjeu économique est considérable. Améliorer le pouvoir de correction d'un code, c'est à même qualité d'information reçue (par exemple, pas plus d'une information binaire fausse sur 10.000 reçues en téléphonie numérique), permettre au système de transmission de fonctionner dans des conditions plus sévères. Il est alors possible de réduire la taille des antennes, le poids des batteries d'alimentation ou l'encombrement des panneaux solaires. Dans les systèmes spatiaux (satellites, sondes, ...), l'économie peut se chiffrer en dizaines de millions de dollars, car le poids des équipements et la puissance du lanceur s'en trouvent notablement réduits. Dans les systèmes cellulaires de téléphonie mobile, améliorer le code, c'est aussi permettre à l'opérateur d'augmenter le nombre d'utilisateurs potentiels dans la cellule. Aujourd'hui, rares sont les systèmes de télécommunications qui n'intègrent pas dans leur spécification un code correcteur d'erreurs.

Un autre domaine d'applications des codes correcteurs est celui des mémoires de masse : disques durs d'ordinateurs, CD-ROM, DVD, ... Les progrès réalisés ces dernières années sur la miniaturisation des motifs élémentaires de mémorisation, magnétiques ou optiques, se sont accompagnés d'une dégradation inévitable des énergies disponibles lors de la lecture des données et donc d'une plus grande vulnérabilité aux perturbations. A cela s'ajoutent des effets accrus d'interférences entre voisins. L'utilisation de techniques déjà éprouvées dans les systèmes de télécommunications, codage et égalisation notamment, s'avère aujourd'hui indispensable pour contrer les effets induits par la miniaturisation de ces dispositifs de stockage. Bien que *Codes et turbocodes* n'aborde pas explicitement ces applications, les concepts qu'on y trouve développés, les algorithmes qui y sont présentés sont aussi d'une grande actualité dans l'industrie des mémoires de masse.

Cet ouvrage est donc principalement consacré au codage correcteur d'erreurs, encore appelé codage de canal, et à ses applications aux communications numériques, en association avec les modulations. Les principes généraux de l'écriture redondante de l'information et la plupart des techniques imaginées jusqu'en 1990 pour protéger les transmissions numériques, sont présentés dans la première moitié du livre (chapitres 1 à 6). Dans cette première partie, un chapitre est également dédié aux différentes techniques de modulation, sans lesquelles les signaux codés ne pourraient être véhiculés dans les milieux réels de transmission. La deuxième partie (chapitres 7 à 11) est consacrée aux turbocodes, inventés plus récemment (1990-93), et dont le pouvoir de correction, qui avoisine les limites théoriques prédites par Shannon, en fait un standard de codage dans des applications de plus en plus nombreuses. Différentes versions de turbocodes, ainsi que la famille des codes LDPC, sont présentées. Enfin, certaines techniques utilisant les principes du turbo-décodage, telles que la turbo-égalisation et la turbo-détection multi-utilisateurs, sont introduites en fin d'ouvrage.

Une caractéristique particulière de ce livre, par comparaison avec la manière dont peut être ailleurs abordé le problème du codage, est le souci de l'application. Les aspects mathématiques n'y sont traités que par nécessité et certains résultats, qui reposent sur des développements complexes, devront être admis. En revanche, les considérations pratiques, en particulier sur les algorithmes et les circuits de traitement, y sont largement détaillées et commentées. De nombreux exemples de performance sont fournis, pour différents schémas de codage et de modulations codées.

Les auteurs du livre sont des enseignants-chercheurs reconnus pour leur expertise dans le domaine des algorithmes et des circuits associés pour les communications. Ils sont notamment à l'origine des turbocodes et de la généralisation de « l'effet turbo » aux différentes fonctions de traitement de l'information dans les récepteurs. Un soin particulier a été apporté dans la rédaction de cet ouvrage collectif, vis-à-vis de l'unité de vue et de la cohérence des notations. Certains concepts, identiques ou similaires, peuvent toutefois y être introduits

à plusieurs reprises et de différentes manières, ce qui – espérons-le – n'enlève rien à la pédagogie de l'ouvrage car celle-ci est l'art de la répétition. *Codes et turbocodes* a été pensé pour être à la fois un livre de découverte du codage et du décodage correcteur d'erreurs, une source précieuse d'informations sur les nombreuses techniques imaginées depuis le milieu du vingtième siècle, et une ouverture vers des problèmes non encore complètement résolus.

Brest, Octobre 2006
Claude Berrou

[1] A. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding algorithm", *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 260-269, Apr. 1967.

[2] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, Vol. 27, July and October 1948.

Nota Bene : tout commentaire sur le contenu de cet ouvrage peut être envoyé à l'adresse électronique suivante : `turbocode@mlistes.enst-bretagne.fr`

Sommaire

Liste des contributeurs **v**

Avant-propos **ix**

1 Introduction **1**
1.1 Messages numériques . . . 3
1.2 Un premier code . . . 4
1.3 Décodage à entrée ferme et décodage à entrée souple . . . 8
1.4 Décodage à sortie ferme et décodage à sortie souple . . . 11
1.5 La mesure de performance . . . 12
1.6 Qu'est-ce qu'un bon code ? . . . 15
1.7 Les familles de codes . . . 17

2 Communications numériques **19**
2.1 Modulations Numériques . . . 19
2.1.1 Introduction . . . 19
2.1.2 Modulations Linéaires Sans Mémoire . . . 22
2.1.3 Modulations de fréquence sans mémoire à M états : MDF-M . . . 29
2.1.4 Modulations avec mémoire par déplacement de fréquence à phase continue : MDF-M PC . . . 31
2.2 Structure et performances du récepteur optimal sur canal gaussien 37
2.2.1 Structure du récepteur cohérent . . . 38
2.2.2 Performances du récepteur cohérent . . . 42
2.3 Transmission sur canal à bande limitée . . . 59
2.3.1 Introduction . . . 59
2.3.2 L'interférence entre symboles . . . 60
2.3.3 Condition d'absence d'IES : critère de Nyquist . . . 63
2.3.4 Expression de la probabilité d'erreur en présence de filtrage de Nyquist . . . 68
2.4 Transmission sur canal à évanouissements . . . 70
2.4.1 Caractérisation d'un canal à évanouissements . . . 70

2.4.2 Transmission sur canal non sélectif en fréquences et à évanouissements lents 73

3 Limites théoriques 83
3.1 La théorie de l'information 83
3.1.1 Canal de transmission 83
3.1.2 Un exemple : le canal binaire symétrique 84
3.1.3 Aperçu sur le théorème fondamental du codage 86
3.1.4 Interprétation géométrique 88
3.1.5 Codage aléatoire 88
3.2 Limites théoriques de performance 91
3.2.1 Canal à entrée binaire et sortie réelle 91
3.2.2 Capacité d'un canal de transmission 92
3.3 Limites pratiques de performance 96
3.3.1 Canal gaussien à entrée binaire 97
3.3.2 Canal gaussien à entrée continue 98
3.3.3 Quelques exemples de limites 100
3.4 Distances minimales requises 101
3.4.1 DMH requise avec la modulation MDP-4 101
3.4.2 DMH requise avec la modulation MDP-8 103
3.4.3 DMH requise avec la modulation MAQ-16 105
3.5 Bibliographie 107

4 Codes en bloc 109
4.1 Les codes en blocs à symboles binaires 110
4.1.1 Matrice génératrice d'un code en blocs binaire 110
4.1.2 Code dual et matrice de contrôle de parité 113
4.1.3 Distance minimale 113
4.1.4 Codes étendus et codes raccourcis 115
4.1.5 Codes produits 115
4.1.6 Exemples de codes en blocs binaires 116
4.1.7 Les codes cycliques 120
4.2 Les codes en blocs à symboles non binaires 130
4.2.1 Les codes de Reed-Solomon 130
4.2.2 Mise en oeuvre du codeur 131
4.3 Décodage et performances des codes à symboles binaires . . . 132
4.3.1 Détection d'erreur 132
4.3.2 Correction des erreurs 134
4.4 Décodage et performances des codes à symboles non binaires . 143
4.4.1 Décodage à entrée ferme des codes de Reed-Solomon . . 143
4.4.2 Méthode directe de Peterson 144
4.4.3 Méthode itérative 150
4.4.4 Performances du décodage à entrée ferme des codes de Reed-Solomon 158
4.5 Bibliographie 158

Annexe : Notions sur les corps de Galois et sur les polynômes minimaux 159

5 Les codes convolutifs et leur décodage 165
5.1 Historique . . . 165
5.2 Représentations des codes convolutifs . . . 167
5.2.1 Représentation générique d'un codeur convolutif . . . 167
5.2.2 Représentation polynomiale . . . 170
5.2.3 Arbre d'un code . . . 171
5.2.4 Treillis d'un code . . . 172
5.2.5 Machine à états d'un code . . . 174
5.3 Distances et performances des codes . . . 175
5.3.1 Du choix d'un bon code . . . 175
5.3.2 Séquences *RTZ* . . . 176
5.3.3 Fonction de transfert et spectre de distances . . . 177
5.3.4 Performances . . . 181
5.4 Le décodage des codes convolutifs . . . 184
5.4.1 Modèle de la chaîne de transmission et notations . . . 184
5.4.2 L'algorithme de Viterbi . . . 185
5.4.3 L'algorithme *Maximum A Posteriori* ou *MAP* . . . 189
5.5 Codes convolutifs en bloc . . . 190
5.5.1 Fermeture de treillis . . . 190
5.5.2 Poinçonnage . . . 193
5.6 Bibliographie . . . 196

6 Concaténation de codes 197
6.1 Concaténation parallèle et concaténation série . . . 199
6.2 Concaténation parallèle et codage *LDPC* . . . 202
6.3 Les permutations . . . 203
6.4 Turbo mots croisés . . . 204
6.5 Bibliographie . . . 206

7 Turbocodes convolutifs 207
7.1 L'histoire des turbocodes . . . 207
7.2 Concaténation multiple de codes CSR . . . 209
7.3 Les turbocodes . . . 211
7.3.1 La terminaison des codes constituants . . . 215
7.3.2 La fonction de permutation . . . 216
7.4 Le décodage des turbocodes . . . 225
7.4.1 Le turbodécodage . . . 225
7.4.2 Le décodage *SISO* et l'information extrinsèque . . . 229
7.4.3 Considérations pratiques . . . 234
7.5 Les turbocodes m-binaires . . . 239
7.5.1 Codeurs CSR $m-$binaires . . . 239
7.5.2 Turbocodes m-binaires . . . 240
7.6 Outils d'analyse . . . 245

7.6.1 Performances théoriques 245
7.6.2 Comportement asymptotique 245
7.6.3 Convergence 249
7.7 Bibliographie 255

8 Turbocodes produits **259**
8.1 Historique 259
8.2 Les codes produits 259
8.3 Le décodage à entrée ferme des codes produits 261
8.3.1 Le décodage ligne-colonne 261
8.3.2 L'algorithme de Reddy-Robinson 262
8.4 Le décodage à entrée souple des codes produits 265
8.4.1 L'algorithme de Chase à sortie pondérée 265
8.4.2 Performances de l'algorithme de Chase-Pyndiah 268
8.4.3 L'algorithme de Fang et Battail 269
8.4.4 L'algorithme de Hartmann-Nazarov 272
8.4.5 Autres algorithmes à entrée souple 276
8.5 Implantation de l'algorithme de Chase-Pyndiah 278
8.6 Bibliographie 280

9 Codes *LDPC* **283**
9.1 Principe des codes *LDPC* 283
9.1.1 Code de parité 284
9.1.2 Définition d'un code *LDPC* 287
9.1.3 Encodage 290
9.1.4 Décodage des codes *LDPC* 294
9.1.5 Construction d'un code *LDPC* 297
9.2 Architecture de décodage de codes *LDPC* pour le canal Gaussien 300
9.2.1 Analyse de la complexité 301
9.2.2 Architecture d'un Processeur de Nœud Générique (PNG) 302
9.2.3 Architecture générique de propagation des messages . . 306
9.2.4 Combinaisons des paramètres de l'architecture 307
9.2.5 Exemple de synthèse d'architecture de décodeurs *LDPC* 310
9.2.6 Algorithme de décodage sous optimaux 312
9.2.7 Influence de la quantification 316
9.2.8 État de l'art des architectures de décodeurs *LDPC* publiées 317
9.3 Bibliographie 319

10 Turbocodes et transmissions à grande efficacité spectrale **325**
10.1 Les turbo-modulations codées en treillis (TMCT) 325
10.2 Les modulations turbocodées pragmatiques 331
10.3 Bibliographie 338

11 Le principe turbo appliqué à l'égalisation et à la détection 341
11.1 La turbo-égalisation 342
11.1.1 Canaux multi-trajets et interférence entre symboles . . . 342
11.1.2 La fonction d'égalisation 345
11.1.3 Combiner égalisation et décodage 349
11.1.4 Principe de la turbo-égalisation 351
11.1.5 La turbo-égalisation *MAP* 354
11.1.6 La turbo-égalisation MEQM 363
11.2 La turbo-détection multi-utilisateurs et son application aux systèmes CDMA 378
11.2.1 Introduction et quelques notations 378
11.2.2 Détection multi-utilisateurs 379
11.2.3 Turbo-CDMA 384
11.3 Conclusions 388
11.4 Bibliographie 389

Index 394

Chapitre 1

Introduction

Redondance, diversité et parcimonie sont les mots clés du codage correcteur d'erreurs. Du côté du décodage, il s'y ajoute l'efficacité, c'est-à-dire le souci de tirer le meilleur parti de toutes les informations disponibles. Pour illustrer ces concepts, considérons une situation simple de la vie courante.

Deux personnes se parlent près d'une route où la circulation est assez intense. Le bruit des moteurs gêne plus ou moins la conversation, avec des pics de perturbation sonore correspondant au passage des véhicules. Supposons dans un premier temps que l'une des personnes émette régulièrement une lettre tirée au hasard : « a », « b » ... ou l'une quelconque des 26 lettres de l'alphabet, avec une même probabilité (soit 1/26). Le message ne contient aucune information rajoutée et il n'existe aucun lien entre les sons émis. L'auditeur, s'il ne lit pas sur les lèvres du locuteur, sera certainement souvent embarrassé pour reconnaître certaines lettres. Il y aura donc des erreurs de transmission.

Maintenant, dans un autre cas de figure, l'une des deux personnes énonce des phrases complètes, portant sur un sujet bien précis, le temps qu'il fait par exemple. Malgré le bruit, l'auditeur comprend mieux ce que dit son interlocuteur que dans le cas des lettres isolées, parce que le message comporte de la *redondance*. Les mots ne sont pas indépendants et les syllabes elles-mêmes ne sont pas concaténées aléatoirement. On sait par exemple qu'après un sujet apparaît généralement un verbe, on devine qu'après « nua », il y aura « ge » même si on l'entend mal, etc. Cette redondance dans la construction du message permet à l'auditeur de mieux le comprendre malgré les conditions difficiles de sa transmission.

Supposons que l'on veuille encore améliorer la qualité de la transmission, dans cette conversation qui va prendre une tournure particulière. Pour être plus sûr d'être compris, le locuteur répète certains de ses mots, par exemple « matin matin ». Toutefois, après la double transmission, le récepteur a compris « matin satin ». Il y a de toute évidence une erreur quelque part mais est-ce « matin » ou « satin » que le récepteur doit retenir ? Aucune correction d'erreur n'est possible en utilisant cette technique de répétition, sauf éventuellement à

émettre le mot plus de deux fois. « matin satin matin » pourra, sans grand risque d'erreur, être traduit en « matin ».

Un codage plus élaboré consiste à transmettre le message original accompagné, non plus du même mot mais d'un synonyme ou d'un équivalent : « matin aube » par exemple. Si l'on reçoit « matin auge » ou « latin aube », la correction est possible, en se référant à un dictionnaire d'équivalence. La règle de décodage serait alors la suivante : en cas d'erreur (si les deux mots reçus ne sont pas directement équivalents) et si l'on trouve deux équivalents en modifiant une lettre, au plus, dans le message reçu, alors la correction est adoptée (« auge » devient « aube ») et le premier des deux mots (« matin ») est accepté comme message original. Il en serait de même avec la réception de « latin aube », où la correction porterait cette fois sur le premier mot. Bien sûr, si de nombreuses erreurs altèrent la transmission et si l'on reçoit « latin auge », on n'y comprendra probablement plus rien. Il existe ainsi une limite au pouvoir de correction. C'est la fameuse *limite de Shannon*, qu'aucun code correcteur ne peut en théorie dépasser.

Par rapport à la simple répétition, le codage par équivalence, plus efficace, utilise un effet de *diversité*. Dans cette analogie de la conversation, cette diversité est portée par une propriété lexicographique : deux mots distincts, lorsqu'ils sont voisins dans leur orthographe (matin et satin), ont peu de chance d'avoir deux équivalents (« aube » et « tissu » par exemple) également proches dans leur orthographe. La diversité, telle qu'elle vient d'être introduite, consiste donc à construire un message redondant sous une forme qui minimise, en cas d'erreurs, les ambiguïtés en réception. On l'appelle aussi diversité temporelle car les mots équivalents dans le message sont transmis à des instants différents et subissent des perturbations d'intensités inégales. Par exemple, « matin » et « aube » peuvent être émis lors du passage d'une moto et d'une bicyclette, respectivement.

Dans les systèmes de télécommunications, des effets de diversité complémentaires peuvent être recherchés. La diversité fréquentielle consiste à découper et envoyer le message dans des bandes de fréquences qui ne sont pas, à un même instant, perturbées de la même manière. L'utilisation de plusieurs antennes d'émission et/ou de réception offre, quant à elle, de la diversité spatiale car les trajets entre antennes n'ont pas le même comportement. L'exploitation conjointe des trois types de diversité : temporelle, fréquentielle et spatiale, conduit à des systèmes de communications très performants.

Enfin, le souci de *parcimonie,* ou d'économie, est imposé par la limitation des ressources, temporelles ou fréquentielles, de la transmission. Le choix de « matin aube » est ainsi certainement plus judicieux, du point de vue de la concision, que « matin potron-minet ». En contre-partie, on pressent que le dernier message pourrait être plus résistant aux erreurs multiples car plus redondant (la réception et la résolution de « mutin potion-minet » n'est pas problématique si l'on utilise la loi de décodage énoncée plus haut et en étendant le pouvoir de correction à deux erreurs). La recherche de performance, à travers le taux de redondance et la contrainte de parcimonie s'opposent donc complètement.

Le codage redondant est en général simple à mettre en œuvre et le logiciel ou le matériel correspondant est de faible complexité. Le décodage, en revanche, fait appel à des techniques calculatoires qui peuvent être lourdes. même si, en définitive, le nombre d'instructions du programme (typiquement quelques centaines, en langage informatique de haut niveau) ou la surface occupée sur silicium (typiquement quelques millimètres carrés) reste modeste.

1.1 Messages numériques

Un message numérique est une suite de caractères ou symboles prélevés dans un alphabet de taille finie. L'information génétique portée par l'ADN, pour emprunter un exemple à la nature, utilise un alphabet de quatre caractères, notés A, T, G et C, d'après les initiales de leurs bases azotées (adénine, thymine, guanine, cytosine). La première technique numérique de transmission fut le Morse (1832), avec son alphabet sonore à deux caractères : le *TIT* ou point, son court de quatre centièmes de seconde et le *TAT* ou tiret, son long de douze centièmes de seconde. Samuel F. B. Morse aurait tout aussi bien pu appeler ces caractères 0 et 1, qui sont aujourd'hui les dénominations universelles de tout alphabet à deux éléments ou alphabet binaire. Les éléments binaires, 0 et 1, ont été baptisés *bits* par J. W. Tukey (1943), en tant que contraction de *binary digit* et après avoir renoncé à *bigit* et *binit*. Shannon a repris le néologisme à son compte lorsqu'il lui fallut introduire le concept de mesure de l'information. Aujourd'hui, l'unité de mesure de l'information est préférablement appelée le *shannon*, pour la distinguer du bit, qui a pris une signification plus électronique.

Un alphabet possédant M symboles est dit alphabet M-aire. Il peut être transcrit en un alphabet binaire en représentant chacun des M symboles par un mot de m bits, avec :

$$\begin{array}{ll} & m = \left\lfloor \log_2(M) \right\rfloor + 1 \quad \text{si } M \text{ n'est pas une puissance de 2} \\ \text{ou :} & \\ & m = \log_2(M) \qquad\quad \text{si } M \text{ est une puissance de 2} \end{array} \tag{1.1}$$

où $\lfloor x \rfloor$ désigne la partie entière de x. Les messages multimédia (voix, musique, images fixes et animées, texte etc.) transitant par les systèmes de communication ou stockés dans les mémoires de masse, sont exclusivement binaires. Cependant, nous aurons parfois à considérer dans cet ouvrage des alphabets à plus de deux éléments. Ce sera le cas au chapitre 4, pour introduire certains codes algébriques. Dans les chapitres 2 et 10, qui traitent des modulations, les alphabets, qu'on appelle alors plus concrètement constellations, contiennent un nombre de symboles qui est une puissance de 2, c'est-à-dire que l'on a précisément : $m = \log_2(M)$.

Les techniques de codage correcteur ne sont mises en œuvre que sur des messages numériques. Rien n'interdit cependant de construire un message analogique redondant. Par exemple, un signal analogique, dans sa dimension temporelle, accompagné ou suivi de sa représentation fréquentielle obtenue grâce

à la transformation de Fourier, réalise un codage judicieusement redondant. Cependant cette technique n'est pas simple et le décodeur reste à imaginer...

Par ailleurs, les messages numériques que l'on considérera dans la suite, avant que soit effectuée l'opération de codage, seront supposés être constitués d'éléments binaires mutuellement indépendants et prenant les valeurs 0 et 1 avec la même probabilité, soit 1/2. Les signaux délivrés par un capteur tel qu'un microphone ou une caméra et qui sont ensuite numérisés pour devenir des séquences binaires, ne satisfont généralement pas ces propriétés d'indépendance et d'équiprobabilité. Il en est de même avec le texte (par exemple, la récurrence du « e » dans un texte français est en moyenne 15 fois supérieure à celle du « f »). Les effets de dépendance ou de disparité dans le message original, qu'ils soient d'origine physique, orthographique, sémantique ou autre, ne sont pas exploitables par le système de communication numérique, qui transmet des 0 et des 1 indépendamment de leur contexte. Pour transformer le message original en un message remplissant les conditions d'indépendance et d'équiprobabilité, une opération dite de codage de source, ou compression numérique, peut être effectuée. Les normes de compression comme JPEG, MPEG, ZIP, MUSICAM... sont aujourd'hui bien intégrées dans le monde des télécommunications, internet en particulier. À la sortie du codeur de source, les propriétés statistiques d'indépendance et d'équiprobabilité sont généralement respectées et le message comprimer peut alors être traité par le codeur de canal, qui ajoutera de la redondance mathématiquement exploitable par le récepteur.

1.2 Un premier code

La figure 1.1 représente un circuit électronique qui réalise un codage correcteur d'expression mathématique très simple et aisément décodable. Le code mis en œuvre est un code de Hamming étendu (en anglais *extended*), qui a été utilisé dans le télétexte (l'un des premiers systèmes de communication numérique) et dont il est aussi question dans le chapitre 4.

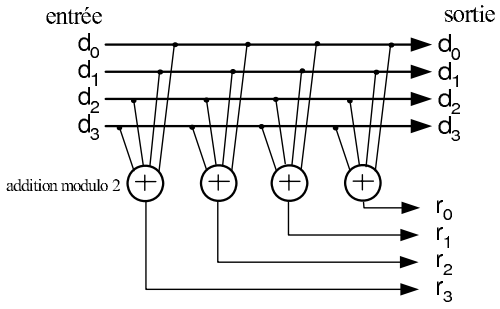


Figure 1.1 – Codeur de Hamming étendu : un code simple et aisément décodable.

Le codeur contient quatre opérateurs identiques réalisant la fonction ou-exclusif, c'est-à-dire l'addition modulo 2. Le ou-exclusif (XOR) de k valeurs binaires b_0, b_1, ..., b_{k-1} est calculé comme :

$$\text{XOR}(b_0, b_1, ..., b_{k-1}) = b_0 \oplus b_1 \oplus ... \oplus b_{k-1} = \sum_{p=0}^{k-1} b_p \text{ modulo } 2 \qquad (1.2)$$

C'est donc tout simplement 0 si le nombre des uns logiques apparaissant dans la séquence considérée est pair et 1 dans le cas contraire. Dans la suite, lorsque des additions modulo 2 porteront sur des valeurs binaires et s'il n'y a pas d'ambiguïté dans la notation, le terme modulo 2 pourra être éludé. Le mot somme devra être lui-même être compris dans le sens d'une addition modulo 2.

Le codeur transforme le message contenant quatre bits de données : $\mathbf{d} = (d_0, d_1, d_2, d_3)$ en un mot de huit bits : $\mathbf{c} = (d_0, d_1, d_2, d_3, r_0, r_1, r_2, r_3)$, appelé *mot de code*. Le mot de code est donc *séparable* en une partie qui est l'information provenant de la source, dite partie *systématique*[1] et une partie ajoutée par le codeur, dite partie *redondante*. Tout code produisant des mots de code sous cette forme est dit code systématique. La plupart des codes, en pratique, sont systématiques mais il existe une exception importante dans la famille des codes convolutifs (chapitre 5).

La loi de construction de la partie redondante, par le codeur particulier de la figure1.1, peut être écrite simplement sous la forme :

$$r_j = d_j \oplus \sum_{p=0}^{3} d_p \qquad (j = 0, ..., 3) \qquad (1.3)$$

L'addition modulo 2 de d_j a pour effet de l'éliminer de sorte que r_j est la somme de les bits de données, d_j excepté. La table 1.1 fournit les seize valeurs possibles de $\mathbf{c}$, c'est-à-dire l'ensemble $\{\mathbf{c}\}$ des mots de code.

Observons en premier lieu que la loi de codage est *linéaire* : la somme de deux mots de code est aussi un mot de code. C'est la linéarité de la relation (1.3) qui assure la linéarité du codage. Tous les codes que nous considérerons dans la suite sont linéaires car tous fondés sur deux opérations linéaires : l'addition et la permutation (incluant le décalage). Puisque le code est linéaire et que la transmission d'un mot de code est éventuellement affectée par un processus également linéaire (l'addition d'une perturbation : bruit, interférence, ...), le choix d'un mot de code, pour expliquer ou justifier les propriétés du code, est complètement indifférent. C'est le mot de code « tout 0 » qui tiendra ce rôle de « représentant » ou de référence pour tous les mots de code, vis-à-vis des propriété générales du couple codeur/décodeur. En réception, la présence des uns sera donc représentative d'erreurs de transmission.

[1] On dira aussi partie d'information car constituée des bits d'information provenant de la source.

d_0	d_1	d_2	d_3	r_0	r_1	r_2	r_3	d_0	d_1	d_2	d_3	r_0	r_1	r_2	r_3
0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1
0	0	0	1	1	1	1	0	1	0	0	1	1	0	0	1
0	0	1	0	1	1	0	1	1	0	1	0	1	0	1	0
0	0	1	1	0	0	1	1	1	0	1	1	0	1	0	0
0	1	0	0	1	0	1	1	1	1	0	0	1	1	0	0
0	1	0	1	0	1	0	1	1	1	0	1	0	0	1	0
0	1	1	0	0	1	1	0	1	1	1	0	0	0	0	1
0	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1

Table 1.1 – Les seize mots de code que peut produire le codeur de la figure 1.1.

Le nombre de 1 contenus dans un mot de code qui n'est pas « tout 0 » est appelé poids de Hamming et est noté w_H. On pourra distinguer le poids relatif à la partie systématique ($w_{H,s}$) et le poids relatif à la partie redondante ($w_{H,r}$). On remarque, dans la table 1.1, que w_H est au moins égal à 4. Du fait de la linéarité, ceci signifie aussi que le nombre de bits qui diffèrent dans deux mots de code est aussi au moins égal à 4. Le nombre de bits qui sont différents, quand on compare deux mots binaires, est appelée distance de Hamming. La plus petite de toutes les distances entre tous les mots de code, considérés deux à deux, est appelée distance minimale de Hamming (DMH) et notée $d_{\min}$. La linéarité impose que c'est aussi le plus faible des poids de Hamming, dans la liste des mots de code en dehors du « tout 0 ». La distance minimale $d_{\min}$ est un paramètre essentiel pour caractériser un code particulier car le pouvoir de correction du décodeur correspondant lui est directement lié.

Édictons maintenant une loi de décodage pour le code de la figure 1.1 : « après réception du mot **c'** = (d_0', d_1', d_2', d_3', r_0', r_1', r_2', r_3') émis par le codeur et éventuellement altéré durant la transmission, le décodeur choisit le mot de code $\hat{\mathbf{c}}$ qui lui est le plus proche, au sens de la distance de Hamming ». La tâche du décodeur est donc de parcourir la table 1.1 et, pour chacun des seize mots de code possibles, de compter le nombre de bits qui diffèrent de **c**'. Le mot de code retenu $\hat{\mathbf{c}}$ est celui qui diffère le moins de **c**'. Lorsque plusieurs solutions sont possibles, le décodeur en retient une au hasard. Mathématiquement, cela s'écrit :

$$\hat{\mathbf{c}} = \mathbf{c} \in \{\mathbf{c}\} \text{ tel que } \sum_{j=0}^{3} d_j \oplus d_j^{'} + \sum_{j=0}^{3} r_j \oplus r_j^{'} \text{ est minimum} \qquad (1.4)$$

Un décodeur capable de mettre en œuvre un tel procédé est dit décodeur à *maximum de vraisemblance* (MV) car tous les cas de figure, ici les seize mots de code possibles, sont passés en revue dans la recherche d'une solution et il n'existe pas de procédé de décodage plus efficace. Avec des codes autres que le très simple code de Hamming, le message à coder contient bien plus que quatre bits (en pratique, cela va de quelques dizaines à quelques dizaines de milliers

de bits) et le décodage à MV est impossible à exécuter car le nombre de mots de code est trop important.

Supposons que le codeur de Hamming émette le mot « tout 0 » que nous avons choisi comme référence et que certains des 0 aient été inversés avant réception, sur le *canal de transmission*. Combien d'erreurs le décodeur s'appuyant sur la loi (1.4) peut-il corriger ? Si $\mathbf{c}$' contient un seul 1 (une seule erreur), le mot « tout 0 » est plus proche de $\mathbf{c}$' que tout autre mot de code qui possède au moins quatre 1. La correction est donc possible. Si $\mathbf{c}$' contient deux 1, par exemple aux deux premières places ($d_0 = d_1 = 1$), le décodeur fait face à une indétermination : quatre mots de code sont à la distance 2 du mot reçu. Il doit donc tirer $\hat{\mathbf{c}}$ au hasard, parmi les quatre solutions possibles, avec le risque de se tromper trois fois sur quatre. Dans cette situation, on peut également faire en sorte que le décodeur ne délivre pas de solution mais indique simplement qu'il est confronté à une indétermination : il joue alors le rôle de *détecteur d'erreurs* non corrigibles. Enfin, si $\mathbf{c}$' contient trois 1, le décodeur trouvera un seul mot de code à la distance 1 et il proposera ce mot de code comme solution la plus probable mais il sera faux.

Le pouvoir de correction du code de Hamming étendu est donc de $t = 1$ erreur. Plus généralement, le pouvoir de correction d'un code ayant une distance minimale $d_{\min}$ est :

$$t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor \tag{1.5}$$

Notons que le pouvoir de correction du code pris en exemple dans cette introduction n'est pas diminué si l'on supprime l'un (n'importe lequel) des symboles de redondance. La DMH passe de 4 à 3 mais le pouvoir de correction est toujours d'une erreur. Cette version tronquée est en fait le code de Hamming original, le premier code correcteur dans l'histoire de la théorie de l'information (1948).

Dans une famille de codes donnée, on en décrit une version particulière par le raccourci (n, k, $d_{\min}$) où n et k sont, respectivement, les longueurs des mots de code et des messages de source. Nous venons donc de définir deux codes de Hamming notés (8, 4, 4) et (7, 4, 3). Le second semble plus intéressant car il offre le même pouvoir de correction ($t = 1$) avec un taux de redondance $\tau = \frac{n-k}{k}$ de 0,75, au lieu de 1 pour le premier. En revanche, le code (7, 4, 3) ne peut tenir le rôle de détecteur d'erreurs : si le mot reçu contient deux erreurs, le décodeur décidera en faveur du seul mot de code, erroné, qui se trouve à une distance de Hamming de 1.

Plutôt que de taux de redondance, on préfèrera le plus souvent utiliser la notion de rendement de codage, noté R, et défini par :

$$R = \frac{k}{n} = \frac{1}{1+\tau} \tag{1.6}$$

Le produit $Rd_{\min}$ apparaîtra dans la suite comme un facteur de mérite essentiel vis-à-vis d'une perturbation apportée par un bruit additif à distribution gaussienne.

1.3 Décodage à entrée ferme et décodage à entrée souple

Poursuivons cette présentation des premiers principes du codage et du décodage en continuant de nous appuyer sur l'exemple du code de Hamming étendu.

Le principe de décodage défini par (1.4) suppose que le mot de code reçu $\mathbf{c}'$ est composé de valeurs binaires, c'est-à-dire que la transmission des données s'est effectuée selon la loi de perturbation donnée par le diagramme de la figure 1.2. Un 0 devient un 1 et vice-versa, avec une probabilité p et les valeurs binaires sont correctement reçues avec la probabilité complémentaire 1 - p. Un tel milieu de transmission est dit *canal binaire symétrique* et le décodeur réalise ce que l'on appelle un *décodage à entrée ferme* (*hard input decoding* en anglais). Dans certains systèmes de communications (fibres optiques, réseaux commutés...) et dans la plupart des matériels de stockage, les décodeurs ne peuvent effectivement exploiter que des informations binaires.

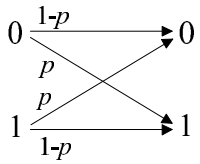


Figure 1.2 – Le canal binaire symétrique avec probabilité d'erreur p.

Lorsque le signal reçu par le décodeur provient d'un dispositif capable de délivrer des estimations de la fiabilité avec laquelle les données reçues représentent les données émises, le pouvoir de correction du décodeur peut s'en trouver nettement amélioré. Pour le démontrer sur l'exemple du code de Hamming étendu, il nous faut préalablement changer l'alphabet et adopter une écriture binaire antipodale (ou symétrique). Aux données binaires systématiques $d = 0$ et $d = 1$, nous ferons correspondre respectivement les valeurs émises $x = -1$ et $x = +1$. De même, aux données binaires redondantes $r = 0$ et $r = 1$, nous ferons correspondre respectivement les valeurs émises $y = -1$ et $y = +1$. On aura donc :

$$\begin{aligned} x &= 2d - 1 = -(-1)^d \\ y &= 2r - 1 = -(-1)^r \end{aligned} \tag{1.7}$$

La figure 1.3 donne un exemple de transmission durant laquelle les valeurs émises -1 et $+1$ sont altérées par un bruit additif de nature analogique. Les valeurs en sortie du canal de transmission sont alors des variables réelles, qui doivent être en pratique écrêtées puis quantifiées[2] lorsque le décodeur est un processeur numérique. Le nombre de bits de quantification, noté N_q, n'a pas

[2] Cette opération s'effectue à l'aide d'un échantillonneur et d'un convertisseur analogique-numérique.

besoin d'être élevé : 3, 4 ou 5 bits suffisent pour représenter finement les échantillons analogiques. Une fois sur deux en moyenne, le bruit est favorable car il est de même signe que la valeur émise. Dans l'autre cas, l'amplitude du signal est atténuée et, lorsque ce bruit défavorable est important, le signe peut s'inverser. Une prise de décision immédiate par seuil (c'est-à-dire est-ce plus grand ou plus petit que le zéro analogique ?) conduirait alors à délivrer une valeur binaire erronée.

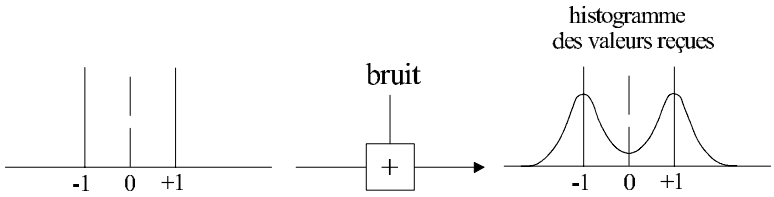


Figure 1.3 – Canal de transmission avec bruit additif de nature analogique.

Puisque le décodeur dispose d'une information sur le degré de fiabilité des valeurs reçues, appelées dans la suite valeurs *souples* ou *pondérées*, le décodage du code de Hamming étendu selon la loi (1.4) n'est plus optimal. La loi de décodage à maximum de vraisemblance à mettre en œuvre pour exploiter ces valeurs pondérées dépend du type de bruit. Un cas de figure important en pratique est le bruit additif blanc gaussien (BABG) représentatif du bruit thermique dans les composants électroniques du récepteur.

U est une variable aléatoire gaussienne de moyenne μ et de variance σ^2 lorsque sa densité de probabilité $p(u)$ s'exprime sous la forme :

$$p(u) = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(u-\mu)^2}{2\sigma^2}) \tag{1.8}$$

Le BABG est une perturbation qui produit, après filtrage adapté et échantillonnage périodique (voir chapitre 2), des échantillons indépendants dont l'amplitude suit la loi de densité de probabilité (1.8)de moyenne nulle et de variance :

$$\sigma^2 = \frac{N_0}{2} \tag{1.9}$$

où N_0 est la densité spectrale de puissance du bruit.

Un canal de transmission sur lequel la seule altération du signal provient d'un BABG est appelé *canal gaussien*. À la sortie d'un tel canal, le décodage à MV s'appuie sur la recherche exhaustive du mot de code qui est à *la plus petite distance euclidienne* du mot reçu. En présence de BABG, il peut être montré que ce critère minimise la probabilité de retenir un mot de code erroné.
En notant X et Y les valeurs reçues correspondant aux symboles émis x et y respectivement, le décodeur à entrées pondérées du code de Hamming étendu

sélectionne donc :

$$\hat{\mathbf{c}} = \mathbf{c} \in \{\mathbf{c}\} \text{ tel que } \sum_{j=0}^{3} (x_j - X_j)^2 + \sum_{j=0}^{3} (y_j - Y_j)^2 \text{ est minimum} \qquad (1.10)$$

Puisque les valeurs émises sont toutes telles que $x_j^2 = 1$ ou $y_j^2 = 1$ et que toutes les distances euclidiennes contiennent X_j^2 et Y_j^2, la loi précédente peut se simplifier sous la forme :

$$\hat{\mathbf{c}} = \mathbf{c} \in \{\mathbf{c}\} \text{ tel que } -\sum_{j=0}^{3} 2x_j X_j - \sum_{j=0}^{3} 2y_j Y_j \text{ est minimum}$$

soit encore :

$$\hat{\mathbf{c}} = \mathbf{c} \in \{\mathbf{c}\} \text{ tel que } \sum_{j=0}^{3} x_j X_j + \sum_{j=0}^{3} y_j Y_j \text{ est maximum} \qquad (1.11)$$

Minimiser la distance euclidienne entre deux mots de code **c** et **c**' revient donc à maximiser le *produit scalaire* $\langle \mathbf{x}, \mathbf{X} \rangle + \langle \mathbf{y}, \mathbf{Y} \rangle = \sum_{j=0}^{3} x_j X_j + \sum_{j=0}^{3} y_j Y_j$ où **x**, **X**, **y** et **Y** représentent les séquences émises et reçues des parties systématique et redondante.

Dans les processeurs de traitement de signal (*DSP* : *Digital Signal Processor*) ou dans les circuits intégrés spécifiques (*ASIC* : *Application Specific Integrated Circuit*), il peut être commode de n'avoir à manipuler que des nombres positifs. La loi (1.11) pourra alors être mise en œuvre sous la forme :

$$\hat{\mathbf{c}} = \mathbf{c} \in \{\mathbf{c}\} \text{ tel que } \sum_{j=0}^{3} (V_{max} + x_j X_j) + (V_{\max} + y_j Y_j) \text{ est maximum} \qquad (1.12)$$

où $[-V_{\max}, V_{\max}]$ est l'intervalle des valeurs que peuvent prendre les échantillons d'entrée X_j et Y_j du décodeur après l'opération d'écrêtage.

En figure 1.4, le mot de code « tout 0 » a été émis et reçu avec trois altérations aux trois premières positions. Ces trois altérations ont inversé les signes des symboles mais leurs amplitudes sont plutôt de faible niveau : $0,2$, $0,4$ et $0,1$. Le décodage à entrée ferme produit un résultat erroné car le mot de code le plus proche en terme de distance de Hamming est $(1,1,1,0,0,0,0,1)$. En revanche, le décodage à entrée souple selon (1.11) délivre bien le mot « tout 0 », auquel correspond le produit scalaire maximum :

$$\begin{aligned} &(-1)(0,2) + (-1)(0,4) + (-1)(0,1) + (-1)(-1) \\ &+ (-1)(-1) + (-1)(-1) + (-1)(-1) + (-1)(-1) \quad = \quad 4,3 \end{aligned}$$

à comparer avec :

$$\begin{aligned} &(+1)(0,2) + (+1)(0,4) + (+1)(0,1) + (-1)(-1) \\ &+ (-1)(-1) + (-1)(-1) + (-1)(-1) + (+1)(-1) \quad = \quad 3,7 \end{aligned}$$

pour le mot concurrent $(1, 1, 1, 0, 0, 0, 0, 1)$. Cet exemple simple illustre l'intérêt de conserver, lorsque cela est possible, les informations de fiabilités dans les prises de décision.

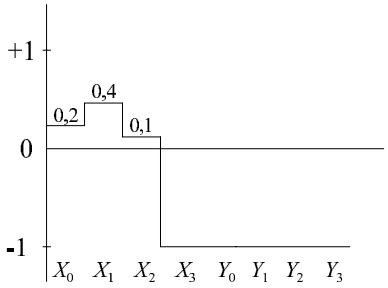


Figure 1.4 – Le mot « tout 0 » (émission de symboles à la valeur -1) a été altéré pendant la transmission sur les trois premières positions. Le décodage à entrée ferme est erroné mais pas le décodage à entrée souple.

Les règles de décodage à MV que nous venons de poser sur un exemple particulier sont aisément généralisables. Cependant, on conçoit bien qu'au-delà d'une certaine longueur de message, un tel principe de décodage est irréaliste. Appliquer le décodage à MV sur des mots de code contenant 240 bits de partie systématique, par exemple, reviendrait à considérer autant de mots de code que d'atomes dans l'univers visible (2^{240}). Malgré cela, pour la plupart des codes connus, des méthodes de décodage non exhaustives ont été mises au point, qui permettent d'approcher de très près du résultat optimal de la méthode à MV.

1.4 Décodage à sortie ferme et décodage à sortie souple

Lorsque la sortie du décodeur n'est pas directement transmise à un destinataire mais doit être utilisée par un autre processeur dont la performance est améliorée grâce à des entrées pondérées, il peut être demandé à ce décodeur en amont d'élaborer de telles valeurs pondérées. On distinguera donc *sortie ferme* (en anglais *hard output*) quand le décodeur délivre des 0 et 1 logiques et *sortie pondérée* ou *souple* (en anglais *soft output*). Dans ce dernier cas, le décodeur accompagne ses décisions binaires de mesures de fiabilité ou poids. L'échelle de sortie des valeurs pondérées est en général la même que l'échelle d'entrée $[-V_{\max}, V_{\max}]$.

Pour le décodeur du code de Hamming étendu, il est relativement facile de construire des décisions pondérées. Lorsque celui-ci a calculé les seize produits scalaires, il les classe par valeur décroissante. En première position, on trouve le produit scalaire du mot de code le plus vraisemblable, c'est-à-dire celui qui décide des signes des décisions pondérées en sortie. Puis, pour chacun des quatre bits de la partie systématique, le décodeur recherche le produit scalaire le plus

élevé qui corresponde à un mot de code concurrent dans lequel le bit d'information en question est opposé à celui de la décision binaire. Le poids associé à cette décision binaire est alors la différence du produit scalaire maximum et du produit scalaire correspondant au mot concurrent. Une division par 2 supplémentaire met la sortie pondérée à l'échelle de l'entrée. Ce procédé est optimal pour une perturbation à BABG. En reprenant l'exemple de la figure 1.4 (qui n'est pas typique d'un BABG), les poids associés aux décisions sur les trois premiers bits seraient identiques et égaux à $(4{,}3 - 3{,}7)/2 = 0{,}3$.

D'un point de vue historique, les premières méthodes de décodage ont été du type entrée et sortie fermes. C'est l'algorithme de Viterbi, détaillé dans le chapitre 5, qui a popularisé l'idée de décodage à entrée souple. Puis les turbocodes, qui sont décodés par traitements répétés et qui requièrent des valeurs pondérées à tout niveau de ce traitement, ont mis à la mode les décodeurs à entrée et sortie souples. Le sigle générique pour qualifier de tels décodeurs est de langue anglaise : *SISO* pour *Soft-Input/Soft-Output*.

1.5 La mesure de performance

La performance d'un couple codeur/décodeur se juge en premier lieu en termes d'erreurs résiduelles à la sortie du décodeur, lorsqu'on s'est fixé un cadre bien précis d'évaluation : type de perturbation, longueur de message, taux de redondance ou rendement de codage etc. D'autres aspects, comme la complexité du décodage, les latences introduites par le codeur et le décodeur, le degré de flexibilité du code (en particulier son aptitude à se conformer à différentes longueurs de message et/ou à différents rendements de codage) sont également à considérer de plus ou moins près suivant les contraintes propres au système de communication.

Les erreurs résiduelles que le décodeur n'est pas parvenu à corriger se mesurent à l'aide de deux paramètres. Le taux d'erreurs binaires (TEB) est le rapport entre le nombre d'erreurs binaires résiduelles et le nombre total de bits d'information transmis. Le taux d'erreurs de mots, de blocs ou de paquets (TEP) est le nombre de mots de code mal décodés (au moins un des bits d'information est faux) ramené au nombre total de mots de code émis. Le rapport entre TEB et TEP est la densité d'erreurs moyenne δ_e dans la partie systématique d'un mot mal décodé :

$$\delta_e = \frac{\bar{w}}{k} = \frac{\text{TEB}}{\text{TEP}} \tag{1.13}$$

où $\bar{w} = k\delta_e$ est le nombre moyen de bits d'information erronés dans la partie systématique d'un bloc mal décodé.

La figure 1.5 donne un exemple typique de représentation graphique de performance de codage et décodage correcteur d'erreurs. L'ordonnée donne le TEB en échelle logarithmique et l'abscisse porte le rapport signal à bruit $\frac{E_b}{N_0}$, exprimé en décibels (dB). N_0 est défini par (1.9) et E_b est l'énergie reçue par

bit d'information. Si E_s est l'énergie reçue pour chacun des symboles du mot de code, E_s et E_b sont reliés par :

$$E_s = RE_b \tag{1.14}$$

La comparaison de procédés de codage et de décodage différents ou la variation de performance d'un procédé particulier avec le rendement de codage s'établissent toujours à même énergie globale de réception. Quand il n'y a pas de codage, l'énergie par mot de code reçu est kE_b. En présence d'un codage, qui augmente le nombre de bits transmis, l'énergie kE_b est à répartir entre les n bits du mot de code, ce qui justifie la relation (1.14). La référence d'énergie à considérer, indépendante du code et du rendement, est donc E_b.

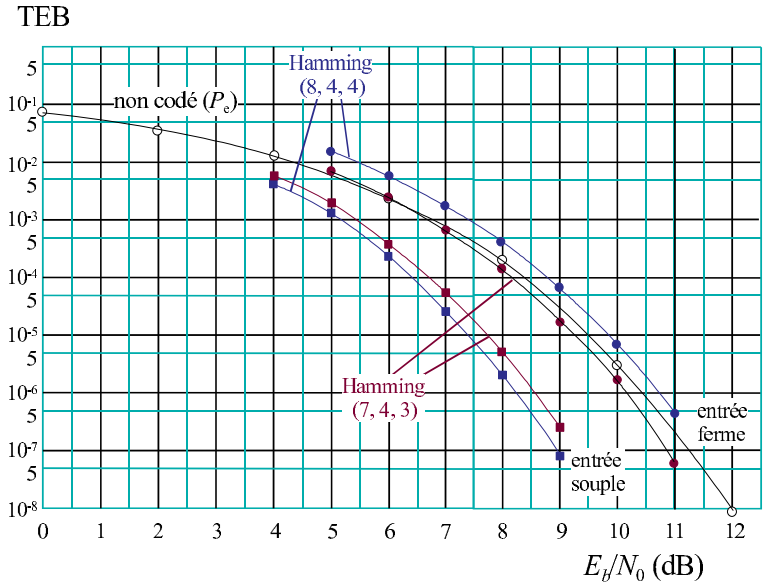


Figure 1.5 – Pouvoir de correction d'erreurs des codes de Hamming (8, 4, 4) et (7, 4, 3) sur canal gaussien, avec décodage à entrée ferme et décodage à entrée souple. Alors que le code (7,4,3) est plus performant lorsque l'entrée est ferme, le code (8,4,4) est préférable lorsque l'entrée est souple.

Sur la figure 1.5, sont tracées les courbes de correction d'erreurs des codes de Hamming (8, 4, 4) et (7, 4, 3) dont il a été question tout au long de cette introduction, sur un canal gaussien. Le décodage à entrée ferme selon (1.4) et le décodage à entrée souple selon (1.11) sont considérés. Dans le diagramme est aussi représentée la courbe de probabilité d'erreur binaire P_e que l'on obtient sur ce canal sans utiliser de codage[3]. Cette courbe est reliée à la fonction d'erreur complémentaire erfc(x) donnée par la relation (2.66) du chapitre 2 :

[3] La distinction entre P_e et TEB n'est que formelle : la valeur de P_e est donnée par une équation tandis que le TEB est obtenu par mesure ou simulation.

$P_e = \frac{1}{2}\text{erfc}\sqrt{\frac{E_b}{N_0}}$. Pour E_b/N_0 grand, le *comportement asymptotique* de P_e est approché par :

$$P_e \approx \frac{1}{2}\frac{\exp\left(-\frac{E_b}{N_0}\right)}{\sqrt{\pi\frac{E_b}{N_0}}} \tag{1.15}$$

Pour évaluer la probabilité $P_{e,\text{mot}}$ que le décodeur à entrée pondérée d'un code de rendement R et de distance minimale $d_{\min}$ délivre un mot de code erroné, on remplace dans la formule précédente $\frac{E_b}{N_0}$ par $Rd_{\min}\frac{E_b}{N_0}$ et on affecte le tout d'un coefficient noté $N(d_{\min})$:

$$P_{e,\text{mot}} = \frac{1}{2}N(d_{\min})\text{erfc}\sqrt{Rd_{\min}\frac{E_b}{N_0}} \approx \frac{1}{2}N(d_{\min})\frac{\exp\left(-Rd_{\min}\frac{E_b}{N_0}\right)}{\sqrt{\pi Rd_{\min}\frac{E_b}{N_0}}} \tag{1.16}$$

Le remplacement de E_b par RE_b provient de (1.14) car l'énergie reçue par symbole est E_s. La multiplication par $d_{\min}$ s'explique par la loi de décodage à MV (relation (1.11)), qui permet qu décodeur de discriminer le mot de code présumé correct de ses concurrents les plus proches parce qu'il en diffère en $d_{\min}$ positions au moins. Enfin, le coefficient $N(d_{\min})$, appelé *multiplicité*, prend en compte le nombre de mots de code concurrents qui sont à la distance minimale. Par exemple, dans le cas du code de Hamming étendu (voir table 1.1), on a $N(d_{\min} = 4) = 14$.

Pour obtenir la probabilité d'erreur binaire $P_{e,\text{bit}}$ du système codé, il suffit de multiplier $P_{e,\text{mot}}$ par la densité d'erreurs moyenne δ_e définie par (1.13) :

$$P_{e,\text{bit}} \approx \frac{1}{2}\delta_e N(d_{\min})\frac{\exp\left(-Rd_{\min}\frac{E_b}{N_0}\right)}{\sqrt{\pi Rd_{\min}\frac{E_b}{N_0}}} \tag{1.17}$$

À la lecture de la table 1.1, on observe que le nombre moyen d'erreurs dans les 14 mots de poids 4, concurrents à la distance minimale du mot « tout 0 », est 2. La formule (1.17) appliquée au code de Hamming étendu est donc :

$$P_{e,\text{bit}} \approx \frac{1}{2}\times\frac{2}{4}\times 14\frac{\exp\left(-\frac{1}{2}\times 4\times\frac{E_b}{N_0}\right)}{\sqrt{\pi\frac{1}{2}\times 4\times\frac{E_b}{N_0}}} = 3,5\frac{\exp\left(-2\frac{E_b}{N_0}\right)}{\sqrt{2\pi\frac{E_b}{N_0}}}$$

Cette expression fournit $P_{e,\text{bit}} = 2,8.10^{-5}$, $1,8.10^{-6}$ et $6,2.10^{-8}$ pour $\frac{E_b}{N_0} = 7$, 8 et 9 dB respectivement, ce qui correspond aux résultats de la simulation de la figure 1.5. Une telle concordance entre équations et expérience ne se retrouve pas aussi nettement pour des codes plus complexes. En particulier, recenser les mots de code concurrents à la distance $d_{\min}$ peut ne pas suffire et il faut alors considérer les mots à la distance $d_{\min}+1$, $d_{\min}+2$, etc.

Pour une même valeur de P_e et $P_{e,\text{bit}}$ respectivement fournies par les relations (1.15) et (1.17), les rapports signal à bruit $\left.\frac{E_b}{N_0}\right|_{NC}$ et $\left.\frac{E_b}{N_0}\right|_{C}$ sans codage (NC) et avec codage (C) sont tels que :

$$Rd_{\min}\left.\frac{E_b}{N_0}\right|_{C} - \left.\frac{E_b}{N_0}\right|_{NC} = \log\left(\delta_e N(d_{\min})\sqrt{\frac{\left.\frac{E_b}{N_0}\right|_{NC}}{Rd_{\min}\left.\frac{E_b}{N_0}\right|_{C}}}\right) \tag{1.18}$$

Si $\delta_e N(d_{\min})$ n'est pas trop éloigné de l'unité, cette relation peut être simplifiée sous la forme :

$$Rd_{\min}\left.\frac{E_b}{N_0}\right|_{C} - \left.\frac{E_b}{N_0}\right|_{NC} \approx 0$$

Le *gain asymptotique*, exprimé en dB, fournit l'écart entre $\left.\frac{E_b}{N_0}\right|_{NC}$ et $\left.\frac{E_b}{N_0}\right|_{C}$:

$$G_a = 10\log\left(\frac{\left.\frac{E_b}{N_0}\right|_{NC}}{\left.\frac{E_b}{N_0}\right|_{C}}\right) \approx 10\log\left(Rd_{\min}\right) \tag{1.19}$$

Comme annoncé plus haut, $Rd_{\min}$ apparaît comme un facteur de mérite qui, dans un bilan de liaison et à faible taux d'erreurs, fixe le gain que peut apporter un procédé de codage sur canal gaussien quand le décodeur est à entrée souple. C'est un paramètre majeur pour les concepteurs de systèmes de communications. Pour d'autres types de canal que le canal gaussien (Rayleigh, Rice ...), le gain asymptotique est toujours supérieur à celui qui est approximativement donné par (1.19).

Dans la figure 1.5, le décodage à entrée souple du code de Hamming (8, 4, 4) donne le meilleur résultat, avec un gain asymptotique observé de l'ordre de 2,4 dB, conforme à la relation (1.18) plus précise que (1.19). Le code (7, 4, 3) est légèrement moins performant car le produit $Rd_{\min}$ est de 12/7 au lieu de 2 pour le code (8, 4, 4). En revanche, le décodage à entrée ferme est défavorable au code étendu car il n'offre pas un pouvoir de correction plus grand malgré un taux de redondance plus élevé. Cet exemple est atypique : dans la très grande majorité des cas pratiques, la hiérarchie des codes que l'on peut établir en se basant sur leur performance sur canal gaussien, avec décodage à entrée souple, est respectée pour d'autres types de perturbations.

1.6 Qu'est-ce qu'un bon code ?

La figure 1.6 représente trois comportements possibles pour un code correcteur d'erreurs et son décodeur associé, sur un canal gaussien. Pour être concret, le bloc d'information est supposé être de longueur $k = 1504$ bits (188 octets, longueur typique de de la compression MPEG) et le rendement de codage 1/2.

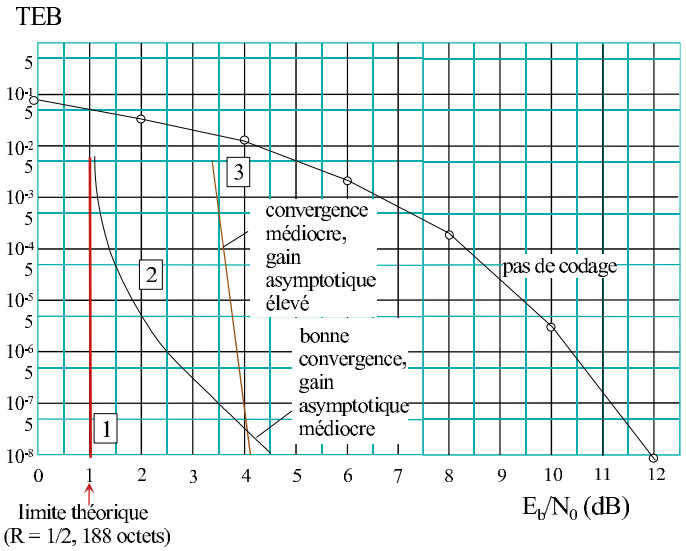


Figure 1.6 – Comportements possibles pour un schéma de codage/décodage sur canal gaussien ($k = 1504$ bits, $R = 1/2$).

La courbe 1 correspond au système idéal. Il existe en effet des limites à la capacité de correction d'un code, quelqu'il soit. Ces limites, dont les premières valeurs ont été établies par Shannon (1947-48) et qui ont été affinées depuis lors pour les situations pratiques, sont données pour être indépassables. Elles dépendent du type de bruit, de la taille de mot de code et du rendement. Le chapitre 3 en fournit les principales valeurs.

La courbe 2 décrit un comportement avec ce que l'on appelle une bonne *convergence* mais aussi avec une DMH médiocre. Bonne convergence signifie que le taux d'erreurs décroît fortement près de la limite théorique (cette région de nette décroissance est appelée *waterfall* en anglais) mais la DMH n'est pas suffisante pour maintenir une pente raide jusqu'à des taux d'erreurs très faibles. Le gain asymptotique, approché par (1.19), est atteint à un taux d'erreur binaire de l'ordre de 10^{-5} dans cet exemple. Au-delà, la courbe reste parallèle à la courbe de taux d'erreurs sans codage : $P_e = \frac{1}{2}\mathrm{erfc}\sqrt{\frac{E_b}{N_0}}$. Le gain asymptotique est ici de l'ordre de 7,5 dB. Ce genre de comportement, que l'on ne rencontrait pas avant les années 1990, est typique de systèmes de codage mettant en œuvre une technique de décodage itérative (turbocodes, LDPC,...), quand la DMH n'est pas très élevée.

La courbe 3 affiche une performance avec une convergence médiocre et un fort gain asymptotique. Un exemple typique en est la concaténation d'un code de Reed-Solomon et d'un code convolutif. Alors que la DMH peut être très grande (autour de 100 par exemple), le décodeur ne sait en tirer parti que relativement loin de la limite théorique. Ce n'est donc pas la qualité du code qui

est en question mais le décodeur qui ne peut pas exploiter toute l'information disponible en réception.

La recherche du couple codeur/décodeur idéal, depuis les travaux de Shannon, a toujours dû faire face à ce dilemme : bonne convergence *versus* DMH élevée. D'excellents codes algébriques tels que les codes BCH (voir paragraphe 4.1.7) ou de Reed-Solomon (voir paragraphe 4.2.1) furent assez rapidement élaborés dans l'histoire du codage correcteur (voir chapitre 4). Les DMH sont élevées (et même parfois optimales) mais il n'est pas toujours facile de mettre en œuvre un décodage à entrée souple. Par ailleurs, les codes algébriques sont généralement « taillés » pour une longueur de mot de code et un rendement de codage bien spécifiques, ce qui en limite le champ des applications. Malgré cela, les codes algébriques rendent des services essentiels dans les applications qui requièrent des taux d'erreurs très faibles, notamment les mémoires de masse et/ou lorsque l'information pondérée n'est pas disponible.

Ce n'est que récemment, avec l'introduction du décodage probabiliste itératif (turbo-décodage), que l'on a pu obtenir une correction d'erreurs efficace près de la limite théorique. Et c'est encore plus récemment que des DMH suffisantes ont pu être obtenues pour éviter un changement de pente défavorable dans la courbe de performance.

Il n'est pas facile de répondre simplement à la question qui fait le titre de cette section. La performance est, bien sûr, le premier critère : pour un taux d'erreurs donné, compté soit en TEB soit en TEP et pour un rendement de codage fixé, un bon code est d'abord celui dont le décodeur offre un bon pouvoir de correction près de la limite théorique correspondante. Une condition préalable en est évidemment l'existence d'un algorithme de décodage (les codes aléatoires n'ont pas de décodeur, par exemple) et que cet algorithme ne soit pas exagérément complexe, dans ses réalisations logicielles et/ou matérielles. En outre, l'utilisation d'entrées souples est un impératif qui peut ne pas être simple à satisfaire.

D'autres critères comme la vitesse de décodage (qui fixe le débit de l'information décodée), la latence (le retard apporté par le décodage) ou la flexibilité (l'aptitude du code à pouvoir être défini pour des longueurs de mots et des rendements de codage variés) sont également à prendre en compte dans le contexte de l'application visée.

Enfin, des facteurs non techniques peuvent également être très importants. La maturité technologique (existe-t-il déjà des applications, des standards ?), les coûts des composants, les éventuels droits de propriété intellectuelle, les préférences stratégiques ou les habitudes sont des éléments qui peuvent peser dans le choix d'une solution de codage.

1.7 Les familles de codes

Jusqu'à ces dernières années, les codes étaient traditionnellement rangés en deux familles considérées comme bien distinctes par leurs principes et leurs

applications : les *codes algébriques* (encore appelés *codes en bloc*) et les *codes convolutifs*. Cette distinction était principalement fondée sur trois observations :

- les codes algébriques sont appropriés à la protection de blocs de données indépendants les uns des autres, les codes convolutifs conviennent à la protection de données en flot continu,
- les rendements de codage des codes algébriques sont plutôt proches de l'unité, alors que les codes convolutifs ont des rendements plus faibles,
- le décodage des codes en bloc est plutôt du type à entrée ferme, celui des codes convolutifs presque toujours à entrée souple.

Aujourd'hui, ces distinctions tendent à s'estomper. Les codes convolutifs peuvent aisément être adaptés pour coder des blocs et la plupart des décodeurs de codes algébriques acceptent des entrées souples. Par l'intermédiaire de la concaténation (chapitre 6), les rendements des codes algébriques peuvent être abaissés à des valeurs comparables à celles des codes convolutifs. Une différence demeure cependant entre les deux sous-familles : le nombre d'états logiques possibles des codeurs algébriques est généralement très élevé, ce qui en empêche le décodage par des méthodes d'état exhaustives. Les algorithmes de décodage s'appuient sur des techniques propres à chaque code. Les codeurs convolutifs ont un nombre restreint d'états, de 2 à 256 en pratique et leur décodage utilise une représentation d'états complète, dite en treillis (chapitre 5). Pour cette raison, cet ouvrage a conservé la structuration classique qui distingue, provisoirement, codes algébriques et codes convolutifs.

Le codage moderne fait appel à des structures concaténées ou composites, qui utilisent plusieurs codeurs élémentaires et dont le décodage s'effectue par passages répétés dans les décodeurs associés et par échange d'informations de nature probabiliste. Les turbocodes (1993) ont ouvert la voie à ce type de traitement itératif et, depuis, de nombreuses structures composites qui s'appuient sur le décodage itératif, ont été imaginées (ou retrouvées). Trois d'entre elles sont approfondies dans cet ouvrage : les turbocodes, dans leur version originale et dans leur évolution récente, les codes produits décodés itérativement et les LDPC (*Low Density Parity Check*). Tous ces codes ont été adoptés dans des normes internationales et la compréhension de leurs procédés de codage et de leurs algorithmes de décodage est une base largement suffisante pour aborder tout autre principe de codage distribué et de décodage itératif associé.

Chapitre 2

Communications numériques

2.1 Modulations Numériques

2.1.1 Introduction

La fonction de modulation a son origine dans les communications radioélectriques. Un émetteur ne peut rayonner une onde électromagnétique que dans une portion limitée et généralement étroite du spectre, qui peut grossièrement être décrite comme une « fenêtre » fréquentielle de largeur Δf centrée sur une fréquence f_0, avec $\Delta f \ll f_0$. Les messages à transmettre, qui peuvent être de nature analogique (par exemple la parole) ou numérique (par exemple le télégraphe Morse), sont représentés par des signaux n'occupant que le bas du spectre de fréquences. Le spectre du signal issu d'un microphone dans le cas de la parole ne s'étend pas au-delà de quelques kilohertz. Il en est de même du signal qui représente par une tension maintenue brièvement (Tit) ou plus longtemps (Tat) les deux éléments du « code » Morse, car la vitesse de manipulation de quelques dizaines de signes par seconde est très petite par rapport à la fréquence f_0 qui se mesure en centaines de kilohertz ou en mégahertz. Une autre utilité de la modulation est le multiplexage fréquentiel qui permet plusieurs communications simultanées sur un même support à large bande (câble ou fibre optique) aisément séparées du fait qu'elles occupent chacune une bande spécifique, disjointe de celle des autres.

Une onde sinusoïdale de fréquence f_0 peut être représentée par la fonction :

$$s(t) = a\cos(2\pi f_0 t + \varphi) \tag{2.1}$$

où t note le temps et où f_0 est constante. La modulation consiste à faire dépendre du signal à transmettre l'un et/ou l'autre des paramètres a, l'amplitude, et φ, la phase. Le signal modulé $s(t)$ possède alors, comme on le souhaite, un spectre étroit centré sur f_0.

Le signal à transmettre sera dit dans la suite *signal modulant*. La modulation consiste à faire varier un des paramètres a et φ en fonction du signal modulant si

celui-ci est analogique. Dans le cas d'un signal numérique, le signal modulant est une suite d'éléments d'un ensemble fini, ou symboles, appliqués au modulateur à des instants discrets qui seront dits *instants significatifs*. Cette suite sera dite *message numérique* et on supposera que les symboles sont des données binaires appliquées périodiquement à l'entrée du modulateur, toutes les T_b secondes, donc avec un débit binaire $D = 1/T_b$ bits par seconde. Les données binaires de cette suite seront supposées indépendantes et identiquement distribuées (iid). Un message numérique donné, par exemple binaire, peut être remplacé par sa « m-ième extension » obtenue en groupant les symboles initiaux par paquets de m. Alors les symboles sont des nombres à m chiffres binaires (ou m-uples), dont le nombre total est $M = 2^m$, appliqués au modulateur à des instants significatifs de période mT_b. Si la m-ième extension est entièrement équivalente au message (dont elle n'est qu'une description différente), le signal modulé par le message d'origine et celui qui l'est par sa m-ième extension n'ont pas les mêmes propriétés, notamment en ce qui concerne sa largeur de bande, d'autant plus étroite que m est grand. Le choix du paramètre entier m permet donc de faire varier les caractéristiques du message modulé.

Une modulation numérique implique de définir un ensemble discret de valeurs pouvant être prises par a et φ à chacun des instants significatifs, en nombre égal à celui des symboles possibles, soit $M = 2^m$ pour la m-ième extension d'un message binaire, et en correspondance bijective avec eux. Cet ensemble est appelé *constellation* pour des raisons qui vont être dites un peu plus loin. Alors un message quelconque sera représenté par un élément (point) de la constellation. On remarquera que les éléments du message numérique à transmettre sont de nature abstraite (par exemple un chiffre binaire, 0 ou 1, ou le m-uple qui représente les symboles de la m-ième extension d'un message binaire) tandis que les paramètres a et φ représentent des grandeurs physiques qui ne peuvent être ajustées ou connues qu'avec une précision finie. Il faut supposer que l'écart entre ces paramètres et leurs valeurs nominales dans la constellation reste négligeable par rapport au bruit qui affecte la transmission.

Considérons le signal complexe :

$$\sigma(t) = a \exp[j(2\pi f_0 t + \varphi)] \tag{2.2}$$

dont $s(t)$ est la partie réelle, où j est solution de l'équation $x^2 + 1 = 0$. On peut représenter $\sigma(t)$ sous la forme du produit

$$\sigma(t) = \alpha \exp(2\pi j f_0 t), \tag{2.3}$$

où seul le premier facteur

$$\alpha = a \exp(j\varphi) \tag{2.4}$$

dépend des paramètres a et φ qui représentent les données à transmettre. Les valeurs prises par ce premier facteur pour toutes les valeurs possibles des paramètres peuvent être représentées par des points dans le plan complexe. L'ensemble de ces points est alors appelé *constellation* et le plan complexe est dit

plan de Fresnel. Le signal modulé (2.1) est la partie réelle du signal complexe $\sigma(t)$ défini par (2.3).

Si la correspondance établie entre le signal modulant et les paramètres variables est instantanée, la modulation est dite *sans mémoire*. Il peut être utile que cette correspondance soit établie entre les paramètres variables et une fonction des valeurs prises antérieurement par le signal modulant. Par exemple, si celui-ci est analogique, un procédé classique de modulation (dit de fréquence) consiste à faire varier φ proportionnellement à l'intégrale du signal modulant par rapport au temps. Dans le cas d'une modulation numérique, de même, le point de la constellation peut être choisi pour représenter le symbole présent à l'instant considéré, et la modulation est alors dite *sans mémoire*, ou bien un symbole obtenu en le combinant avec d'autres symboles antérieurs. Une modulation peut donc être analogique ou numérique, avec ou sans mémoire. Nous nous restreindrons dans toute la suite aux modulations numériques qui différeront les unes des autres par la forme et le nombre de points de leur constellation, ainsi qu'éventuellement par un effet de mémoire. Quand celui-ci est obtenu en combinant le symbole appliqué avec des symboles antérieurs, il peut être interprété comme une transformation préliminaire du message numérique. Par ailleurs, il est souvent nécessaire d'assurer la continuité de la phase pour améliorer la forme du spectre des signaux modulés, ce qui implique un effet de mémoire.

Le choix d'un système de modulation dépend de nombreux facteurs. Les signaux modulés seront émis sur un canal imparfait, perturbé par l'addition de signaux parasites nommés collectivement *bruit* et souvent, en radioélectricité, affectés de variations de l'amplitude du signal reçu dues par exemple à un changement rapide des conditions de propagation, phénomène nommé évanouissement. Malgré ces défauts du canal, on souhaite recevoir les messages modulants avec une faible probabilité d'erreur, ce qui implique que les signaux qui leur sont associés soient aussi différents que possible. D'autre part, qu'il s'agisse de radioélectricité ou de multiplexage, le spectre radioélectrique est commun à plusieurs utilisateurs dont chacun perturbe les autres. On souhaite donc concentrer la puissance émise dans un intervalle de fréquence aussi bien délimité que possible, ce qui implique de choisir les paramètres de la modulation pour doter le spectre de la forme la mieux appropriée. Le spectre d'un signal de la forme générale (2.3) se compose d'un *lobe principal* centré sur f_0 qui concentre la plus grande partie de la puissance émise et de *queues* ou *lobes secondaires* où la densité spectrale décroît plus ou moins vite par rapport à la fréquence centrale f_0. Quel que soit le système de modulation, la largeur du lobe principal est proportionnelle à la *rapidité de modulation* $R = D/m = 1/mT_b$, exprimée en bauds. La décroissance de la densité spectrale suffisamment loin de la fréquence centrale ne dépend que des discontinuités du signal modulant et de ses dérivées. Elle varie en $1/(f - f_0)^{2(d+1)}$ où d est l'ordre le plus petit d'une dérivée du signal discontinue ($d = 0$ si le signal modulé lui-même est discontinu). On remarquera qu'il n'est possible d'augmenter la valeur de d qu'en

introduisant un retard croissant entre l'instant où un symbole modulant est appliqué au modulateur et l'instant caractéristique lui correspondant.

Les principaux paramètres associés à la modulation sont donc la taille et la forme de la constellation employée (dont dépend, sur un canal donné, la probabilité d'erreur), la largeur du lobe principal du spectre du signal modulé et la décroissance de sa densité spectrale loin de la fréquence centrale. Ils sont dans une large mesure en conflit : par exemple, on ne pourra réduire la largeur du lobe central qu'en augmentant la taille de la constellation, au détriment de la probabilité d'erreur pour une même puissance. Le choix d'un système de modulation ne peut donc que résulter d'un compromis adapté à une application déterminée. Outre les paramètres indiqués, la complexité de la réalisation doit être prise en compte. Par exemple, une mise en forme améliorant la décroissance des lobes secondaires du spectre par augmentation de l'ordre d de la première dérivée discontinue, ou bien l'augmentation de la taille de la constellation pour diminuer le lobe central du spectre, conduisent à accroître la complexité du modulateur.

2.1.2 Modulations Linéaires Sans Mémoire

Modulation par déplacement d'amplitude à M états : MDA-M

Observons d'abord un cas particulier de modulation d'amplitude : la modulation par tout ou rien (en anglais *OOK*, pour *On Off Keying*), pour laquelle a dans l'expression (2.1) prend la valeur 0 ou A. L'un des états binaires correspond donc à une extinction de la porteuse. La présence et l'absence de la porteuse peuvent être reconnues indépendamment de la connaissance de la phase φ, en mesurant l'énergie du signal reçu pendant un intervalle de temps petit à l'échelle de la période $1/f_0$ de l'onde porteuse, ce que l'on appelle *détection incohérente*. La figure 2.1 représente une tranche temporelle d'un signal modulé en amplitude par tout ou rien.

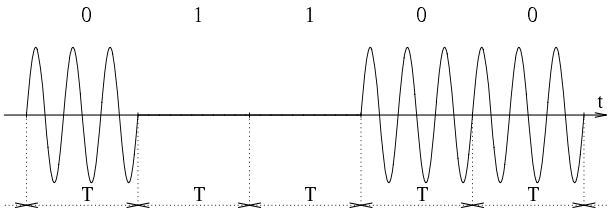


Figure 2.1 – Modulation d'amplitude par tout ou rien (*OOK* : *On Off Keying*).

Dans le cas général de la modulation d'amplitude à M états (ou *Amplitude Shift Keying – ASK* en anglais), l'amplitude de la porteuse est la grandeur modulée $a_j = A_j h(t)$ pour $j = 1, 2, ..., M$ où A_j prend une valeur parmi $M = 2^m$ valeurs en fonction du groupe de données présenté à l'entrée du modulateur et $h(t)$ est une porte temporelle d'amplitude unité et de durée T. Le modulateur

délivre donc des signaux de la forme :

$$s_j(t) = A_j h(t) \cos(2\pi f_0 t + \varphi_0) \tag{2.5}$$

$$\text{avec } A_j = (2j - 1 - M)A \quad j = 1, 2, \cdots, M \text{ où } A \text{ est constante} \tag{2.6}$$

et :

$$\begin{array}{rcll} h(t) & = & 1 & \text{pour} \quad t \in [0, T[\\ & = & 0 & \text{ailleurs} \end{array} \tag{2.7}$$

L'amplitude A_j du signal modulé est constante pendant une durée T puis change de valeur, le signal modulé transmet donc $\log_2(M)$ données binaires toutes les T secondes. On peut noter que la moitié des amplitudes nominales A_j sont négatives. L'identité $a\cos\varphi = -a\cos(\varphi + \pi)$ implique alors une démodulation *cohérente* où la phase φ est connue. La phase n'étant souvent connue qu'à un multiple de π près, un message binaire où les symboles sont ± 1 ne sera démodulé qu'au signe près.

Pour un signal MDA-M, les différents états du signal modulé sont situés sur une droite et sa constellation est donc à une dimension dans le plan de Fresnel. Il existe de nombreuses façons de faire l'association entre la valeur de l'amplitude du signal modulé et la réalisation particulière d'un groupe de données de $m = \log_2(M)$ données. En général, à deux valeurs adjacentes prises par l'amplitude on associe deux groupes de données qui diffèrent par une seule valeur binaire. Cette association particulière est appelée *codage de Gray*. Elle permet de minimiser les erreurs commises par le récepteur. En effet, lorsque le récepteur sélectionne une amplitude adjacente à l'amplitude émise à cause du bruit, ce qui correspond à la situation la plus fréquente, on commet une seule erreur pour $m = \log_2(M)$ données transmises. Nous avons représenté sur la figure 2.2 deux exemples de constellations de signal modulé en amplitude avec codage de Gray.

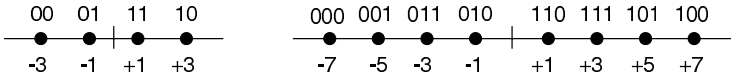


Figure 2.2 – Exemple de constellations de signal MDA-4 et MDA-8 avec codage de Gray

L'énergie moyenne E_s utilisée pour transmettre un symbole M-aire est égale à :

$$E_s = \int_0^T \mathrm{E}\left\{A_j^2\right\} \cos^2(2\pi f_0 t + \varphi_0) dt$$

où $\mathrm{E}\left\{A_j^2\right\}$, espérance de A_j^2 a pour expression $A^2(M^2 - 1)/3$
En faisant l'hypothèse que $f_0 >> \frac{1}{T}$, la relation précédente donne l'énergie moyenne E_s :

$$E_s = \frac{A^2 T}{2} \frac{(M^2 - 1)}{3} \tag{2.8}$$

L'énergie moyenne E_b utilisée pour transmettre une donnée binaire est :

$$E_b = \frac{E_s}{\log_2(M)} \tag{2.9}$$

Pour une transmission à flot continu de données, le signal modulé en amplitude peut s'écrire sous la forme :

$$S(t) = A \sum_i a_i h(t - iT) \cos(2\pi f_0 t + \varphi_0) \tag{2.10}$$

où les $\{a_i\}$ sont une suite de symboles M-aires, appelés symboles de modulation, qui prennent les valeurs $(2j - 1 - M)$, $j = 1, 2, \cdots, M$. Dans l'expression du signal modulé, i est l'indice temporel.

Le signal $S(t)$ peut encore se mettre sous la forme :

$$S(t) = \Re_e \left\{ s_e(t) \exp\left(j(2\pi f_0 t + \varphi_0)\right) \right\} \tag{2.11}$$

où $s_e(t)$ est l'enveloppe complexe du signal $S(t)$ avec :

$$s_e(t) = A \sum_i a_i h(t - iT) \tag{2.12}$$

En tenant compte du fait que les données d_i délivrées par la source d'information sont iid, les symboles de modulation a_i sont indépendants, à moyenne nulle et de variance égale à $(M^2 - 1)/3$.

Il peut être montré que la densité spectrale de puissance (dsp) du signal $S(t)$ est égale à :

$$\gamma_S(f) = \frac{1}{4}\gamma_{s_e}(f - f_0) + \frac{1}{4}\gamma_{s_e}(f + f_0) \tag{2.13}$$

avec :

$$\gamma_{s_e}(f) = \frac{M^2 - 1}{3} A^2 T \left(\frac{\sin \pi f T}{\pi f T} \right)^2 \tag{2.14}$$

La dsp de $s_e(t)$ exprimée en dB est représentée dans la figure 2.3 en fonction de la fréquence normalisée fT, pour $M = 4$ et $A^2T = 1$.

La dsp de $S(t)$ est centrée sur la fréquence porteuse f_0 et son enveloppe décroît en f^2. Elle est constituée d'un lobe principal de largeur $2/T$ et de lobes secondaires qui s'annulent périodiquement en $f_0 \pm k/T$.

Remarque
La bande occupée est, en toute rigueur infinie. Pratiquement, on peut décider de ne transmettre qu'un pourcentage de la puissance du signal $S(t)$ et dans ce cas, la bande occupée est finie. Si, par exemple, on décide de transmettre 99% de la puissance du signal modulé, ce qui n'induit qu'une faible distorsion du signal $S(t)$, alors la bande occupée est d'environ $8/T$ où $1/T$ est la rapidité de modulation. On verra dans la section 2.3 qu'il est possible de réduire fortement cette bande sans dégrader les performances de la modulation. Cette remarque est valable pour toutes les modulations linéaires.

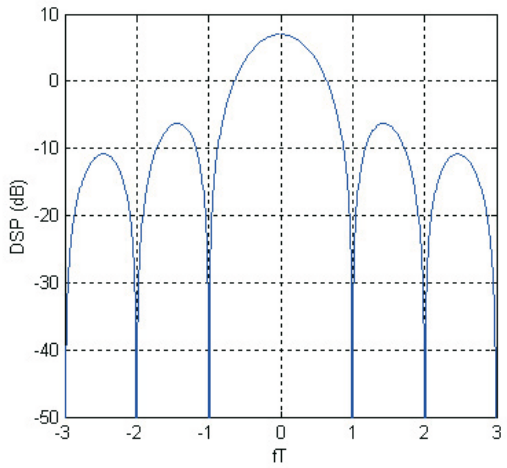


Figure 2.3 – Densité spectrale de puissance (dsp) de l'enveloppe complexe d'un signal MDA-4, avec $A^2T = 1$.

Modulation par déplacement de phase à M états : MDP-M

Pour cette modulation appelée aussi *Phase Shift Keying – M-PSK* en anglais, c'est la phase de la porteuse qui est la grandeur modulée. Le modulateur délivre des signaux de la forme :

$$s_j(t) = Ah(t)\cos(2\pi f_0 t + \varphi_0 + \phi_j) \tag{2.15}$$

où f_0 est la fréquence de la porteuse, φ_0 sa phase et $h(t)$ une porte d'amplitude unité et de durée T. La phase modulée ϕ_j prend une valeur parmi $M = 2^m$ avec :

$$\phi_j = (2j+1)\frac{\pi}{M} + \theta_0 \quad 0 \leq j \leq (M-1) \tag{2.16}$$

Les différents états de la phase sont équirépartis sur un cercle de rayon A. La phase θ_0 est fixée à $-\pi/2$ pour une modulation MDP-2 et à 0 pour une modulation MDP-M avec $M > 2$ états.

Le signal modulé peut encore s'écrire sous la forme :

$$s_j(t) = Ah(t)\left[\cos\phi_j \cos(2\pi f_0 t + \varphi_0) - \sin\phi_j \sin(2\pi f_0 t + \varphi_0)\right] \tag{2.17}$$

Sous cette forme, le signal MDP-M peut s'exprimer comme la somme de deux porteuses, $\cos(2\pi f_0 t + \varphi_0)$ et $-\sin(2\pi f_0 t + \varphi_0)$ en quadrature, modulées en amplitude par $\cos\phi_j$ et $\sin\phi_j$ avec $\cos^2\phi_j + \sin^2\phi_j = 1$. On peut vérifier que lorsque M est un multiple de 4, les valeurs possibles de l'amplitude des deux porteuses sont identiques.

Dans la figure 2.4 nous avons représenté deux constellations d'un signal modulé en phase avec codage de Gray. Les constellations sont à deux dimensions

et les différents états du signal modulé sont sur un cercle de rayon A. On dit que la constellation est circulaire.

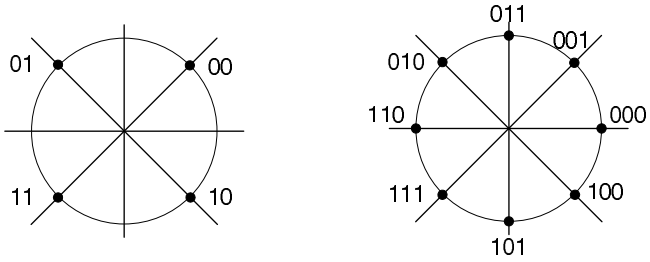


Figure 2.4 – Exemples de constellations de signal modulé en phase avec codage de Gray.

L'énergie E_s pour transmettre un état de phase c'est-à-dire un groupe de $\log_2(M)$ données binaires, est égale à :

$$E_s = \int_0^T A^2 \cos^2(2\pi f_0 t + \varphi_0 + \phi_j)dt = \frac{A^2 T}{2} \quad \text{si } f_0 >> 1/T \tag{2.18}$$

L'énergie E_s est toujours la même quel que soit l'état de phase émis. L'énergie utilisée pour transmettre une donnée binaire est $E_b = E_s/\log_2(M)$.

Pour une transmission à flot continu de données, le signal modulé peut s'écrire sous la forme :

$$\begin{aligned} S(t) = A \Bigg[\sum_i a_i h(t - iT) \cos(2\pi f_0 t + \varphi_0) \\ - \sum_i b_i h(t - iT) \sin(2\pi f_0 t + \varphi_0) \Bigg] \end{aligned} \tag{2.19}$$

où les symboles de modulation a_i et b_i prennent leurs valeurs dans les ensembles suivants :

$$\begin{array}{l} a_i \in \left\{ \cos\left[(2j+1)\frac{\pi}{M} + \theta_0\right] \right\} \quad 0 \le j \le (M-1) \\ b_i \in \left\{ \sin\left[(2j+1)\frac{\pi}{M} + \theta_0\right] \right\} \quad 0 \le j \le (M-1) \end{array} \tag{2.20}$$

Le signal $S(t)$ peut encore se mettre sous la forme donnée par (2.11) avec :

$$s_e(t) = A \sum_i c_i h(t - iT), \quad c_i = a_i + j b_i \tag{2.21}$$

En tenant compte du fait que les données d_i délivrées par la source d'information sont iid, les symboles de modulation c_i sont indépendants, à moyenne nulle et de variance unité.

La dsp du signal $S(t)$ est encore égale à :

$$\gamma_S(f) = \frac{1}{4}\gamma_{s_e}(f - f_0) + \frac{1}{4}\gamma_{s_e}(f + f_0)$$

avec cette fois-ci :

$$\gamma_{s_e}(f) = A^2 T \left(\frac{\sin \pi f T}{\pi f T} \right)^2 \tag{2.22}$$

La dsp a une allure similaire à celle de la figure 2.3.

Modulation d'amplitude sur deux porteuses en quadrature : MAQ-M

Pour cette modulation appelée aussi *Quadrature Amplitude Modulation – M-QAM* en anglais, ce sont deux porteuses en quadrature $\cos(2\pi f_0 t + \varphi_0)$ et $-\sin(2\pi f_0 t + \varphi_0)$ qui sont modulées en amplitude. Le modulateur délivre des signaux de la forme :

$$s_j(t) = A_j^c h(t) \cos(2\pi f_0 t + \varphi_0) - A_j^s h(t) \sin(2\pi f_0 t + \varphi_0) \tag{2.23}$$

où f_0 est la fréquence de la porteuse, φ_0 sa phase et $h(t)$ une porte d'amplitude unité et de durée T.

Deux situations peuvent se présenter selon que la longueur m des groupes de données à l'entrée du modulateur est paire ou non. Si m est pair, alors $M = 2^m$ est un carré parfait (4, 16, 64, 256, ...) ; dans le cas contraire, M est simplement une puissance de deux (8, 32, 128, ...).

Lorsque m est pair, le groupe de données peut être séparé en deux sous-groupes de longueur $m/2$, chacun étant associé respectivement aux amplitudes A_j^c et A_j^s qui prennent leurs valeurs dans l'ensemble $(2j - 1 - \sqrt{M})A, \quad j = 1, 2, \cdots, \sqrt{M}$. Sur la figure 2.5 sont représentées les constellations des modulations MAQ-16 et MAQ-64. Ces constellations sont dites carrées.

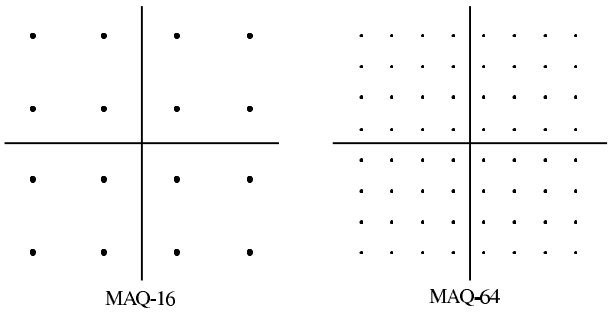


Figure 2.5 – Constellations de deux modulations de type MAQ.

Lorsque m est impair, le signal modulé MAQ-M ne peut plus être obtenu comme une combinaison de deux porteuses en quadrature modulées en amplitude. Toutefois, on peut construire le signal MAQ-M à partir d'un signal MAQ-N modulé classiquement sur deux porteuses en quadrature, où N est un carré immédiatement supérieur à M en interdisant $(N-M)$ états. Par exemple, la modulation MAQ-32 peut être obtenue à partir d'une modulation MAQ-36 où A_j^c et A_j^s prennent les valeurs $(\pm A, \pm 3A, \pm 5A)$ en interdisant les quatre

états d'amplitude $(\pm 5A, \pm 5A)$ pour les couples (A_j^c et A_j^s). La constellation de la modulation MAQ-32 est représentée dans la figure 2.6.

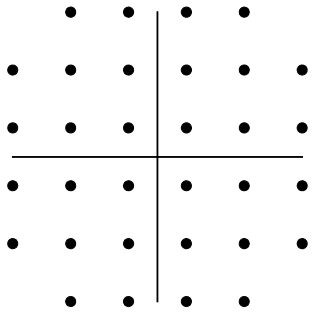


Figure 2.6 – Constellation de la modulation MAQ-32.

Le signal MAQ-M peut encore s'écrire sous la forme :

$$s_j(t) = V_j h(t) \cos(2\pi f_0 t + \varphi_0 + \phi_j) \tag{2.24}$$

avec :

$$V_j = \sqrt{(A_j^c)^2 + (A_j^s)^2} \qquad \phi_j = \arctan \frac{A_j^s}{A_j^c}$$

Sous cette forme, la modulation MAQ-M peut être considérée comme une modulation combinée de phase et d'amplitude. En supposant que la phase prenne $M_1 = 2^{m_1}$ états et l'amplitude $M_2 = 2^{m_2}$ états, le signal modulé transmet $\log_2(M_1 M_2) = m_1 + m_2$ données toutes les T secondes. La figure 2.7 représente la constellation d'une modulation combinée de phase et d'amplitude pour $M = 16$ ($M_1 = 4$, $M_2 = 4$).

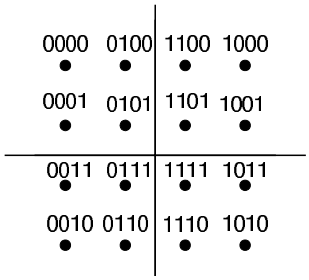


Figure 2.7 – Constellation d'une modulation combinée de phase et d'amplitude pour $M = 16$.

L'énergie moyenne E_s pour transmettre le couple (A_j^c, A_j^s) c'est-à-dire un groupe de $\log_2(M)$ données binaires, est égale à :

$$E_s = \int_0^T \mathrm{E}\left\{V_j^2\right\} \cos^2(2\pi f_0 t + \varphi_0 + \phi_j) dt \tag{2.25}$$

Pour un groupe de données de longueur m paire, $\mathrm{E}\left\{V_j^2\right\} = 2A^2(M-1)/3$ et ainsi, pour $f_0 >> 1/T$, l'énergie moyenne E_s est égale à :

$$E_s = A^2 T \frac{M-1}{3} \tag{2.26}$$

L'énergie moyenne utilisée pour transmettre une donnée binaire est $E_b = E_s/\log_2(M)$.

Pour une transmission à flot continu de données, le signal modulé peut s'écrire sous la forme :

$$\begin{aligned} S(t) = A \Bigg[& \sum_i a_i h(t-iT) \cos(2\pi f_0 t + \varphi_0) \\ & - \sum_i b_i h(t-iT) \sin(2\pi f_0 t + \varphi_0) \Bigg] \end{aligned} \tag{2.27}$$

où les symboles de modulation a_i et b_i prennent les valeurs $(2j - 1 - \sqrt{M})$, $j = 1, 2, \cdots, \sqrt{M}$ pour $M = 2^m$ avec m pair. Le signal $S(t)$ peut être exprimé par les relations (2.11) et (2.21) :

$$S(t) = \Re_e \left\{ s_e(t) \exp j(2\pi f_0 t + \varphi_0) \right\}$$

avec $s_e(t) = A \sum_i c_i h(t - iT), \quad c_i = a_i + j b_i$

Les données binaires d_i délivrées par la source d'information étant iid, les symboles de modulation c_i sont indépendants, à moyenne nulle et de variance égale à $2(M-1)/3$.

La dsp du signal $S(t)$ est encore donnée par (2.13) avec :

$$\gamma_{s_e}(f) = \frac{2(M-1)}{3} A^2 T \left(\frac{\sin \pi f T}{\pi f T} \right)^2 \tag{2.28}$$

L'occupation spectrale d'un signal modulé MAQ-M est donc la même que celle des signaux MDA-M et MDP-M.

2.1.3 Modulations de fréquence sans mémoire à M états : MDF-M

Pour cette modulation appelée aussi *Frequency Shift Keying – M-FSK* en anglais, c'est la fréquence qui est la grandeur modulée. Le modulateur délivre des signaux de la forme :

$$s_j(t) = A h(t) \cos(2\pi(f_0 + f_j)t + \varphi_j) \tag{2.29}$$

où $f_j = j\Delta f$, $j = 1, 2, \cdots, M$ et $h(t)$ est une porte d'amplitude unité et de durée T. Les φ_j sont des phases aléatoires indépendantes de réalisation constante sur l'intervalle $[0, T[$. Les signaux $s_j(t)$ peuvent donc être délivrés par des oscillateurs indépendants puisqu'il n'existe aucune relation entre les phases φ_j.
Calculons le coefficient de corrélation entre deux signaux modulés prenant des états de fréquence différents.

$$\rho_{j,n} = \int_0^T A^2 \cos(2\pi(f_0 + j\Delta f)t + \varphi_j) \cos(2\pi(f_0 + n\Delta f)t + \varphi_n)dt \qquad (2.30)$$

Après intégration et en supposant $f_0 >> 1/T$, on obtient :

$$\rho_{j,n} = \frac{A^2T}{2}\left(\frac{\sin(2\pi(j-n)\Delta fT + \varphi_j - \varphi_n)}{2\pi(j-n)\Delta fT} - \frac{\sin(\varphi_j - \varphi_n)}{2\pi(j-n)\Delta fT}\right) \qquad (2.31)$$

En choisissant $\Delta f = 1/T$, $\rho_{j,n} = 0\ \forall j \neq n$ et les M signaux modulés sont orthogonaux. Le choix de signaux orthogonaux est généralement retenu car il permet de séparer aisément les signaux de fréquences différentes en réception. Aux instants iT où le signal MDF-M change de fréquence, le signal modulé présente une discontinuité car les phases φ_j sont indépendantes. On parle alors de modulation de fréquence à phase discontinue. La figure 2.8 donne un exemple de signal MDF-2.

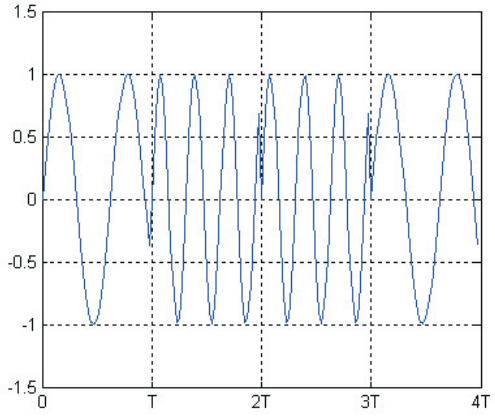


Figure 2.8 – Signal modulé en fréquence avec phase discontinue.

L'énergie E_s utilisée pour transmettre un groupe de données s'obtient comme :

$$E_s = \int_0^T A^2 \cos^2(2\pi(f_0 + \frac{j}{T})t + \varphi_j)dt = \frac{A^2T}{2} \quad \text{si} \quad f_0 >> 1/T \qquad (2.32)$$

et l'énergie E_b utilisée pour transmettre une donnée binaire s'obtient comme :

$$E_b = \frac{E_s}{\log_2(M)} \tag{2.33}$$

Pour une transmission à flot continue de données, le signal modulé peut s'écrire sous la forme :

$$S(t) = A \sum_i h(t - iT) \cos(2\pi(f_0 + \frac{a_i}{T})t + \varphi_i) \tag{2.34}$$

où le symbole de modulation a_i est égal à 1, 2, $\cdots$, M, $M = 2^m$. Toutes les T secondes, le signal modulé $S(t)$ transmet un groupe de $\log_2(M)$ données binaires.

Pour une modulation MDF-2, la densité spectrale de puissance du signal $S(f)$ est égale à :

$$\gamma_S(f) = \frac{1}{4}\left(\gamma(f - f_1) + \gamma(f + f_1) + \gamma(f - f_2) + \gamma(f + f_2)\right) \tag{2.35}$$

où $f_1 = f_0 + 1/T$ et $f_2 = f_0 + 2/T$ et :

$$\gamma(f) = \frac{A^2 T}{4}\left(\frac{\sin \pi f T}{\pi f T}\right)^2 + \frac{A^2}{4}\delta(f) \tag{2.36}$$

où $\delta(f)$ représente la distribution de Dirac. La dsp d'un signal MDF-2 possède une partie continue et une partie discrète. En se limitant aux deux lobes principaux de cette densité spectrale de puissance, la bande de fréquences occupée par un signal MDF-2 est de $3/T$, soit trois fois la rapidité de modulation. Rappelons qu'à même rapidité de modulation, un signal MDP-M ou MAQ-M occupe une bande de fréquence de seulement $2/T$. La partie discrète correspond à deux raies spectrales situées en f_1 et f_2.

2.1.4 Modulations avec mémoire par déplacement de fréquence à phase continue : MDF-M PC

Pour les modulations de fréquence à phase continue, appelées *Continuous Phase Frequency Shift Keying – CP M-FSK* en anglais, le signal modulé ne présente pas de discontinuités aux instants de changement de fréquence. La figure 2.9 représente une tranche temporelle d'un signal MDF-M PC pour $M = 2$. Le signal MDF-M PC a pour expression :

$$S(t) = A \cos(2\pi f_0 t + \varphi_0 + \phi(t)) \tag{2.37}$$

où f_0 est la fréquence de la porteuse et φ_0 sa phase.

L'excursion de la fréquence instantanée ($f_i = d\phi/dt$) est :

$$f(t) = \frac{1}{2\pi}\frac{d\phi}{dt} = h \sum_i a_i g(t - iT) \tag{2.38}$$

où h est appelé indice de modulation et les symboles M-aires a_i prennent leurs valeurs dans l'alphabet $\{\pm 1, \pm 3, \cdots, \pm(2p+1), \cdots, \pm(M-1)\}$; $M = 2^m$. La fonction $g(t)$ est causale et à support limité :

$$\begin{array}{rcll} g(t) & \neq & 0 & t \in [0, LT[\,, \quad L \text{ entier} \\ & = & 0 & \text{ailleurs} \end{array} \tag{2.39}$$

En posant $q(t) = \int_0^t g(\tau)d\tau$ et en imposant pour normaliser :

$$q(t) = \frac{1}{2} \quad \text{si } t \geq LT$$

on peut écrire la phase $\phi(t)$ sur l'intervalle $[iT, (i+1)T[$:

$$\phi(t) = 2\pi h \sum_{n=i-L+1}^{i} a_n q(t-nT) + \pi h \sum_{n=-\infty}^{i-L} a_n \tag{2.40}$$

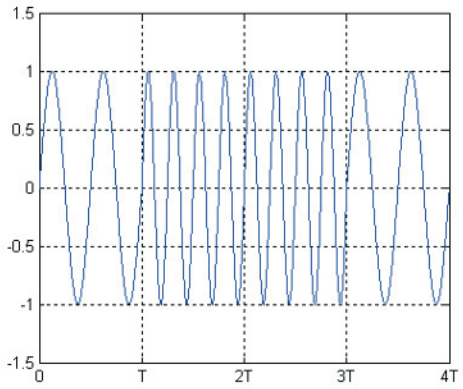


Figure 2.9 – Tranche temporelle d'un signal MDF-2 PC.

Lorsque $L = 1$, les modulations de fréquence à phase continue sont dites à réponse totale (*full response* en anglais) alors que pour $L > 1$, elles sont dites à réponse partielle (*partial response*).

Pour illustrer les modulations de fréquence à phase continue, nous allons considérer trois exemples, la modulation *MSK* (*Minimum Shift Keying*), la modulation *L-RC* (*Raised Cosine pulse*) de durée L et la modulation *GMSK* (*Gaussian Minimum Shift Keying*).

Modulation de fréquence à phase continue avec un indice de modulation $h = 1/2$: *Minimum Shift Keying* ou *MSK*

Pour cette modulation, l'indice h est égal à $1/2$ et les symboles a_i sont binaires (± 1). La fonction $g(t)$ de l'équation (2.39) est une porte d'amplitude

$1/2T$ et de durée T. Ainsi la fonction $q(t)$ est égale à :

$$\begin{array}{rcll} q(t) & = & 0 & t \leq 0 \\ q(t) & = & \frac{t}{2T} & 0 \leq t \leq T \\ q(t) & = & \frac{1}{2} & t \geq T \end{array} \tag{2.41}$$

La modulation *MSK* est une modulation de fréquence à phase continue et à réponse totale ($L = 1$).

Sur l'intervalle $[iT, (i+1)T[$, la phase $\phi(t)$ du signal *MSK* a pour expression :

$$\phi(t) = \frac{\pi}{2} a_i \frac{(t - iT)}{T} + \frac{\pi}{2} \sum_{n=-\infty}^{i-1} a_n \tag{2.42}$$

L'évolution de la phase $\phi(t)$ en fonction du temps est représentée sur la figure 2.10. On peut remarquer que la phase $\phi(t)$ varie linéairement sur un intervalle de durée T et qu'il n'y a pas de discontinuité aux instants iT.

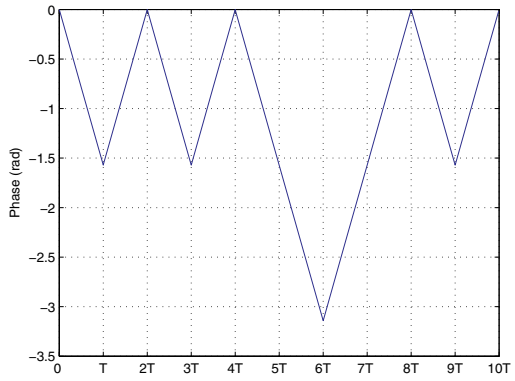


Figure 2.10 – Variation de la phase d'un signal *MSK* en fonction du temps.

En utilisant les expressions (2.37) et (2.42), le signal *MSK* peut s'écrire sous la forme :

$$S(t) = A \cos \left[2\pi (f_0 + \frac{a_i}{4T}) t - i \frac{\pi}{2} a_i + \theta_i + \varphi_0 \right] \quad iT \leq t < (i+1)T \tag{2.43}$$

avec :

$$\theta_i = \frac{\pi}{2} \sum_{n=-\infty}^{i-1} a_i \tag{2.44}$$

Le signal *MSK* utilise deux fréquences instantanées pour transmettre les symboles binaires $a_i = \pm 1$.

$$\begin{array}{l} f_1 = f_0 + \frac{1}{4T} \quad \text{si} \quad a_i = +1 \\ f_2 = f_0 - \frac{1}{4T} \quad \text{si} \quad a_i = -1 \end{array} \tag{2.45}$$

L'excursion de fréquence est alors la moitié du débit binaire, $\Delta f = f_1 - f_2 = 1/2T$. C'est l'écart de fréquence minimal qui permettent l'utilisation d'un démodulateur cohérent, d'où l'adjectif *minimum* dans le nom de la modulation.

Le signal modulé *MSK* peut aussi s'écrire sous la forme :

$$\begin{aligned} S(t) = A \Bigg[& \sum_i c_{2i-1} h(t-2iT) \cos \tfrac{\pi t}{2T} \cos(2\pi f_0 t + \varphi_0) \\ & - \sum_i c_{2i} h(t-(2i+1)T) \sin \tfrac{\pi t}{2T} \sin(2\pi f_0 t + \varphi_0) \Bigg] \end{aligned} \tag{2.46}$$

où les symboles c_i se déduisent des symboles a_i par un codage par transition.

$$c_{2i} = a_{2i} c_{2i-1} \quad \text{et} \quad c_{2i-1} = a_{2i-1} c_{2i-2} \tag{2.47}$$

et $h(t)$ est une porte d'amplitude unité, mais de durée $2T$.

$$\begin{aligned} h(t) &= 1 \quad \text{si } t \in [-T, T[\\ &= 0 \quad \text{ailleurs} \end{aligned} \tag{2.48}$$

La modulation *MSK* peut être vue comme une modulation en amplitude des termes $\cos \frac{\pi t}{2T} \cos(2\pi f_0 t + \varphi_0)$ et $-\sin \frac{\pi t}{2T} \sin(2\pi f_0 t + \varphi_0)$ par deux trains numériques binaires $u_c(t) = \sum_i c_{2i-1} h(t-2iT)$ et $u_s(t) = \sum_i c_{2i} h(t-(2t+1)T)$ dont les transitions sont décalées de T. Chaque train numérique permet de transmettre une donnée binaire toutes les $2T$ secondes et ainsi, le débit binaire acheminé par une modulation *MSK* est de $D = 1/T_b$ avec $T = T_b$.

La modulation *MSK* est un cas particulier des modulations *CP M-FSK* puisqu'elle est linéaire. La dsp, où l'on peut noter que contrairement à (2.36) il n'y a pas de raies spectrales, vaut :

$$\gamma_S(f) = \frac{1}{4}\gamma(f - f_0) + \frac{1}{4}\gamma(-f - f_0) \tag{2.49}$$

avec :

$$\gamma(f) = \frac{16A^2T}{\pi^2} \left(\frac{\cos 2\pi fT}{1 - 16f^2T^2} \right)^2 \tag{2.50}$$

Sur la figure 2.11 est représentée la dsp de l'enveloppe complexe d'un signal *MSK* exprimée en dB en fonction de la fréquence normalisée fT_b. Nous y avons aussi tracé la dsp de l'enveloppe complexe d'un signal MDP-4. Pour que la comparaison de ces deux densités spectrales de puissance ait un sens, nous avons supposé que le débit était identique pour ces deux modulations (soit $T = 2T_b$ pour la modulation MDP-4).

La largeur du lobe principal de la densité spectrale de puissance d'une modulation *MSK* est de $3/2T_b$ alors qu'elle n'est que de $1/T_b$ pour la modulation MDP-4. Ainsi, pour un même débit transmis, le lobe principal de la modulation *MSK* occupe 50% de bande de plus que celui de la modulation MDP-4. Toutefois, l'enveloppe de la dsp d'un signal *MSK* décroît en f^{-4} alors qu'elle ne décroît qu'en f^{-2} pour la modulation MDP-4. Ceci a pour conséquence que la bande de fréquences B qui contient 99% de la puissance du signal modulé pour la *MSK* est de $1,2/T_b$ alors qu'elle est d'environ $8/T_b$ pour la MDP-4.

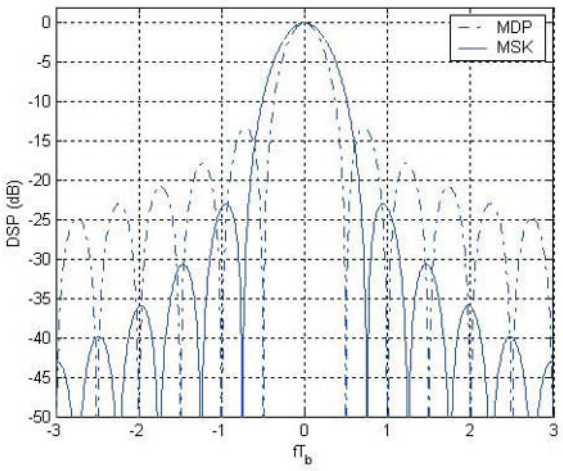


Figure 2.11 – Densité spectrale de puissance de l'enveloppe complexe des signaux *MSK* et MDP-4.

Modulation en cosinus surélevé : *Raised Cosine pulse* ou *L-RC*

Pour cette modulation, la fonction $g(t)$ de l'équation (2.39) a pour expression :

$$\begin{array}{rcll} g(t) & = & \frac{1}{2LT}(1-\cos\frac{2\pi t}{LT}) & \text{pour } 0 \leq t \leq LT \\ & = & 0 & \text{ailleurs} \end{array} \tag{2.51}$$

La densité spectrale de puissance de cette modulation décroît d'autant plus rapidement que le paramètre L est grand. Par exemple, la dsp est à -40 dB pour une modulation $2RC$ ($h = 1/2$, $a_i = \pm 1$) pour $fT = 1,2$ alors que pour une modulation $4RC$ ($h = 1/2$,$a_i = \pm 1$), ce niveau est atteint pour $fT = 0,7$. La fonction $q(t)$ est égale à :

$$\begin{array}{rcll} q(t) & = & \frac{t}{2LT} - \frac{1}{4\pi}\sin\frac{2\pi t}{LT} & 0 \leq t \leq LT \\ & = & \frac{1}{2} & t > LT \end{array} \tag{2.52}$$

Modulation *Gaussian Minimum Shift Keying* ou *GMSK*

L'indice h de cette modulation est égal à $1/2$ et les symboles a_i sont binaires (± 1). La fonction $g(t)$ de l'équation (2.39) est définie par le produit de convolution :

$$g(t) = \zeta(t) * \chi(t) \tag{2.53}$$

où $\zeta(t)$ est une porte d'amplitude $1/T$ ($T = T_b$) sur l'intervalle $[-T/2, T/2]$ et $\chi(t)$ est la réponse impulsionnelle d'un filtre gaussien de bande, B_g, à mi-densité spectrale :

$$\chi(t) = \sqrt{\frac{2\pi}{\ln 2}} B_g \exp(-2\pi^2 B_g^2 t^2)/\ln 2 \tag{2.54}$$

Après calcul du produit de convolution, la fonction $g(t)$ peut s'écrire sous la forme :

$$g(t) = \frac{1}{2T}\left[\text{erf}\left(\pi B_g \sqrt{\frac{2}{\ln 2}}(t+\frac{T}{2})\right) - \text{erf}\left(\pi B_g \sqrt{\frac{2}{\ln 2}}(t-\frac{T}{2})\right)\right] \tag{2.55}$$

où $\text{erf}(x)$ représente la fonction d'erreur définie par :

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-u^2) du$$

La figure 2.12 représente la variation de $g(t)$ en fonction de la variable normalisée t/T, pour différentes valeurs de la bande passante normalisée $B_N = B_g T$. On notera que le graphe de la fonction $g(t)$ a été décalé de $2T$ pour $B_N = 0,2$ et de $1,5T$ pour $B_N = 0,3$.

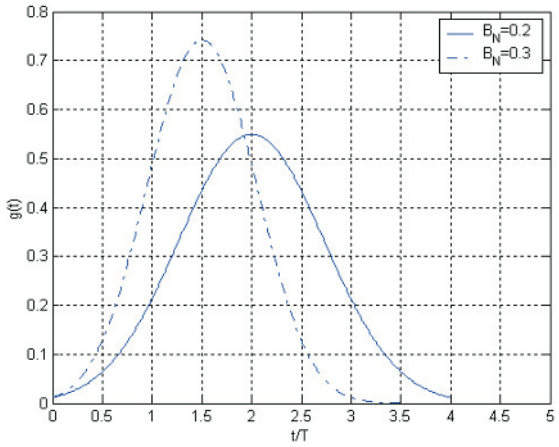


Figure 2.12 – Variation de la fonction $g(t)$ pour deux valeurs de B_N.

Le terme B_N permet de fixer l'étalement temporel de la fonction $g(t)$. Ainsi pour $B_N = 0,2$, cette fonction est approximativement de durée $4T$ alors que sa durée est seulement de $3T$ pour $B_N = 0,3$. Lorsque B_N tend vers l'infini, elle devient une porte de durée T (cas de la modulation *MSK*). La décroissance du spectre est d'autant plus rapide loin de la fréquence centrale que les dérivées du signal s'annulent jusqu'à un ordre plus élevé. Le produit de convolution (2.53) est un moyen d'augmenter sa concentration temporelle en

conservant la continuitée des dérivées de tous ordres. La gaussienne (*Gaussian MSK*) est la meilleure de ce point de vue puisque ses dérivées de tous ordres s'annulent en son centre, mais son support temporel est infini. La modulation *GMSK* est donc une modulation de fréquence à phase continue et à réponse partielle ($L > 1$).

Sur l'intervalle $[iT, (i+1)T[$, la phase $\phi(t)$ du signal *GMSK* est égale à :

$$\phi(t) = \pi \sum_{n=i-L+1}^{i} a_n q(t - nT) + \frac{\pi}{2} \sum_{n=-\infty}^{i-L} a_n \tag{2.56}$$

où $L = 3$ si $B_N = 0,3$ et $L = 2$ si $B_N = 0,2$.
Ainsi sur un intervalle $[iT, (i+1)T[$, la phase $\phi(t)$ du signal *GMSK* dépend du symbole a_i mais aussi des symboles antérieurs au symbole a_i (a_{i-1}, a_{i-2}, $\cdots$, a_{i-L+1}). Cette modulation non linéaire présente un effet de mémoire qui lui confère de bonnes propriétés spectrales. La modulation *GMSK*, avec une bande passante normalisée $B_N = 0,3$ a été retenue pour le système de téléphonie mobile GSM (Groupe Spécial Mobile et plus tard *Global System for Mobile communication*). Précisons qu'il n'existe pas d'expression simple de la densité spectrale de puissance d'un signal *GMSK*. Pour des valeurs de la bande passante normalisée de 0,3 ou de 0,2, la densité spectrale de puissance du signal *GMSK* ne présente pas de lobes secondaires et sa décroissance en fonction de la fréquence est très rapide. Ainsi à -10 dB la bande occupée par le signal *GMSK* est approximativement de 200 kHz, et à -40 dB de 400 kHz pour le débit normalisé $D = 271$ kbit/s.

2.2 Structure et performances du récepteur optimal sur canal gaussien

L'objet de ce chapitre est de déterminer la structure et les performances du récepteur optimal pour des modulations avec et sans mémoire sur canal à bruit additif blanc gaussien (BABG). Le type de récepteur considéré est le *récepteur cohérent* où la fréquence et la phase des signaux émis par le modulateur sont supposées connues du récepteur. En effet, le récepteur cohérent est capable de générer localement des signaux ayant la même fréquence et la même phase que ceux délivrés par le modulateur contrairement au récepteur dit *récepteur non cohérent* ou *différentiel*.

De manière générale, le récepteur est constitué d'un démodulateur dont l'objet est de translater en bande de base le signal modulé, c'est-à-dire supprimer la fréquence porteuse, et d'un circuit de décision chargé d'estimer les blocs de données transmis. Le récepteur est optimal au sens où il garantit une probabilité d'erreur minimale sur les blocs de données estimés.

2.2.1 Structure du récepteur cohérent

Soient $s_j(t)$, $j = 1, 2, \cdots, M$ les signaux émis sur le canal de transmission perturbé par un BABG $b(t)$, de moyenne nulle et de densité spectrale de puissance bilatérale égale à $N_0/2$. Sur l'intervalle de temps $[0, T[$, le signal reçu par le récepteur est égal à :

$$r(t) = s_j(t) + b(t)$$

Les M signaux $s_j(t)$ engendrent un espace de dimension $N \leqslant M$, et peuvent être représentés sous forme d'une série de fonctions $\nu_p(t)$ orthonormées et pondérées.

$$s_j(t) = \sum_{p=1}^{N} s_{jp}\nu_p(t)$$

où s_{jp} est un scalaire égal à la projection du signal $s_j(t)$ sur la fonction $\nu_p(t)$.

$$s_{jp} = \int_0^T s_j(t)\nu_p(t)dt$$

Le bruit peut aussi être représenté sous forme d'une série de fonctions orthonormées mais de longueur infinie (développement de Karhunen Loeve). Lorsque le bruit est blanc, on montre que les fonctions orthonormées peuvent être choisies de façon arbitraire. Nous allons donc prendre les mêmes fonctions orthonormées que celles utilisées pour représenter les signaux $s_j(t)$, mais après extension à l'infini de cette base de fonctions :

$$b(t) = \sum_{p=1}^{\infty} b_p\nu_p(t) = \sum_{p=1}^{N} b_p\nu_p(t) + b'(t)$$

où b_p est un scalaire égal à la projection de $b(t)$ sur la fonction $\nu_p(t)$.

$$b_p = \int_0^T b(t)\nu_p(t)dt$$

Les quantités b_p sont des variables aléatoires gaussiennes, de moyenne nulle, non corrélées et de variance $\sigma^2 = N_0/2$.

$$\mathrm{E}\{b_p b_n\} = \int_0^T\int_0^T \mathrm{E}\{b(t)b(t')\}\nu_p(t)\nu_n(t')dtdt'$$

Le bruit étant blanc, $\mathrm{E}\{b(t)b(t')\} = \frac{N_0}{2}\delta(t-t')$ et ainsi :

$$\mathrm{E}\{b_p b_n\} = \frac{N_0}{2}\int_0^T \nu_p(t)\nu_n(t)dt = \frac{N_0}{2}\delta_{n,p} \tag{2.57}$$

où $\delta_{n,p}$ est le symbole de Kronecker, égal à 1 si $n = p$ et à 0 si $n \neq p$.

En utilisant les représentations des signaux $s_j(t)$ et du bruit $b(t)$ par leur série respective, on peut écrire :

$$r(t) = \sum_{p=1}^{N}(s_{jp} + b_p)\nu_p(t) + \sum_{p=N+1}^{\infty} b_p\nu_p(t) = \sum_{p=1}^{N} r_p\nu_p(t) + b'(t)$$

Conditionnellement à l'émission du signal $s_j(t)$, les quantités r_p sont des variables aléatoires gaussiennes, de moyenne et de variance $N_0/2$. Elles sont non corrélées au bruit $b'(t)$. En effet, nous avons :

$$\mathrm{E}\left\{r_p b'(t)\right\} = \mathrm{E}\left\{(s_{jp} + b_p)\sum_{n=N+1}^{\infty} b_n\nu_n(t)\right\} \quad \forall\, p = 1,\, 2,\, \cdots,\, N$$

En tenant compte du fait que les variables b_n, quel que soit n, sont à moyenne nulle et non corrélées, on obtient :

$$\mathrm{E}\left\{r_p b'(t)\right\} = \sum_{n=N+1}^{\infty} \mathrm{E}\left\{b_p b_n\right\}\nu_n(t) = 0 \quad \forall\, p = 1,\, 2,\, \cdots,\, N \tag{2.58}$$

Les quantités r_p et le bruit $b'(t)$ sont donc indépendants puisque gaussiens.

En conclusion, le récepteur optimal peut baser sa décision sur l'observation des seules quantités r_p $p = 1,\, 2,\, \cdots,\, N$ avec :

$$r_p = \int_0^T r(t)\nu_p(t)dt \tag{2.59}$$

Le passage du signal $r(t)$ délivré par le canal de transmission aux N quantités r_p s'appelle la démodulation.

Exemple

Considérons une modulation MDP-M pour laquelle les signaux $s_j(t)$ sont de la forme :

$$s_j(t) = Ah(t)\cos(2\pi f_0 t + \varphi_0 + \phi_j)$$

Les signaux $s_j(t)$ engendrent un espace à $N = 2$ dimensions si $M > 2$. Les fonctions $\nu_p(t) \quad p = 1,\, 2$ orthonormées ont respectivement pour expressions :

$$\begin{aligned} \nu_1(t) &= \sqrt{\tfrac{2}{T}}\cos(2\pi f_0 t + \varphi_0) \\ \nu_2(t) &= \sqrt{\tfrac{2}{T}}\sin(2\pi f_0 t + \varphi_0) \end{aligned}$$

et les signaux $s_j(t)$ peuvent s'écrire :

$$s_j(t) = A\sqrt{\frac{T}{2}}\cos\phi_j h(t)\nu_1(t) - A\sqrt{\frac{T}{2}}\sin\phi_j h(t)\nu_2(t)$$

Après démodulation, l'observation $R = (r_1, r_2)$ est égale à :

$$r_1 = A\sqrt{\frac{T}{2}}\cos\phi_j + b_1 \quad r_2 = A\sqrt{\frac{T}{2}}\sin\phi_j + b_2$$

L'observation $R = (r_1, r_2)$ ne dépend plus que des états de phase ϕ_j et du bruit car la fréquence porteuse f_0 a été supprimée. On dit que l'observation $R = (r_1, r_2)$ est en *bande de base.*

L'opération de démodulation nécessite de connaître la fréquence f_0 et la phase φ_0 de la porteuse, les signaux $\nu_p(t)$ devant être synchrones avec la porteuse générée par le modulateur. C'est la raison pour laquelle on parle de démodulation synchrone ou de démodulation cohérente.

Les N intégrateurs du démodulateur peuvent être remplacés par N filtres de réponse impulsionnelle $h(T-t)$ suivis chacun d'un échantillonneur à l'instant $t = T$.

$$s_j(t)\nu_j(t) * h(T-t) = \int_{-\infty}^{+\infty} s_j(\tau)\nu_j(\tau)h(T-t+\tau)d\tau$$

En échantillonnant en $t = T$, on obtient :

$$s_j(t)\nu_j(t) * h(T-t)\,|_{t=T} = \int_0^T s_j(\tau)\nu_j(\tau)d\tau$$

ce qui est bien égal à la sortie de l'intégrateur.
Le filtre de réponse impulsionnelle $h(T-t)$ est appelé le *filtre adapté* au signal $h(t)$ de durée T. On peut montrer que ce filtre maximise le rapport signal à bruit à sa sortie à l'instant $t = T$.

Pour une transmission à flot continu de données, l'intégration se fait sur chaque intervalle $[iT,\ (i+1)T[$ $i = 1, 2, \cdots$ et, si on utilise des filtres adaptés, l'échantillonnage est réalisé aux instants $(i+1)T$.

Après démodulation, le récepteur doit prendre une décision sur le groupe de données émis sur chaque intervalle de temps $[iT,\ (i+1)T[$. Pour cela, il recherche le signal $s_j(t)$ le plus vraisemblable en utilisant la règle de décision du maximum de vraisemblance *a posteriori* :

$$\hat{s}_j(t) \quad \text{si} \quad \Pr\{s_j(t)\,|\,R\} > \Pr\{s_p(t)\,|\,R\} \quad \forall\, p \neq j \quad p = 1, 2, \cdots, M$$

où $\hat{s}_j(t)$ est le signal retenu et $R = (r_1 \cdots r_p \cdots r_N)$ la sortie du démodulateur. Pour simplifier les notations, la marque du temps a été omise pour les composantes de l'observation R. $\Pr\{s_j(t)/R\}$ désigne la probabilité de $s_j(t)$ conditionnellement à la connaissance de l'observation R.

En utilisant la règle de Bayes, le maximum de vraisemblance *a posteriori* peut encore s'écrire :

$$\hat{s}_j(t) \quad \text{si} \quad \pi_j p(R|s_j(t)) > \pi_p p(R|s_p(t)) \quad \forall\, p \neq j \quad p = 1,\, 2,\, \cdots,\, M$$

où $\pi_j = \Pr\{s_j(t)\}$ désigne la probabilité *a priori* d'émettre le signal $s_j(t)$ et $p(R|s_j(t))$ est la densité de probabilité de l'observation R conditionnellement à l'émission par le modulateur du signal $s_j(t)$.

En tenant compte du fait que les composantes $r_p = s_{jp} + b_p$ de l'observation R conditionnellement à l'émission du signal $s_j(t)$ sont gaussiennes, non corrélées, de moyenne s_{jp} et de variance $N_0/2$, on peut écrire :

$$\begin{aligned}\hat{s}_j(t) \quad \text{si} \quad & \pi_j \prod_{p=1}^{N} p(r_p|s_j(t)) > \pi_n \prod_{p=1}^{N} p(r_p|s_n(t)) \\ & \forall n \neq j \quad p = 1,\, 2,\, \cdots,\, M\end{aligned}$$

En remplaçant les densités de probabilité par leurs expressions respectives on obtient :

$$\begin{aligned}\hat{s}_j(t) \quad \text{si} \quad & \pi_j \left(\tfrac{1}{\sqrt{\pi N_0}}\right)^N \exp\left(-\tfrac{1}{N_0} \sum_{p=1}^{N} (r_p - s_{jp})^2\right) \\ & > \pi_n \left(\tfrac{1}{\sqrt{\pi N_0}}\right)^N \exp\left(-\tfrac{1}{N_0} \sum_{p=1}^{N} (r_p - s_{np})^2\right)\end{aligned}$$

Après simplification :

$$\hat{s}_j(t) \quad \text{si} \quad \sum_{p=1}^{N} r_p s_{jp} + C_j > \sum_{p=1}^{N} r_p s_{np} + C_n \;\forall\, n \neq j \;\; n = 1,\, 2,\, \cdots,\, M \quad (2.60)$$

où $C_j = \frac{N_0}{2} \ln(\pi_j) - \frac{E_j}{2}$ avec $E_j = \sum_{p=1}^{N} (s_{jp})^2$.

En remarquant que :

$$\int_0^T r(t) s_j(t) dt = \int_0^T \sum_{p=1}^{N} r_p \nu_p(t) \sum_{m=1}^{N} s_{jm} \nu_m(t) dt$$

et en se rappelant que les fonctions $\nu_p(t)$ sont orthonormées, on obtient :

$$\int_0^T r(t) s_j(t) dt = \sum_{p=1}^{N} r_p s_{jp}$$

De la même façon :

$$\int_0^T s_j^2(t) dt = \int_0^T \sum_{p=1}^{N} s_{jp} \nu_p(t) \sum_{m=1}^{N} s_{jm} \nu_m(t) dt$$

et finalement :

$$\int_0^T s_j^2(t)dt = \sum_{p=1}^{N} s_{jp}^2$$

En tenant compte de ce qui précède, la règle de décision du maximum de vraisemblance *a posteriori* peut encore se mettre sous la forme :

$$\hat{s}_j(t) \text{ si } \int_0^T r(t)s_j(t)dt + C_j > \int_0^T r(t)s_n(t)dt + C_n \quad \forall\, n \neq j \quad n = 1, 2, \cdots, M \tag{2.61}$$

où $C_j = \frac{N_0}{2}\ln(\pi_j) - \frac{E_j}{2}$ avec $E_j = \int_0^T s_j^2(t)dt$.

Si tous les signaux $s_j(t)$ sont émis avec la même probabilité ($\pi_j = 1/M$), le terme C_j se réduit à $-E_j/2$. De plus, si tous les signaux $s_j(t)$ ont de plus la même énergie $E_j = E$ (cas des modulations par déplacement de la phase ou de la fréquence), alors la règle de décision du maximum de vraisemblance *a posteriori* se simplifie et devient :

$$\hat{s}_j(t) \text{ si } \int_0^T r(t)s_j(t)dt > \int_0^T r(t)s_n(t)dt \quad \forall\, n \neq j \quad n = 1, 2, \cdots, M \tag{2.62}$$

2.2.2 Performances du récepteur cohérent

Modulation par déplacement d'amplitude à M états

Pour une modulation MDA-M, les signaux $s_j(t)$ sont de la forme :

$$s_j(t) = A_j h(t)\cos(2\pi f_0 t + \varphi_0)$$

avec :

$$A_j = (2j - 1 - M)A \quad j = 1, 2, \cdots, M$$

Ils engendrent un espace de dimension $N = 1$ et ainsi l'observation R en sortie du démodulateur se réduit à une composante r.

$$r = \int_0^T r(t)\nu(t)dt$$

avec $\nu(t) = \sqrt{\frac{2}{T}}\cos(2\pi f_0 t + \varphi_0)$.

Pour une transmission à flot continu de données, l'estimation des symboles a_i se fait en intégrant le produit $r(t)\nu(t)$ sur chaque intervalle de temps $[iT, (i+1)T[$. Si un filtre adapté est utilisé plutôt qu'un intégrateur, l'échantillonnage en sortie du filtre est réalisé aux instants $(i+1)T$.

En se plaçant sur l'intervalle $[0, T[$ et en supposant les données d'informations d_i *iid,* tous les états d'amplitude sont équiprobables et la règle de décision(2.60) conduit à :

$$\hat{A}_j \text{ si } rs_j - \frac{1}{2}s_j^2 > rs_n - \frac{1}{2}s_n^2 \quad \forall\, n \neq j \tag{2.63}$$

où $\hat{A}_j$ désigne l'amplitude estimée avec :

$$s_j = \int_0^T A_j \cos(2\pi f_0 t + \varphi_0)\nu(t)dt = A_j\sqrt{\frac{T}{2}} \text{ si } f_0 >> \frac{1}{T} \tag{2.64}$$

Le récepteur cohérent, représenté dans la figure 2.13, prend sa décision en comparant l'observation r à un ensemble de $(M-1)$ seuils de la forme :

$$\begin{aligned} -(M-2)A\sqrt{\tfrac{T}{2}}, \cdots, -2pA\sqrt{\tfrac{T}{2}}, \cdots, -2A\sqrt{\tfrac{T}{2}}, 0, \\ 2A\sqrt{\tfrac{T}{2}}, \cdots, 2pA\sqrt{\tfrac{T}{2}}, \cdots, (M-2)A\sqrt{\tfrac{T}{2}} \end{aligned} \tag{2.65}$$

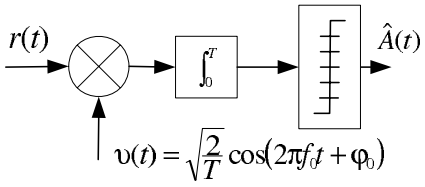


Figure 2.13 – Récepteur cohérent pour modulation MDA-M.

Exemple

Considérons une modulation MDA-4, les trois seuils étant $-2A\sqrt{\frac{T}{2}}$, 0, $2A\sqrt{\frac{T}{2}}$. Les décisions sont les suivantes :

$$\begin{aligned} \hat{A}_j = -3A \quad &\text{si} \quad r < -2A\sqrt{\tfrac{T}{2}} \\ \hat{A}_j = -A \quad &\text{si} \quad -2A\sqrt{\tfrac{T}{2}} < r < 0 \\ \hat{A}_j = A \quad &\text{si} \quad 0 < r < 2A\sqrt{\tfrac{T}{2}} \\ \hat{A}_j = 3A \quad &\text{si} \quad r > 2A\sqrt{\tfrac{T}{2}} \end{aligned}$$

L'émission d'un état d'amplitude A_j correspond à la transmission d'un groupe de $\log_2(M)$ données binaires d_i. La probabilité d'erreur sur un groupe de données peut se calculer comme la valeur moyenne des probabilités d'erreur conditionnelles Pe_{2j-1-M} données par :

$$Pe_{2j-1-M} = \Pr\left\{\hat{A}_j \neq (2j-1-M)A \,|\, A_j = (2j-1-M)A\right\}$$

La probabilité d'erreur moyenne sur les symboles, notée Pes, est égale à :

$$Pes = \frac{1}{M}\sum_{j=1}^{M} Pe_{2j-1-M}$$

Les probabilités d'erreur conditionnelles peuvent être classées en deux types. Le premier type correspond aux probabilités que l'observation dépasse ou soit inférieure à un certain seuil et le second, aux probabilités que l'observation ne soit pas comprise entre deux seuils.

TYPE 1 : Probabilités que l'observation dépasse ou soit inférieure à un seuil

$$\begin{aligned} Pe_{(M-1)} &= \Pr\left\{\hat{A}_j \neq (M-1)A \,|\, A_j = (M-1)A\right\} \\ &= \Pr\left\{r < (M-2)A\sqrt{\tfrac{T}{2}} \,|\, A_j = (M-1)A\right\} \end{aligned}$$

$$\begin{aligned} Pe_{-(M-1)} &= \Pr\left\{\hat{A}_j \neq -(M-1)A \,|\, A_j = -(M-1)A\right\} \\ &= \Pr\left\{r > -(M-2)A\sqrt{\tfrac{T}{2}} \,|\, A_j = -(M-1)A\right\} \end{aligned}$$

TYPE 2 : Probabilités que l'observation ne soit pas comprise entre deux seuils

$$Pe_{2j-1-M} = \Pr\left\{\hat{A}_j \neq (2j-1-M)A \,|\, A_j = (2j-1-M)A\right\}$$

$$\begin{aligned} Pe_{2j-1-M} = 1- \quad \Pr\Big\{&(2j-2-M)A\sqrt{\tfrac{T}{2}} < r < (2j-M)A\sqrt{\tfrac{T}{2}} \,| \\ &A_j = (2j-1-M)A\Big\} \end{aligned}$$

L'observation r est gaussienne conditionnellement à une réalisation de l'amplitude A_j, de moyenne $\pm A_j\sqrt{T/2}$ et de variance $N_0/2$. Les probabilités conditionnelles ont pour expressions :

$$Pe_{(M-1)} = Pe_{-(M-1)} = \frac{1}{2}\mathrm{erfc}\sqrt{\frac{A^2T}{2N_0}}$$

$$Pe_{(2j-1-M)} = \mathrm{erfc}\sqrt{\frac{A^2T}{2N_0}}$$

où la fonction d'erreur complémentaire est toujours définie par :

$$\mathrm{erfc}(x) = 1 - \mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_{x}^{+\infty} \exp(-u^2)du \tag{2.66}$$

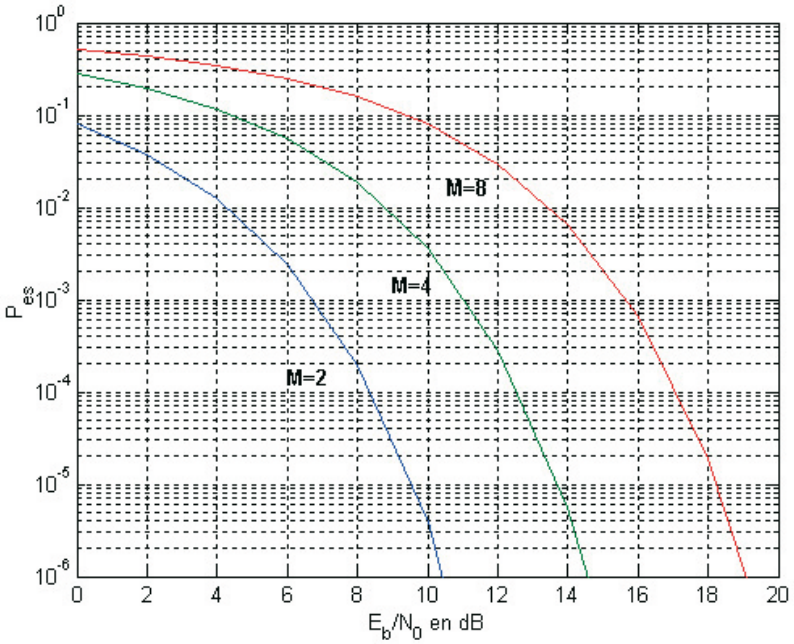


Figure 2.14 – Probabilité d'erreur moyenne Pes en fonction du rapport E_b/N_0 pour différentes valeurs du paramètre M d'une modulation MDA-M.

Pour le calcul de la probabilité d'erreur moyenne sur les groupes de données, nous avons deux probabilités conditionnelles de type 1 et $(M-2)$ probabilités conditionnelles de type 2.

$$Pes = \frac{M-1}{M}\text{erfc}\sqrt{\frac{A^2T}{2N_0}}$$

En introduisant l'énergie moyenne $E_s = \frac{A^2T}{2}\frac{(M^2-1)}{3}$ reçue par groupe de données, la probabilité d'erreur moyenne est encore égale à :

$$Pes = \frac{M-1}{M}\text{erfc}\sqrt{\frac{3}{M^2-1}\frac{E_s}{N_0}}$$

La probabilité d'erreur Pes peut aussi s'exprimer en fonction de l'énergie moyenne $E_b = E_s/\log_2(M)$ reçue par donnée d_i :

$$Pes = \frac{M-1}{M}\text{erfc}\sqrt{\frac{3\log_2(M)}{M^2-1}\frac{E_b}{N_0}}$$

ou encore en fonction de la puissance moyenne P reçue et du débit $D = 1/T_b$ transmis :

$$Pes = \frac{M-1}{M}\text{erfc}\sqrt{\frac{3\log_2(M)}{M^2-1}\frac{P}{N_0 D}} \tag{2.67}$$

La figure 2.14 fournit la probabilité d'erreur moyenne Pes en fonction du rapport E_b/N_0 pour différentes valeurs du paramètre M.

La probabilité d'erreur par donnée binaire Peb peut se déduire de la probabilité d'erreur moyenne Pes dans le cas où un codage de Gray est utilisé et sous l'hypothèse d'un rapport signal à bruit suffisamment élevé. En effet, dans ce cas on a généralement une donnée binaire fausse parmi les $\log_2(M)$ données transmises. (on suppose que l'amplitude du symbole reçu a une valeur immédiatement inférieure ou supérieure à la valeur de l'amplitude émise).

$$Peb \cong \frac{Pes}{\log_2(M)} \quad \text{si} \quad \frac{E_b}{N_0} >> 1 \tag{2.68}$$

Modulation par déplacement de phase à M états

Pour une modulation MDP-M, les signaux $s_j(t)$ sont de la forme :

$$s_j(t) = A\cos(2\pi f_0 t + \varphi_0 + \phi_j) \tag{2.69}$$

avec :

$$\phi_j = (2j+1)\frac{\pi}{M} + \theta_0 \quad j = 0,\, 1,\, \cdots,\, (M-1)$$

Les signaux $s_j(t)$, pour $M > 2$, engendrent un espace à deux dimensions. L'observation R en sortie du démodulateur est donc constituée de deux composantes (r_1, r_2) avec :

$$r_1 = \int_0^T r(t)\nu_1(t)dt \quad r_2 = \int_0^T r(t)\nu_2(t)dt$$

où $\nu_1(t) = \sqrt{\frac{2}{T}}\cos(2\pi f_0 t + \varphi_0)$ et $\nu_2(t) = -\sqrt{\frac{2}{T}}\sin(2\pi f_0 t + \varphi_0)$.
En utilisant la règle de décision (2.21) et en supposant les données d'information iid, tous les états de phase sont équiprobables et la décision est la suivante :

$$\hat{\phi}_j \quad \text{si} \quad \sum_{p=1}^{2} r_p s_{jp} > \sum_{p=1}^{2} r_p s_{np} \quad \forall n \neq j \tag{2.70}$$

avec :

$$s_{j1} = A\sqrt{\frac{T}{2}}\cos\phi_j \quad \text{et} \quad s_{j2} = A\sqrt{\frac{T}{2}}\sin\phi_j \quad \text{si} \quad f_0 >> \frac{1}{T} \tag{2.71}$$

En tenant compte des expressions de s_{j1} et de s_{j2}, la règle de décision peut encore s'écrire :

$$\hat{\phi}_j \quad \text{si} \quad r_1\cos\phi_j + r_2\sin\phi_j > r_1\cos\phi_n + r_2\sin\phi_n \quad \forall n \neq j \tag{2.72}$$

Le récepteur cohérent pour une modulation MDP-M est représenté en figure 2.15. Il est constitué de deux voies dites en phase (projection du signal reçu sur $\nu_1(t) = \sqrt{2/T}\cos(2\pi f_0 t + \varphi_0)$) et en quadrature (projection du signal reçu sur $\nu_2(t) = \sqrt{2/T}\sin(2\pi f_0 t + \varphi_0)$) et d'un circuit de décision.

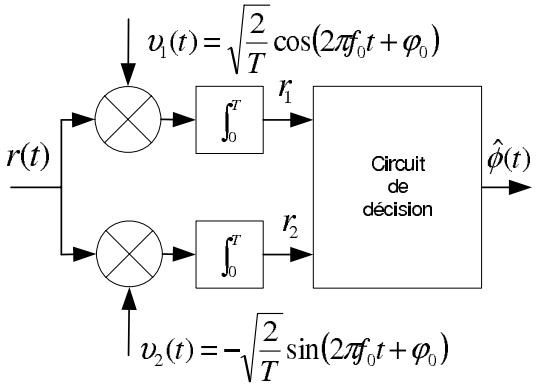


Figure 2.15 – Récepteur cohérent pour modulation MDP-M.

L'émission par le modulateur d'un état de phase correspond à la transmission d'un groupe de $\log_2(M)$ données. La probabilité d'erreur sur un groupe de données binaires, quelle que soit la valeur de M, ne possède pas d'expression analytique. Toutefois, à fort rapport signal à bruit, cette probabilité est bien approchée par l'expression suivante :

$$Pes \cong \text{erfc}\left[\sqrt{\log_2(M)\frac{E_b}{N_0}}\sin\frac{\pi}{M}\right] \text{ si } \frac{E_b}{N_0} >> 1 \tag{2.73}$$

En notant que $E_b = PT_b$ et $D = 1/T_b$, le rapport E_b/N_0 est encore égal à P/N_0D où P est la puissance reçue du signal modulé.

Pour un codage de Gray, la probabilité d'erreur par donnée binaire à fort rapport signal à bruit est égale à :

$$Peb \cong \frac{Pes}{\log_2(M)} \quad \text{si} \quad \frac{\bar{E}_b}{N_0} >> 1 \tag{2.74}$$

Cas de la modulation MDP-2

Pour cette modulation, la phase ϕ_j prend les valeurs 0 ou π. Chaque état de la phase est donc associé à une donnée binaire. En adoptant le codage suivant :

$$\phi_j = 0 \rightarrow d_i = 1 \quad \phi_j = \pi \rightarrow d_i = 0$$

la règle de décision pour une modulation MDP-2 est simple :

$$\hat{d}_i = 1 \quad \text{si} \quad r_1 > 0 \qquad \hat{d}_i = 0 \quad \text{si} \quad r_1 < 0 \tag{2.75}$$

L'observation r_2 n'est pas utilisée pour le décodage des données d_i car l'espace engendré par les signaux modulés à deux états de phase est de dimension $N = 1$.

Pour la modulation MDP-2, il existe une expression exacte de la probabilité d'erreur Peb par donnée binaire. En supposant les données binaires iid, cette probabilité d'erreur est égale à :

$$Peb = \frac{1}{2}\Pr\{r_1 > 0 | \phi_j = \pi\} + \frac{1}{2}\Pr\{r_1 < 0 | \phi_j = 0\}$$

La sortie r_1 du démodulateur est :

$$r_1 = \pm\sqrt{E_b} + b$$

où $E_b = A^2T/2$ est l'énergie reçue par donnée binaire d'information transmise et b est un BABG, de moyenne nulle et de variance égale à $N_0/2$.

$$Peb = \tfrac{1}{2}\tfrac{1}{\sqrt{\pi N_0}} \int_0^{\infty} \exp(-\tfrac{1}{N_0}(r_1 + \sqrt{E_b})^2) dr_1 + \tfrac{1}{2}\tfrac{1}{\sqrt{\pi N_0}} \int_{-\infty}^{0} \exp\left(-\tfrac{1}{N_0}(r_1 - \sqrt{E_b})^2\right) dr_1$$

En introduisant la fonction d'erreur complémentaire, la probabilité d'erreur Peb est égale à :

$$Peb = \frac{1}{2}\mathrm{erfc}\sqrt{\frac{E_b}{N_0}} \tag{2.76}$$

Cas de la modulation MDP-4

Pour cette modulation, la phase ϕ_j prend quatre valeurs $\pi/4$, $3\pi/4$, $5\pi/4$, $7\pi/4$. À chaque état de la phase sont associées deux données binaires. Pour des états de phase équiprobables, la règle du maximum de vraisemblance *a posteriori* conduit aux décisions suivantes :

$$\hat{\phi}_j = \tfrac{\pi}{4} \quad \text{si} \quad r_1 > 0;\, r_2 > 0$$

$$\hat{\phi}_j = \tfrac{3\pi}{4} \quad \text{si} \quad r_1 < 0;\, r_2 > 0$$

$$\hat{\phi}_j = \tfrac{5\pi}{4} \quad \text{si} \quad r_1 < 0;\, r_2 < 0$$

$$\hat{\phi}_j = \tfrac{7\pi}{4} \quad \text{si} \quad r_1 > 0;\, r_2 < 0$$

En considérant le codage de Gray suivant :

$$\frac{\pi}{4} \to 11 \quad \frac{3\pi}{4} \to 01 \quad \frac{5\pi}{4} \to 00 \quad \frac{7\pi}{4} \to 10$$

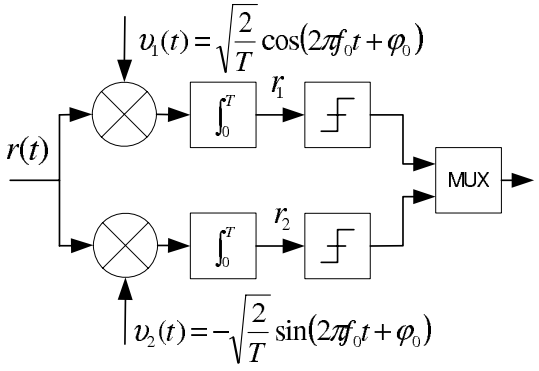


Figure 2.16 – Récepteur cohérent pour modulation MDP-4.

l'estimation des données binaires peut se faire en comparant séparément les sorties r_1 et r_2 du démodulateur à un seuil fixé à zéro. Le récepteur cohérent pour une modulation MDP-4 est représenté en figure 2.16.

Une expression exacte de Peb peut toutefois être donnée en observant simplement que le récepteur cohérent pour une modulation MDP-4 est constitué de deux voies identiques à celle d'un récepteur MDP-2. La probabilité d'erreur Peb pour une modulation MDP-4 est alors la même que pour une modulation MDP-2 soit :

$$Peb = \frac{1}{2}\text{erfc}\sqrt{\frac{E_b}{N_0}} \tag{2.77}$$

La probabilité d'erreur par couple binaire Pes est égale à :

$$Pes = 1 - (1 - Peb)^2$$

À fort rapport signal à bruit, la probabilité d'erreur Peb est très inférieure à l'unité et ainsi, pour la modulation MDP-4 :

$$Pes = 2Peb \quad \text{si} \quad \frac{E_b}{N_0} >> 1 \tag{2.78}$$

La figure 2.17 fournit la probabilité d'erreur Pes en fonction du rapport E_b/N_0 pour différentes valeurs du paramètre M[1].

Modulation d'amplitude sur deux porteuses en quadrature – MAQ-M

Pour la modulation MAQ-M, les signaux $s_j(t)$ délivrés par le modulateur sont de la forme :

$$s_j(t) = A_{jc}h(t)\cos(2\pi f_0 t + \varphi_0) - A_{js}h(t)\sin(2\pi f_0 t + \varphi_0) \tag{2.79}$$

[1] Pour $M = 2$, c'est la relation exacte (2.76) qui est utilisée car $Pes = Peb$, pour $M > 2$, Pes est fournit par l'équation approchée (2.73).

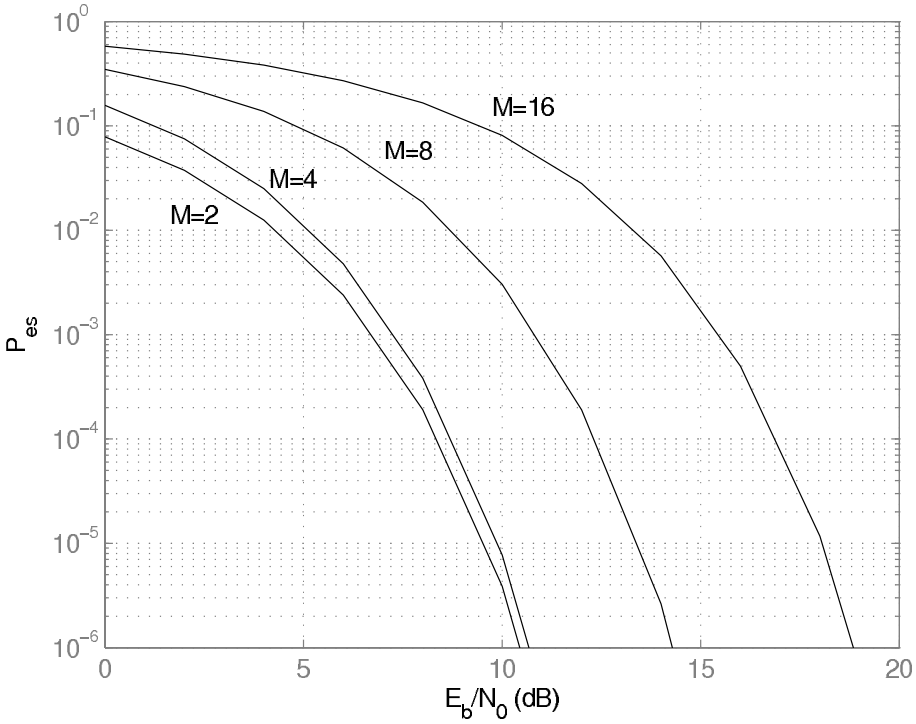


Figure 2.17 – Probabilité d'erreur Pes en fonction du rapport E_b/N_0 pour différentes valeurs de M d'une modulation MDP-M.

Deux situations peuvent se présenter selon que la longueur m des groupes de données à l'entrée du modulateur est paire ou non. Lorsque $M = 2^m$ avec m pair, le groupe de données peut être séparé en deux sous-groupes de longueur $m/2$, chaque sous-groupe étant associé respectivement aux amplitudes A_{jc} et A_{js} avec :

$$\begin{aligned} A_{jc} &= (2j - 1 - \sqrt{M})A \quad j = 1, 2, \cdots, \sqrt{M} \\ A_{js} &= (2j - 1 - \sqrt{M})A \quad j = 1, 2, \cdots, \sqrt{M} \end{aligned} \tag{2.80}$$

Dans ce cas, la modulation MAQ-M est équivalente à deux modulations MDA-$\sqrt{M}$ ayant des porteuses en quadrature. Le récepteur cohérent pour une modulation MAQ-M est constitué de deux voies dites en phase et en quadrature, chaque voie, semblable à un récepteur pour modulation MDA$-\sqrt{M}$, réalise l'estimation d'un groupe de $m/2$ données binaires. Le récepteur MAQ-M est représenté sur la figure 2.18.

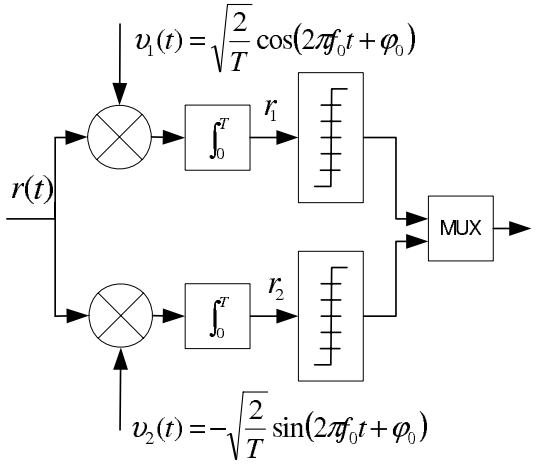


Figure 2.18 – Récepteur cohérent pour modulation MAQ-M avec $M = 2^m$ et m pair.

La probabilité d'erreur sur un groupe de $m/2$ données binaires est égale à la probabilité d'erreur d'une modulation MDA$-\sqrt{M}$ soit :

$$Pe_{m/2} = \frac{\sqrt{M}-1}{\sqrt{M}}\text{erfc}\sqrt{\frac{3\log_2(\sqrt{M})}{M-1}\frac{E_b}{N_0}} \tag{2.81}$$

La probabilité d'erreur de symbole Pes sur le groupe de m données binaires est donc égale à :

$$Pes = 1 - (1 - Pe_{m/2})^2$$

Lorsque m est impair, la probabilité d'erreur Pes sur un groupe de m données binaires peut être bornée supérieurement par l'expression suivante :

$$Pes \leqslant 2\text{erfc}\sqrt{\frac{3\log_2(\sqrt{M})}{M-1}\frac{E_b}{N_0}} \tag{2.82}$$

La probabilité d'erreur Peb par donnée binaire peut se déduire de Pes si un codage de Gray est utilisé et pour un rapport signal à bruit suffisamment élevé :

$$Peb = \frac{Pes}{\log_2(M)} \tag{2.83}$$

Pour les valeurs de $M \geqslant 8$, les performances des modulations MDP-M et MAQ-M peuvent facilement être comparées. En effet, en faisant l'approximation suivante :

$$\sin\left(\frac{\pi}{M}\right) \cong \frac{\pi}{M} \quad \text{si} \quad M \geqslant 8$$

la probabilité d'erreur par donnée binaire pour une modulation MDP-M peut s'écrire :

$$Peb = \frac{1}{\log_2(M)} \text{erfc} \sqrt{\log_2(M) \frac{\pi^2}{M^2} \frac{E_b}{N_0}} \quad \text{si} \quad \frac{E_b}{N_0} >> 1 \tag{2.84}$$

En négligeant les coefficients qui pondèrent les fonctions d'erreur complémentaires, la modulation MDP-M nécessite d'augmenter le rapport E_b/N_0 de :

$$10 \log \left(\frac{3M^2}{2(M-1)\pi^2} \right) \text{dB}$$

pour obtenir la même probabilité d'erreur que la modulation MAQ-M. Si on compare, par exemple, les performances de la modulation MDP-16 à celles de la modulation MAQ-16, on constate que la première nécessite environ 4 dB de plus pour le rapport E_b/N_0 pour obtenir les mêmes probabilités d'erreur.

Modulation par déplacement de fréquence – MDF-M

Pour une modulation MDF-M, le modulateur délivre des signaux de la forme :

$$s_j(t) = Ah(t) \cos(2\pi f_j t + \varphi_j) \tag{2.85}$$

où les fréquences f_j sont choisies de manière à ce que les M signaux $s_j(t)$ soient orthogonaux. L'espace engendré par ces signaux est donc de dimension $N = M$ et les vecteurs $\nu_j(t)$ sont de la forme :

$$\nu_j(t) = \sqrt{\frac{2}{T}} \cos(2\pi f_j t + \varphi_j) \quad j = 1, 2, \cdots M \tag{2.86}$$

En supposant les données d'information d_i iid, les signaux $s_j(t)$ sont équiprobables. Ils ont de plus la même énergie E et ainsi, la règle du maximum de vraisemblance *a posteriori* conduit à la décision suivante :

$$\hat{s}_j(t) \quad \text{si} \quad \sum_{p=1}^{M} r_p s_{jp} > \sum_{p=1}^{M} r_p s_{np} \quad \forall n \neq j \tag{2.87}$$

où s_{jp} est égal à :

$$s_{jp} = \int_0^T s_j(t) \nu_p(t) dt = A \sqrt{\frac{T}{2}} \delta_{jp} \tag{2.88}$$

Compte tenu de l'expression de s_{jp}, la règle de décision (2.87) se simplifie et devient :

$$\hat{s}_j(t) \quad \text{si} \quad r_j > r_n \ \forall n \neq j \tag{2.89}$$

Le récepteur optimal cohérent pour une modulation MDF-M est représenté sur la figure 2.19.

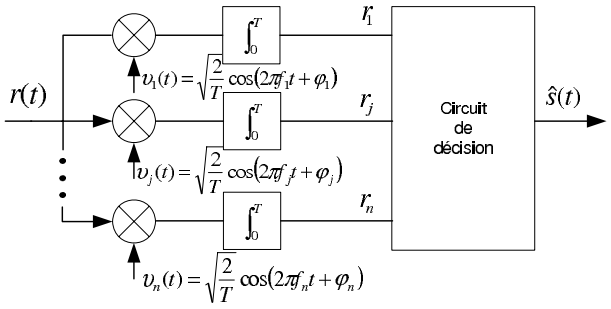


Figure 2.19 – Récepteur cohérent pour modulation MDF-M.

Conditionnellement à l'émission du signal $s_j(t)$, les M sorties du démodulateur sont de la forme :

$$r_j = \sqrt{E_s} + b_j \quad r_p = b_p \quad \forall p \neq j$$

où b_j et b_p sont des BABG, de moyenne nulle et de variance égale à $N_0/2$.
La probabilité de décision correcte sur un groupe de données binaires, conditionnellement à l'émission du signal $s_j(t)$ est égale à :

$$Pc_j = \int_{-\infty}^{+\infty} \Pr\{b_1 < r_j, \cdots, b_p < r_j, \cdots, b_M < r_j\}\, p(r_j) dr_j$$

Les bruits étant non corrélés et donc indépendants puisque gaussiens, nous avons :

$$\Pr\{b_1 < r_j, \cdots, b_p < r_j, \cdots, b_M < r_j\} = \left(\int_{-\infty}^{r_j} \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{b^2}{N_0}\right) db \right)^{M-1}$$

et ainsi la probabilité de décision correcte est égale à :

$$Pc_j = \int_{-\infty}^{+\infty} \left(\int_{-\infty}^{r_j} \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{1}{N_0} b^2\right) db \right)^{M-1} \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{1}{N_0}(r_j - \sqrt{E_s})^2\right) dr_j$$

Après changement de variables, la probabilité de décision correcte peut s'exprimer en fonction du rapport E_s/N_0.

$$Pc_j = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} \left(\int_{-\infty}^{y} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) dx \right)^{M-1} \exp\left(-\frac{1}{2}(y - \sqrt{\frac{E_s}{N_0}})^2\right) dy \qquad (2.90)$$

La probabilité de décision correcte est la même quel que soit le signal émis. Les signaux $s_j(t)$ étant équiprobables, la probabilité de décision correcte moyenne

Pc est donc égale à la probabilité conditionnelle Pc_j. La probabilité d'erreur sur les groupes de données binaires est alors égale à :

$$Pes = 1 - Pc$$

La probabilité d'erreur peut aussi s'exprimer en fonction du rapport E_b/N_0 où E_b est l'énergie utilisée pour transmettre une donnée binaire avec $E_b = E_s/\log_2(M)$.

On peut aussi chercher à déterminer la probabilité d'erreur par donnée binaire Peb. Tous les $M-1$ groupes de données faux apparaissent avec la même probabilité :

$$\frac{Pes}{M-1} \tag{2.91}$$

Dans un groupe de données faux, on peut avoir k données fausses parmi m et ceci peut se produire de $\begin{pmatrix} m \\ k \end{pmatrix}$ façons possibles. Ainsi, le nombre moyen de données fausses dans un groupe est :

$$\sum_{k=1}^{m} k \binom{m}{k} \frac{Pes}{M-1} = m \frac{2^{m-1}}{2^m - 1} Pes$$

et finalement la probabilité d'erreur par donnée binaire est égale :

$$Peb = \frac{2^{m-1}}{2^m - 1} Pes \tag{2.92}$$

où $m = \log_2(M)$.

La probabilité d'erreur pour une modulation MDF-M ne possède pas d'expression simple et il faut faire appel au calcul numérique pour déterminer cette probabilité en fonction du rapport E_b/N_0. On montre que pour une probabilité d'erreur Peb donnée, le rapport E_b/N_0 nécessaire diminue lorsque M augmente. On montre également que la probabilité Pes tend vers une valeur arbitrairement petite lorsque M tend vers l'infini et ceci pour un rapport E_b/N_0 de $4 \ln 2$ soit -1,6 dB.

Pour une transmission binaire ($M = 2$), il existe une expression de la probabilité d'erreur Peb.

Supposons que le signal émis soit $s_1(t)$, nous avons alors :

$$r_1 = \sqrt{E_b} + b_1 \quad r_2 = b_2$$

La décision peut se prendre en comparant $z = r_1 - r_2$ à un seuil fixé à zéro. La probabilité d'erreur Peb_1 conditionnellement à l'émission de $s_1(t)$, est égale à :

$$Peb_1 = \Pr\{z < 0 \,|\, s_1(t)\}$$

En supposant les deux signaux équiprobables, la probabilité d'erreur Peb a pour expression :

$$Peb = \frac{1}{2}(Peb_1 + Peb_2)$$

Les bruits b_1 et b_2 sont gaussiens, non corrélés, de moyenne nulle et de même variance égale à $N_0/2$. La variable z, conditionnellement à l'émission du signal $s_1(t)$, est donc gaussienne, de moyenne $\sqrt{E_b}$ et de variance N_0. Ainsi la probabilité Peb_1 est égale à :

$$Peb_1 = \int_{-\infty}^{0} \frac{1}{\sqrt{2\pi N_0}} \exp\left(-\frac{(z-\sqrt{E_b})^2}{2N_0}\right) dz$$

En introduisant la fonction d'erreur complémentaire, Peb_1 s'écrit :

$$Peb_1 = \frac{1}{2}\text{erfc}\sqrt{\frac{E_b}{2N_0}}$$

Il est facile de vérifier que la probabilité d'erreur conditionnellement à l'émission du signal $s_2(t)$ est identique à la probabilité d'erreur conditionnellement à l'émission du signal $s_1(t)$ et ainsi, on obtient :

$$Peb = \frac{1}{2}\text{erfc}\sqrt{\frac{E_b}{2N_0}} \tag{2.93}$$

Si on compare les performances de la modulation MDF-2 à celles de la MDP-2 on constate que la première nécessite 3 dB de plus de rapport signal à bruit pour obtenir les mêmes performances que la seconde.

Modulation par déplacement de fréquence à phase continue – MDF-M PC

Pour les modulations de fréquence à phase continue, le signal modulé a pour expression :

$$S(t) = A\cos(2\pi f_0 t + \phi(t)) \tag{2.94}$$

où la phase $\phi(t)$, sur l'intervalle $[iT, (i+1)T[$, est égale à :

$$\phi(t) = 2\pi h \sum_{n=i-L+1}^{i} a_n q(t-nT) + \theta_{i-L} \tag{2.95}$$

avec :

$$\theta_{i-L} = \pi h \sum_{n=-\infty}^{i-L} a_n$$

h est l'indice de modulation et les symboles a_i sont M-aires dans le cas général. Ils prennent leurs valeurs dans l'alphabet $\{ \pm 1, \pm 3, \cdots, \pm(2p+1), \cdots, \pm(M -1) \}$; $M = 2^m$.
Si l'indice de modulation $h = m/p$ où m et p sont des entiers relativement premiers, la phase θ_{i-L} prend ses valeurs dans les ensembles suivants :

$$\begin{aligned} \theta_{i-L} &\in \left\{0, \tfrac{\pi m}{p}, \tfrac{2\pi m}{p}, \cdots, \tfrac{(p-1)\pi m}{p}\right\} \quad \text{si} \quad m \text{ pair} \\ \theta_{i-L} &\in \left\{0, \tfrac{\pi m}{p}, \tfrac{2\pi m}{p}, \cdots, \tfrac{(2p-1)\pi m}{p}\right\} \quad \text{si} \quad m \text{ impair} \end{aligned} \tag{2.96}$$

L'évolution de la phase $\phi(t)$ peut être représentée par un treillis dont les états sont définis par $(a_{i-L+1}, a_{i-L+2}, \cdots, a_{i-1}; \theta_{i-L})$ soit :

$$\begin{array}{l} (M^{L-1}p)\,\text{états} \quad \text{si } m \text{ pair} \\ \\ (M^{L-1}2p)\,\text{états} \quad \text{si } m \text{ impair} \end{array} \tag{2.97}$$

À noter que la complexité du treillis augmente très rapidement avec les paramètres M et L. Par exemple, pour une modulation à symboles quaternaires ($M = 4$) à réponse partielle d'indice de modulation $h = 1/2$ et de paramètre $L = 4$, le treillis possède 256 états. Pour la *MSK* et la *GMSK*, les symboles a_i

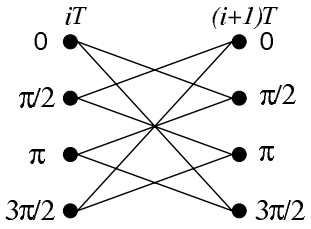


Figure 2.20 – Treillis associé à la phase $\phi(t)$ pour la modulation *MSK*.

sont binaires ($M = 2$) et l'indice de modulation h est $1/2$ soit $m = 1$ et $p = 2$. La phase θ_{i-L} prend donc ses valeurs dans l'ensemble $\{0, \pi/2, \pi, 3\pi/2\}$ et le treillis associé à la phase $\phi(t)$ possède $2^{L-1} \times 4$ états. Sur la figure 2.20 est représenté le treillis associé à la phase $\phi(t)$ pour la modulation *MSK*.
Pour décoder les symboles a_i on utilise l'algorithme de Viterbi dont le principe est rappelé ci-après. Pour chaque intervalle de temps $[iT, (i+1)T[$, procéder de la façon suivante :

- pour chaque branche l quittant un état du treillis à l'instant iT calcul de la métrique z_i^l telle que définie plus tard soit, pour la *MSK* et la *GMSK*, $2^L \times 4$ métriques à calculer ;
- pour chaque chemin convergeant à l'instant $(i+1)T$ vers un état du treillis calcul de la métrique cumulée, puis sélection du chemin ayant la métrique cumulée la plus grande, appelé chemin survivant ;
- parmi les chemins survivants, remontée sur s branches du chemin ayant la métrique cumulée la plus grande et décodage du symbole a_{i-s} ;
- poursuite de l'algorithme sur l'intervalle de temps suivant.

La métrique z_i^l de branche a pour expression :

$$z_i^l = \int_{iT}^{(i+1)T} r(t)\cos(2\pi f_0 t + \phi_i^l(t) + \varphi_0)dt$$

où $r(t) = s(t) + b(t)$ est le signal reçu par le récepteur et $b(t)$ est un BABG, de moyenne nulle et de densité spectrale de puissance égale à $N_0/2$. La quantité

$\phi_i^l(t)$ représente une réalisation de la phase $\phi(t)$ associée à la branche l du treillis sur l'intervalle de temps $[iT,\ (i+1)T[$.
En tenant compte du fait que le bruit peut se mettre sous la forme $b(t) = b_c(t)\cos(2\pi f_0 t + \varphi_0) - b_s(t)\sin(2\pi f_0 t + \varphi_0)$ et que $f_0 >> 1/T$, la métrique de branche peut encore s'écrire :

$$z_i^l = \int_{1T}^{(i+1)T} r_c(t)\cos\phi_l(t)dt + \int_{iT}^{(i+1)T} r_s(t)\sin\phi_l(t)dt \tag{2.98}$$

où les signaux $r_c(t)$ et $r_s(t)$ sont obtenus après transposition en bande de base de $r(t)$(multiplication de $r(t)$ respectivement par $\cos(2\pi f_0 t + \varphi_0)$ et $-\sin(2\pi f_0 t + \varphi_0)$ puis filtrage passe-bas).

$$\begin{aligned}\cos\phi_i^l(t) &= \cos\left(2\pi h \sum_{n=i-L+1}^{i} a_n^l q(t-nT) + \theta_{i-L}^l\right)\\ \sin\phi_i^l(t) &= \sin\left(2\pi h \sum_{n=i-L+1}^{i} a_n^l q(t-nT) + \theta_{i-L}^l\right)\end{aligned} \tag{2.99}$$

En posant :

$$\psi_i^l(t) = 2\pi h \sum_{n=i-L+1}^{i} a_n^l q(t-nT)$$

la métrique de branche z_i^l peut encore s'écrire sous la forme :

$$z_i^l = \cos\theta_{i-L}^l A_l + \sin\theta_{i-L}^l B_l \tag{2.100}$$

avec :

$$\begin{aligned}A_i^l &= \int_{iT}^{(i+1)T} (r_c(t)\cos\psi_i^l(t) + r_s(t)\sin\psi_i^l(t))dt\\ B_i^l &= \int_{iT}^{(i+1)T} (r_s(t)\cos\psi_i^l(t) - r_c(t)\sin\psi_i^l(t))dt\end{aligned}$$

Pour la modulation *MSK*, il est possible de décoder les symboles a_i en utilisant un récepteur similaire à celui de la modulation MDP-4. En effet, le signal *MSK* peut se mettre sous la forme suivante :

$$\begin{aligned}S(t) = A\Big[&\sum_i c_{2i-1} h(t-2iT)\cos\tfrac{\pi t}{2T}\cos(2\pi f_0 t + \varphi_0)\\ &- \sum_i c_{2i} h(t-(2i+1)T)\sin\tfrac{\pi t}{2T}\sin(2\pi f_0 t + \varphi_0)\Big]\end{aligned} \tag{2.101}$$

où les symboles c_i se déduisent des symboles a_i par un codage par transition.

$$c_{2i} = a_{2i}c_{2i-1} \quad \text{et} \quad c_{2i-1} = a_{2i-1}c_{2i-2} \tag{2.102}$$

et $h(t)$ est une porte d'amplitude unité de durée $2T$:

$$\begin{aligned} h(t) &= 1 \text{ si } t \in [-T, T[\\ &= 0 \text{ ailleurs} \end{aligned}$$

Le récepteur cohérent pour la *MSK* est représenté dans la figure 2.21. Il comprend deux filtres adaptés à $h(t)$ de réponse impulsionnelle $h(2T - t)$. Les symboles c_{2i-1} et c_{2i} sont décodés en comparant des échantillons prélevés en sortie des filtres adaptés respectivement aux instants $2iT$ et $(2i + 1)T$.

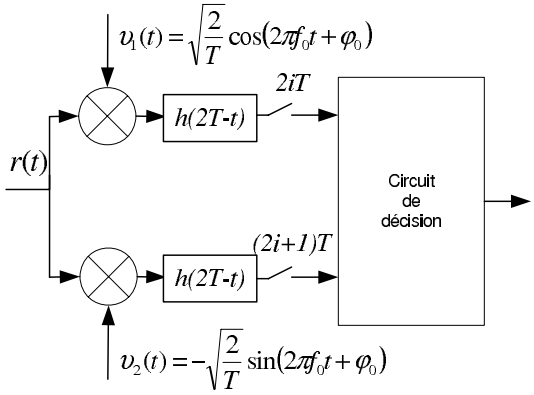


Figure 2.21 – Récepteur cohérent pour modulation *MSK*.

Il est facile de montrer que les probabilités d'erreur sur les symboles binaires c_{2i-1} et c_{2i} sont identiques et égaux à :

$$Pe_{c_i} = \frac{1}{2}\text{erfc}\sqrt{\frac{E_b}{N_0}} \tag{2.103}$$

où E_b est l'énergie utilisée pour transmettre un symbole c_i.
Pour obtenir les données binaires a_i à partir des symboles c_i il faut utiliser en sortie du récepteur cohérent un décodeur différentiel régi par les équations suivantes :

$$a_{2i} = c_{2i}c_{2i-1} \quad \text{et} \quad a_{2i-1} = c_{2i-1}c_{2I-2}$$

La probabilité d'erreur Peb sur a_i est :

$$Peb = 1 - (1 - Pe_{c_i})^2$$

ainsi pour $Pe_{c_i} << 1$, la probabilité d'erreur Peb est bien approchée par :

$$Peb \approx 2Pe_{c_i} \tag{2.104}$$

En première approximation, les performances de la modulation *MSK* sont identiques à celles de la modulation MDP-4.

2.3 Transmission sur canal à bande limitée

2.3.1 Introduction

Jusqu'à présent, dans ce chapitre, il était supposé que la bande de fréquences allouée à la transmission était infinie. Nous allons maintenant nous placer dans une situation plus réaliste où une bande de fréquences de largeur W est disponible pour transmettre le signal modulé. Dans cette bande W le canal est supposé avoir une réponse plate et une phase linéaire. Nous nous limiterons au cas des modulations linéaires de type MDA-M, MDP-M et MAQ-M qui possèdent une densité spectrale de puissance constituée d'un lobe principal de largeur $2/T$, où $1/T$ est la rapidité de modulation, avec des lobes secondaires qui s'annulent périodiquement en n/T. La bande occupée par un signal modulé linéairement est donc, en toute rigueur, infinie. Le signal modulé doit en conséquence être filtré par un filtre d'émission avant d'être placé à l'entrée du canal de transmission. Nous allons maintenant déterminer quelle est la bande W minimale nécessaire pour transmettre le signal modulé sans dégradation des performances par rapport à une transmission à bande infinie. La réponse en fréquences des filtres d'émission et de réception sera également établie.

Dans ce qui suit nous raisonnerons sur l'enveloppe complexe du signal modulé et sur la réponse équivalente en bande de base du filtre d'émission. cela évite, sans nuire à la généralité du propos, d'introduire la fréquence porteuse qui complique les notations.

L'enveloppe complexe d'un signal MDA-M, MDP-M et MAQ-M a pour expression :

$$s_e(t) = A \sum_i c_i h(t - iT) \tag{2.105}$$

où $h(t)$ est une porte d'amplitude unité et de durée T et $c_i = a_i + jb_i$ est un symbole de modulation avec :

$$\begin{array}{lll} \text{MDA-M} & c_i = a_i & b_i = 0 \\ \text{MDP-M} & a_i = \cos\phi_i & b_i = \sin\phi_i \\ \text{MAQ-M} & a_i \text{ et } b_i & \text{symboles} \sqrt{M} - \text{aires} \end{array}$$

Soit $g(t)$ la réponse impulsionnelle du filtre passe-bande d'émission centré sur la fréquence porteuse. Cette réponse peut s'écrire sous la forme :

$$g(t) = g_c(t)\cos(2\pi f_0 t + \theta_0) - g_s(t)\sin(2\pi f_0 t + \theta_0) \tag{2.106}$$

ou de manière équivalente :

$$g(t) = \Re_e \left\{ g_e(t) \exp\left[j(2\pi f_0 t + \theta_0) \right] \right\} \tag{2.107}$$

où $g_e(t) = g_c(t) + jg_s(t)$ est la réponse équivalente en bande de base du filtre d'émission. La sortie $e(t)$ du filtre d'émission est égale à :

$$e(t) = A \sum_i c_i z(t - iT) \tag{2.108}$$

où $z(t) = h(t) * g_e(t)$ est, dans le cas général, une forme d'onde complexe alors que $h(t)$ est réel.

2.3.2 L'interférence entre symboles

Après passage dans le filtre d'émission, le signal modulé occupe une bande W et ainsi, le signal en sortie du canal de transmission a pour expression :

$$r(t) = e(t) + b(t) \tag{2.109}$$

où $b(t)$ est un BABG complexe, de moyenne nulle et de densité spectrale de puissance égale à $2N_0$.

Le récepteur cohérent utilise un filtre de réception suivi d'un échantillonneur à l'instant $t_0 + nT$, où t_0 est un instant qui peut être choisi arbitrairement. La sortie du filtre de réception de réponse impulsionnelle $g_r(t)$ a pour expression :

$$y(t) = A \sum_i c_i x(t - iT) + b'(t) \tag{2.110}$$

où :

$$\begin{aligned} x(t) &= z(t) * g_r(t) \\ b'(t) &= b(t) * g_r(t) \end{aligned}$$

En échantillonnant le signal $y(t)$ à l'instant $t_0 + nT$, on obtient :

$$y(t_0 + nT) = A \sum_i c_i x(t_0 + (n - i)T) + b'(t_0 + nT) \tag{2.111}$$

En considérant que dans le cas général $x(t) = p(t) + jq(t)$ est une forme d'onde complexe, l'échantillon $y(t_0 + nT)$ peut encore s'écrire sous la forme :

$$\begin{aligned} y(t_0 + nT) = {} & Ac_n p(t_0) + A \sum_{i \neq 0} c_{n-i} p(t_0 + iT) \\ & + jA \sum_i c_{n-i} q(t_0 + iT) + b'(t_0 + nT) \end{aligned} \tag{2.112}$$

Le premier terme $Ac_n p(t_0)$ représente l'information souhaitée pour le décodage du symbole c_n, les deux termes suivants sont des termes d'Interférence Entre Symboles (IES). Examinons les sorties des deux voies du récepteur, dite voies en phase et en quadrature, correspondant respectivement à la partie réelle et à la partie imaginaire de $y(t_0 + nT)$. On peut remarquer que la voie en phase (respectivement la voie en quadrature) dépend des symboles a_i (respectivement des symboles b_i) mais aussi des symboles b_i (respectivement des symboles a_i).

On dit parfois qu'il existe de la diaphonie entre les deux voies du récepteur. Bien entendu l'IES est un phénomène qui ne peut que dégrader la qualité de la transmission. C'est la raison pour laquelle il est important de définir la condition à satisfaire pour annuler l'IES. Mais avant cela, nous allons indiquer une manière simple de caractériser l'IES en traçant le diagramme de l'œil, appelé ainsi par analogie avec la forme de l'œil humain, en sortie du filtre de réception des voies en phase et en quadrature du récepteur.
Le diagramme de l'œil est la figure obtenue en superposant toutes les traces ou réalisations du signal $y_c(t)$ non bruité où $y_c(t)$ est la partie réelle de $y(t)$. On obtient également le diagramme de l'œil à partir de $y_s(t)$ non bruité où $y_s(t)$ est la partie imaginaire de $y(t)$.

$$\begin{aligned} y_c(t) &= A\sum_i a_i p(t-iT) - A\sum_i b_i q(t-iT) \\ y_s(t) &= A\sum_i b_i p(t-iT) + A\sum_i a_i q(t-iT) \end{aligned} \tag{2.113}$$

Analysons, par exemple, la sortie $y_c(t)$ du filtre de réception de la voie en phase sur l'intervalle de temps $[t_1 + nT,\ t_1 + (n+1)T[$ où t_1 représente un instant arbitraire. En remplaçant t par $t + t_1 + nT$, le signal $y_c(t)$ peut s'écrire :

$$\begin{aligned} y_c(t+t_1+nT) = A\sum_i a_{n-i} p(t+t_1+iT) \\ -A\sum_i b_{n-i} q(t+t_1+iT) \quad \text{pour} \quad 0 \leqslant t \leqslant T \end{aligned} \tag{2.114}$$

En supposant $x(t + t_1 + iT)$ négligeable en dehors de l'intervalle $[t_1 - L_1T,\ t_1 + L_2T]$, chaque somme de l'expression précédente comporte $(L_1 + L_2 + 1)$ termes soit $4^{(L_1+L_2+1)}$ traces possibles pour le diagramme de l'œil. Décalons l'intervalle d'observation d'une quantité T, le nombre de traces qui constituent le diagramme de l'œil est toujours de $4^{(L_1+L_2+1)}$, le diagramme de l'œil est donc une figure qui se répète toutes les T secondes. Son analyse peut, en conséquence, être limitée à un intervalle de durée T.

Le diagramme de l'œil peut être visualisé sur un oscilloscope. En effet, les différentes traces de $p(t)$ et $q(t)$ peuvent être conservées à l'écran si la vitesse de balayage de l'oscilloscope est suffisamment grande devant la durée de rémanence du tube cathodique ou, mieux, si l'oscilloscope est à mémoire. Nous avons représenté le diagramme de l'œil d'une modulation MDP-2 avec et sans IES sur la figure 2.22. Pour cette modulation, les symboles c_i sont réels et le récepteur comprend une seule voie.

$$y_c(t+t_1+nT) = A\sum_i a_{n-i} p(t+t_1+iT) \text{ pour } 0 \leqslant t \leqslant T \tag{2.115}$$

En l'absence d'IES, à l'instant de décision, toutes les traces de $p(t)$ passent par un point unique. Plus le diagramme de l'œil est ouvert à l'instant de décision et plus l'immunité de la transmission au bruit est grande. De la même façon,

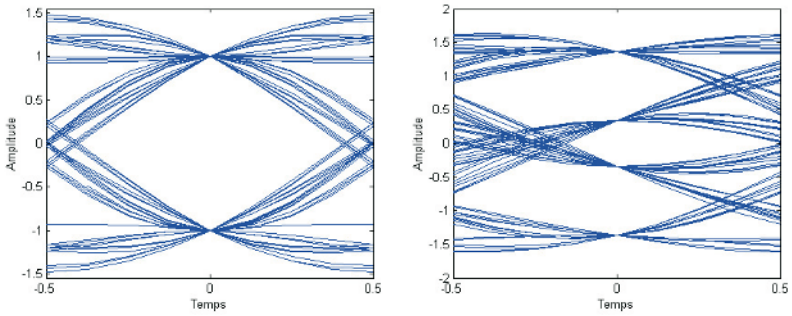


Figure 2.22 – Diagramme de l'œil d'une modulation MDP-2 avec (a) et sans (b) IES.

plus l'ouverture horizontale du diagramme de l'œil est importante et moins la transmission est sensible aux erreurs de positionnement de l'instant de décision. En présence d'IES, les différentes traces de $p(t)$ ne passent plus par un point unique à l'instant de décision et l'IES contribue à fermer le diagramme de l'œil.

La sortie du filtre de réception aux instants $t_0 + nT$ de la voie en phase du récepteur est égale à :

$$y_c(t_0 + nT) = Aa_n p(t_0) + A \sum_{i \neq 0} a_{n-i} p(t_0 + iT) - A \sum_i b_{n-i} q(t_0 + iT) \tag{2.116}$$

Pour une modulation MAQ-M, le signal utile vaut $A(2j - 1 - \sqrt{M})p(t_0)$, pour $j = 1, \cdots \sqrt{M}$ et la décision est prise en comparant le signal $y_c(t_0 + nT)$ à un ensemble de seuils séparés de $2p(t_0)$. Il y aura des erreurs en l'absence de bruit si l'IES, pour certaines configurations des symboles c_i, est telle que le signal reçu se trouve en dehors de la zone de décision correcte. Ceci se produit si l'IES est supérieure en valeur absolue à $p(t_0)$. Cette situation se traduit par la condition suivante :

$$\text{Max}_{a_{n-i}, b_{n-i}} \left| \sum_{i \neq 0} a_{n-i} p(t_0 + iT) - \sum_i b_{n-i} q(t_0 + iT) \right| > |p(t_0)|$$

En tenant compte du fait que la plus grande valeur prise par les symboles a_i et b_i est $\sqrt{M} - 1$, la condition précédente devient :

$$D_{\max} = (\sqrt{M} - 1) \frac{\left(\sum_{i \neq 0} |p(t_0 + iT)| + \sum_i |q(t_0 + iT)| \right)}{|p(t_0)|} \geqslant 1 \tag{2.117}$$

La quantité $D_{\max}$ est appelée distorsion maximale. Lorsque la distorsion maximale est supérieure à l'unité, le diagramme de l'œil est fermé aux instants de décision et des erreurs sont possibles, même en l'absence de bruit.

2.3.3 Condition d'absence d'IES : critère de Nyquist

L'absence d'IES se traduit par les conditions suivantes :

$$p(t_0 + iT) = 0 \quad \forall\, i \neq 0 \tag{2.118}$$

$$q(t_0 + iT) = 0 \quad \forall\, i \tag{2.119}$$

ce qui peut encore s'écrire en utilisant le signal complexe $x(t) = p(t) + jq(t)$.

$$x(t_0 + iT) = p(t_0)\delta_{0,i} \quad \forall\, i \tag{2.120}$$

où $\delta_{0,i}$ est le symbole de Kronecker.

Introduisons le signal échantillonné $x_E(t)$, défini par :

$$x_E(t) = x(t) \sum_i \delta(t - t_0 - iT) \tag{2.121}$$

On peut remarquer que le peigne de Dirac $u(t) = \sum_i \delta(t - t_0 - iT)$ est périodique, de période T. Il peut donc être décomposé en série de Fourier :

$$u(t) = \frac{1}{T} \sum_i \exp\left(-j2\pi \frac{i}{T} t_0\right) \exp\left(-j2\pi \frac{i}{T} t\right) \tag{2.122}$$

Dans la mesure où l'on cherche à déterminer la bande de fréquences W minimale nécessaire à la transmission sans IES du signal modulé, il est judicieux de travailler dans l'espace des fréquences. En prenant la transformée de Fourier notée $X_E(f)$ de la relation (2.121) et en tenant compte de l'expression précédente de $u(t)$, on obtient :

$$X_E(f) = \frac{1}{T} \sum_i \exp\left(-j2\pi \frac{i}{T} t_0\right) X(f - \frac{i}{T}) \tag{2.123}$$

Le signal échantillonné, d'après la relation (2.121), peut encore s'écrire :

$$x_E(t) = \sum_i x(t_0 + iT)\delta(t - t_0 - iT) \tag{2.124}$$

qui, après transformée de Fourier et en tenant compte de la condition d'absence d'IES, devient :

$$X_E(f) = p(t_0) \exp(-j2\pi f t_0) \tag{2.125}$$

L'égalité des relations (2.123) et (2.125) donne :

$$\sum_i \exp\left(j2\pi(f - \frac{i}{T})t_0\right) X(f - \frac{i}{T}) = Tp(t_0)$$

En posant :

$$X^{t_0}(f) = \frac{X(f)}{p(t_0)} \exp(j2\pi f t_0)$$

la condition d'absence d'IES peut s'exprimer à partir de $X^{t_0}(f)$ par la relation suivante :

$$\sum_{i=-\infty}^{+\infty} X^{t_0}(f - \frac{i}{T}) = T \tag{2.126}$$

Cette condition d'absence d'IES est appelée *critère de Nyquist*.

Rappelons que le canal de transmission de réponse en fréquences $C(f)$ possède une bande passante W.

$$C(f) = 0 \quad \text{pour} \quad |f| > W \tag{2.127}$$

Considérons la relation (2.126) pour trois situations.

1. $X^{t_0}(f)$ occupe une bande de fréquence $W < 1/2T$. La relation (2.126) étant une somme de fonctions décalées de $1/T$, il n'existe pas de fonctions $X^{t_0}(f)$ qui permettent de vérifier le critère de Nyquist. La bande de fréquences W nécessaire à une transmission sans IES est donc supérieure ou égale à $1/2T$.
2. $X^{t_0}(f)$ occupe une bande de fréquences $W = 1/2T$. Il existe une solution unique pour satisfaire le critère de Nyquist :

$$\begin{array}{rcll} X^{t_0}(f) & = & T & |f| \leqslant W \\ & = & 0 & \text{ailleurs} \end{array} \tag{2.128}$$

   soit encore :

$$\begin{array}{rcll} X(f) & = & Tp(t_0)\exp(-j2\pi t_0) & |f| \leqslant W \\ & = & 0 & \text{ailleurs} \end{array} \tag{2.129}$$

   ce qui, sur le plan temporel, donne :

$$x(t) = p(t_0)\frac{\sin\left[\pi(t - t_0)/T\right]}{\pi(t - t_0)/T} \tag{2.130}$$

   Cette solution correspond à une réponse $x(t)$ en toute rigueur non causale. Toutefois, dans la mesure où la fonction $\sin y/y$ décroît assez rapidement en fonction de son argument y, il est possible, en choisissant t_0 suffisant grand, de considérer $x(t)$ comme pratiquement causale. Avec cette solution, le diagramme de l'œil a une ouverture horizontale qui tend vers zéro et ainsi, toute imprécision sur l'instant d'échantillonnage peut conduire à des erreurs même en l'absence de bruit. En conclusion, cette solution est purement théorique et n'a donc pas d'application pratique.
3. $X^{t_0}(f)$ occupe une bande de fréquences $W > 1/2T$. Dans ce cas, il existe de nombreuses solutions qui permettent de vérifier le critère de Nyquist. Parmi ces solutions, la plus utilisée est la fonction en cosinus surélevé définie par :

$$\begin{array}{lll} X^{t_0}(f) = T & \text{si} & 0 \leqslant |f| \leqslant \frac{1-\alpha}{2T} \\ \frac{T}{2}\left[1+\sin\left(\frac{\pi T}{\alpha}\left(\frac{1}{2T}-|f|\right)\right)\right] & \text{si} & \frac{1-\alpha}{2T} \leqslant |f| \leqslant \frac{1+\alpha}{2T} \\ 0 & \text{si} & |f| > \frac{1+\alpha}{2T} \end{array} \tag{2.131}$$

soit encore :

$$X(f) = p(t_0) X^{t_0}(f) \exp(-j2\pi f t_0) \tag{2.132}$$

dont la réponse temporelle est :

$$x(t) = p(t_0) \frac{\sin \frac{\pi(t-t_0)}{T}}{\frac{\pi(t-t_0)}{T}} \frac{\cos \frac{\pi\alpha(t-t_0)}{T}}{1-4\alpha^2\frac{(t-t_0)^2}{T^2}} \tag{2.133}$$

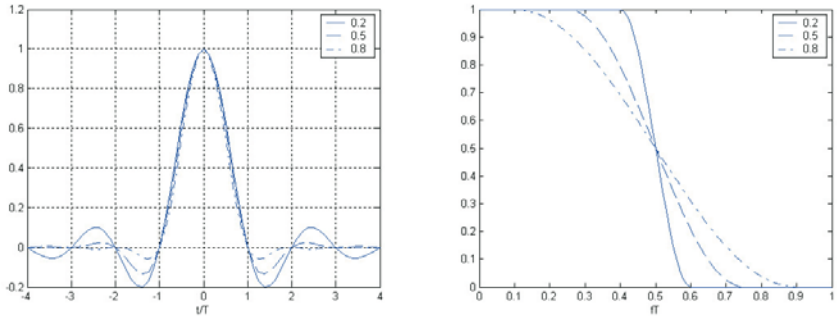


Figure 2.23 – Réponses fréquentielle et temporelle d'une fonction en cosinus surélevé pour différentes valeurs du facteur de retombée α.

La figure 2.23 représente les réponses fréquentielle $X^{t_0}(f)$ et temporelle $x(t)$ d'une fonction en cosinus surélevé pour différentes valeurs de α, appelé facteur de retombée (*roll-off* en anglais).

La bande de fréquences occupée avec la fonction en cosinus surélevé est $W = (1+\alpha)/2T$; $0 \leqslant \alpha \leqslant 1$. La fonction $x(t)$ est encore non causale au sens strict mais sa décroissance est d'autant plus forte que le facteur de retombée augmente. Ainsi, en choisissant t_0 suffisamment élevé la mise en œuvre d'un cosinus surélevé est possible. Sur la figure 2.24 sont tracés des diagrammes de l'œil obtenus avec des fonctions en cosinus surélevé pour différentes valeurs du facteur de retombée.

Toutes les traces de $x(t)$ passent par un point unique à l'instant d'échantillonnage $t_0 + nT$ et ceci quelle que la valeur du facteur de retombée. Notons que l'ouverture horizontale du diagramme de l'œil est d'autant plus importante que le facteur de retombée est grand. Pour $\alpha = 1$, l'ouverture de l'œil est maximale et égale à T ; la sensibilité à une imprécision sur l'instant d'échantillonnage est alors minimale.

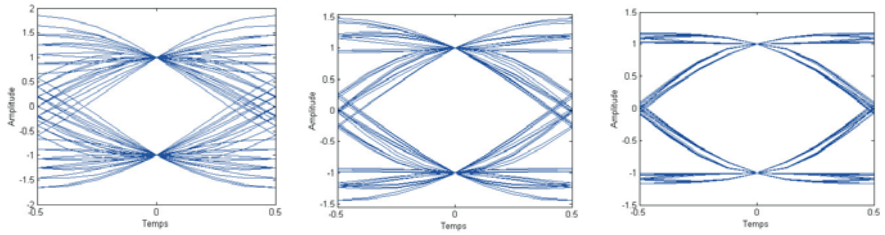


Figure 2.24 – Diagrammes de l'œil pour des modulations à symboles binaires (MDP-2 ou MDP-4) pour différentes valeurs du facteur de retombée α $(0.2, 0.5, 0.8)$.

Nous venons de déterminer la réponse globale $X(f)$ que doit satisfaire la chaîne de transmission pour garantir l'absence d'IES. Les réponses en fréquences des filtres d'émission et de réception vont maintenant être établies.

Répartition optimale du filtrage entre l'émission et la réception

Nous avons vu que le filtre de réception doit être adapté la forme d'onde placée à son entrée soit dans notre cas :

$$g_r(t) = z(t_0 - t) \tag{2.134}$$

où $z(t)$ résulte du filtrage de $h(t)$ par le filtre d'émission :

$$z(t) = h(t) * g_e(t)$$

la réponse en fréquences $G_r(f)$ du filtre de réception est égale à :

$$G_r(f) = Z^*(f) \exp(-j2\pi f t_0) \tag{2.135}$$

où $*$ désigne la conjugaison.

Bien entendu la réponse globale de la chaîne de transmission doit vérifier le critère de Nyquist, ce qui se traduit par :

$$Z(f)G_r(f) = p(t_0)CS_\alpha(f) \exp(-j2\pi f t_0) \tag{2.136}$$

où $CS_\alpha(f) = X^{t_0}(f)$ est la réponse en cosinus surélevé de facteur de retombée α. Dans la relation précédente nous avons considéré que le canal transmet intégralement le signal placé à son entrée.
En exprimant la fonction $Z^*(f)$ à partir de la relation précédente,

$$Z^*(f) = \frac{p(t_0)CS_\alpha(f)}{G_r^*(f)} \exp(j2\pi f t_0) \tag{2.137}$$

puis en remplaçant $Z^*(f)$ par son expression dans la relation (2.135), on obtient le module de la réponse en fréquences du filtre de réception :

$$|G_r(f)| = \sqrt{p(t_0)}\sqrt{CS_\alpha(f)} \tag{2.138}$$

Le module de la réponse en fréquences du filtre d'émission s'obtient sans difficulté à partir de :

$$|G_e(f)| = \frac{|Z(f)|}{|H(f)|} \tag{2.139}$$

En utilisant la relation (2.137), on peut déterminer le module de $|Z(f)|$.

$$|Z(f)| = \sqrt{p(t_0)}\sqrt{CS_\alpha(f)} \tag{2.140}$$

En remplaçant $|Z(f)|$ par sa valeur dans la relation (2.139), on obtient le module de la réponse du filtre de réception :

$$|G_e(f)| = \frac{\sqrt{p(t_0)}\sqrt{CS_\alpha(f)}}{|H(f)|} \tag{2.141}$$

Nous avons obtenu le module des réponses en fréquences des filtres d'émission et de réception. Ces filtres sont donc définis à une phase arbitraire près. En répartissant le retard t_0 entre les filtres d'émission et de réception, on obtient :

$$\begin{aligned} G_r(f) &= \sqrt{p(t_0)}\sqrt{CS_\alpha(f)}\exp(-j2\pi f t_1) \\ G_e(f) &= \frac{\sqrt{p(t_0)}\sqrt{CS_\alpha(f)}}{H(f)}\exp(-j2\pi f(t_0 - t_1)) \end{aligned} \tag{2.142}$$

où t_1 est un retard inférieur à t_0 qui peut être choisi arbitrairement.

Les réponses des filtres $G_e(f)$ et $G_r(f)$ montrent que le filtrage de Nyquist doit être équiréparti entre l'émission et la réception. Le terme $H(f)$ qui apparaît au dénominateur du filtre d'émission entraîne que la sortie de ce filtre ne dépend plus de la forme d'onde utilisée par le modulateur. La réponse $H(f)$, transformée de Fourier d'une porte de durée T, présente des zéros aux fréquences n/T ce qui, en toute rigueur, rend irréalisable le filtre de réponse $G_e(f)$. Toutefois, le filtre $G_e(f)$ doit être réalisé uniquement dans la bande $[-(1+\alpha)/2T;\ (1+\alpha)/2T]$ où $H(f)$ ne présente pas de zéros.
Pour terminer, déterminons la dsp du signal modulé en sortie du filtre d'émission et vérifions qu'il occupe bien une bande de fréquences $W = (1+\alpha)/2T$.

La dsp du signal modulé en bande de base à l'entrée du filtre d'émission est égale à :

$$\gamma_{s_e}(f) = 2\frac{(M-1)}{3}A^2T\left(\frac{\sin \pi fT}{\pi fT}\right)^2 \tag{2.143}$$

La dsp du signal modulé en bande de base en sortie du filtre d'émission a pour expression :

$$\gamma_e(f) = \gamma_{s_e}(f)\,|G_e(f)|^2$$

En remplaçant $|G_e(f)|$ par son expression, on obtient :

$$\gamma_e(f) = 2\frac{(M-1)}{3}\frac{A^2}{T}p(t_0)CS_\alpha(f) \tag{2.144}$$

En considérant le signal modulé sur fréquence porteuse, sa dsp est donnée par l'expression (2.13) :

$$\gamma(f) = \frac{1}{4}\gamma_e(f - f_0) + \frac{1}{4}\gamma_e(f + f_0)$$

En conclusion, la bande de fréquences occupée par le signal modulé sur fréquence porteuse s'étend de $f_0 - (1+\alpha)/2T$ à $f_0 + (1+\alpha)/2T$, soit une bande de largeur $2W = (1+\alpha)/T$ ou encore $(1+\alpha)R_m$ où R_m est la rapidité de modulation. L'efficacité spectrale de la modulation MAQ-M exprimée en bit/s/Hz est alors égale à :

$$\eta = \frac{D}{2W} = \frac{R_m \log_2(M)}{(1+\alpha)R_m} = \frac{\log_2(M)}{(1+\alpha)} \tag{2.145}$$

où D est le débit de la transmission en bit/s.

L'efficacité spectrale augmente en fonction du nombre d'états M de la modulation mais les performances de la modulation, en terme de probabilité d'erreur, se dégradent en fonction de ce paramètre M.

2.3.4 Expression de la probabilité d'erreur en présence de filtrage de Nyquist

Déterminons la probabilité d'erreur sur les données binaires délivrées par la source en considérant, par exemple, une modulation MDP-4. Dans ce cas, chaque symbole a_i (respectivement b_i) transmet une donnée binaire d_i toutes les T secondes. La probabilité d'erreur sur les données d_i est donc identique à la probabilité d'erreur sur les symboles a_i ou b_i. Calculons, par exemple, la probabilité d'erreur sur le symbole a_i.

La sortie du filtre de réception de la voie en phase à l'instant de décision $t_0 + nT$ est égale à :

$$y_c(t_0 + nT) = Aa_n p(t_0) + b'_c(t_0 + nT) \tag{2.146}$$

où $b'_c(t_0 + nT)$ est la partie réelle du bruit $b'(t_0 + nT)$.

En supposant les données d_n iid, la probabilité d'erreur sur les symboles a_n a pour expression :

$$\begin{aligned} Pe_{a_n} = \tfrac{1}{2}\Pr\{y_c(t_0 + nT) > 0 | a_n = -1\} \\ \tfrac{1}{2}\Pr\{y_c(t_0 + nT) < 0 | a_n = +1\} \end{aligned} \tag{2.147}$$

Puisque $y_c(t_0 + nT)$ est gaussien, de moyenne $Aa_n p(t_0)$ et de variance $\sigma^2_{b'_c}$, la probabilité d'erreur Pe_{a_n} est égale à :

$$Pe_{a_n} = \frac{1}{2}\mathrm{erfc}\frac{Ap(t_0)}{\sigma_{b'_c}\sqrt{2}} \tag{2.148}$$

La variance $\sigma^2_{b'_c}$ du bruit $b'_c(t_0 + nT)$ est égale à :

$$\sigma^2_{b'_c} = N_0 \int_{-\infty}^{+\infty} |G_r(f)|^2 df = N_0 \int_{-\infty}^{+\infty} p(t_0) C S_\alpha(f) df = N_0 p(t_0) \tag{2.149}$$

Introduisons la puissance émise en sortie du filtre d'émission :

$$P = \frac{1}{4} \int_{-\infty}^{+\infty} \gamma_e(f - f_0) df + \frac{1}{4} \int_{-\infty}^{+\infty} \gamma_e(f + f_0) df = \frac{1}{2} \int_{-\infty}^{+\infty} \gamma_e(f) df \tag{2.150}$$

En remplaçant $\gamma_e(f)$ par son expression, on obtient :

$$P = \int_{-\infty}^{+\infty} \frac{A^2}{T} p(t_0) C S_\alpha(f) df = \frac{A^2}{T} p(t_0) \tag{2.151}$$

En utilisant les expressions (2.149) et (2.151), la probabilité d'erreur est égale à :

$$Pe_{a_n} = \frac{1}{2} \text{erfc} \sqrt{\frac{PT}{2N_0}} \tag{2.152}$$

L'énergie E_b utilisée pour transmette une donnée d'information d_n est :

$$E_b = PT_b \tag{2.153}$$

où T_b est l'inverse du débit binaire de la transmission.

Pour une modulation MDP-4, $T = 2T_b$ et la probabilité d'erreur par donnée binaire d_n est finalement :

$$Pe_{d_n} = \frac{1}{2} \text{erfc} \sqrt{\frac{E_b}{N_0}} \tag{2.154}$$

La probabilité d'erreur en présence de filtrage de Nyquist pour une modulation MDP-4 est identique à celle obtenue pour une transmission sur un canal à bande infinie. Ce résultat est aussi vrai pour les autres modulations linéaires de type MDA-M, MDP-M et MAQ-M.

En conclusion de cette section, on peut dire que le filtrage selon le critère de Nyquist d'une modulation linéaire permet de réduire la bande nécessaire à sa transmission à $(1 + \alpha)R_m$, où R_m est la rapidité de modulation. Ce filtrage ne dégrade pas les performances de la modulation, c'est-à-dire qu'il conduit à la même probabilité d'erreur par donnée binaire que celle d'une transmission sur canal à bande infinie.

2.4 Transmission sur canal à évanouissements

2.4.1 Caractérisation d'un canal à évanouissements

Considérons une transmission sur canal à trajets multiples où l'émetteur, mobile par rapport au récepteur, délivre un signal $s(t) = A\exp(j2\pi ft)$ non modulé d'amplitude A et de fréquence$f$. Le signal $s(t)$ se propage en se réfléchissant sur différents obstacles et ainsi, le récepteur reçoit M répliques du signal émis, chaque réplique étant affectée d'une atténuation $\rho_n(t)$, d'un retard $\tau_n(t)$ et d'un décalage de fréquence Doppler $f_n^d(t)$. Les atténuations, les retards et les fréquences Doppler sont fonctions du temps pour tenir compte de l'évolution temporelle du canal. Pour simplifier les notations, on omettra dans la suite, la variable t pour les atténuations, les retards et les fréquences Doppler.

Soit $r(t)$ la réponse du canal de transmission au signal $s(t)$, qui s'écrit de manière générale sous la forme :

$$r(t) = \sum_{n=1}^{M} \rho_n \exp\left[j2\pi(f_n^d + f)(t - \tau_n)\right] \tag{2.155}$$

En faisant apparaître $s(t)$, le signal reçu peut encore s'écrire :

$$r(t) = \sum_{n=1}^{M} \rho_n \exp\left[j2\pi(f_n^d t - (f_n^d + f)\tau_n))\right] s(t) \tag{2.156}$$

et ainsi la réponse en fréquence du canal de transmission est définie par :

$$c(f,t) = \sum_{n=1}^{M} \rho_n \exp\left[-j2\pi(f\tau_n - f_n^d t + f_n^d \tau_n)\right] \tag{2.157}$$

La réponse du canal à trajets multiples est généralement sélective en fréquences c'est-à-dire qu'elle ne transmet pas de la même façon toutes les composantes fréquentielles du signal placé à son entrée, certaines composantes étant plus atténuées que d'autres. Le canal va donc provoquer des distorsions du signal transmis. De plus, leur évolution au cours du temps peut être plus ou moins rapide.

Pour illustrer la sélectivité en fréquences d'un canal à trajets multiples, nous avons tracé sur la figure 2.25 le carré du module de la réponse en fréquences de ce canal pour $M = 2$, en l'absence de décalge de fréquence Doppler ($f_n^d = 0$) et en fixant τ_1 à zéro.

$$|c(f)|^2 = \rho_1^2\left[(1 + \alpha\cos 2\pi f\tau_2)^2 + \alpha^2 \sin^2 2\pi f\tau_2\right] \tag{2.158}$$

avec $\alpha = \rho_2/\rho_1$.

Deux paramètres sont maintenant introduits : la bande de cohérenceB_c et le temps de cohérence t_c qui permettent de caractériser le canal de transmission vis-à-vis de la sélectivité en fréquences et de sa vitesse d'évolution.

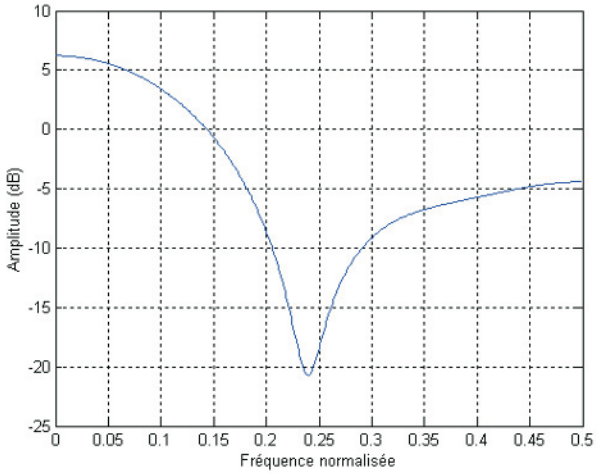


Figure 2.25 – Réponse en fréquences d'un canal à trajets multiples.

Bande de cohérence

Il existe plusieurs définitions de la bande de cohérence mais la définition la plus courante est :

$$B_c \approx \frac{1}{T_m} \tag{2.159}$$

où T_m est l'écart type du profil de densité de puissance du canal en fonction des retards τ_n des différents trajets.

Deux signaux modulés dont les fréquences porteuses sont séparées d'une quantité supérieure à la bande de cohérence du canal sont atténués par celui-ci de manière non corrélée. Ainsi, à un instant donné, si l'un des deux signaux est fortement atténué par le canal, l'autre sera, avec une forte probabilité, peu affecté par celui-ci.

Lorsque que la bande B occupée par un signal modulé est inférieure à la bande de cohérence, le canal est non sélectif en fréquences vis-à-vis de ce signal. Il présente une réponse en fréquences plate et une phase linéaire sur la bande B occupée par le signal modulé. Le signal modulé subit juste une atténuation et un déphasage en traversant le canal. Nous allons illustrer ce point en considérant la transmission d'un signal MAQ-M par un canal non sélectif en fréquences.

Soit $s(t)$ le signal MAQ-M placé à l'entrée du canal à trajets multiples non sélectif en fréquences.

$$s(t) = A\left[\sum_i a_i h(t-iT)\cos 2\pi f_0 t - \sum_i b_i h(t-iT)\sin 2\pi f_0 t\right]$$

et soit $r(t)$ la réponse du canal à $s(t)$:

$$r(t) = A \sum_{n=1}^{M} \rho_n \left[\sum_i a_i h(t - \tau_n - T) \cos\left(2\pi(f_0 + f_n^d)(t - \tau_n)\right) - \sum_i b_i h(t - \tau_n - T) \sin\left(2\pi(f_0 + f_n^d)(t - \tau_n)\right) \right]$$

Pour un canal non sélectif en fréquences, nous avons $B < B_c$. En notant que la bande B est proportionnelle à $1/T$, cela entraîne $T > T_m$ soit encore $T >> \tau_n \quad \forall n$. Dans l'expression de $r(t)$, nous pouvons donc négliger τ_n devant T ce qui donne :

$$r(t) = A \sum_{n=1}^{M} \rho_n \left[\sum_i a_i h(t - T) \cos(2\pi f_0 t + \varphi_n(t)) - \sum_i b_i h(t - T) \sin(2\pi f_0 t + \varphi_n(t)) \right]$$

avec :

$$\varphi_n(t) = f_n^d t - (f_0 + f_n^d)\tau_n$$

En posant :

$$a_c(t) = \sum_{n=1}^{M} \rho_n \cos\varphi_n(t) \text{ et } a_s(t) = \sum_{n=1}^{M} \rho_n \sin\varphi_n(t)$$

et :

$$\cos\phi(t) = \frac{a_c(t)}{\sqrt{a_c^2(t) + a_s^2(t)}} \text{ et } \sin\phi(t) = \frac{a_s(t)}{\sqrt{a_c^2(t) + a_s^2(t)}}$$

le signal $r(t)$ peut encore s'écrire :

$$r(t) = A\alpha(t) \left[\sum_i a_i h(t - iT) \cos(2\pi f_0 t + \phi(t)) - \sum_i b_i h(t - iT) \sin(2\pi f_0 t + \phi(t)) \right] \tag{2.160}$$

avec $\alpha(t) = \sqrt{a_c^2(t) + a_s^2(t)}$.

Pour un canal à trajets multiples non sélectif en fréquences, le signal modulé MAQ-M subit uniquement une atténuation $\alpha(t)$ et un déphasage $\phi(t)$. En modélisant les atténuations ρ_n, les retards τ_n, et les fréquences doppler f_n^d par des variables aléatoires mutuellement indépendantes alors, pour M suffisamment grand et pour t donné, $a_c(t)$ et $a_s(t)$ tendent vers des variables aléatoires gaussiennes non corrélées (théorème de la limite centrale). L'atténuation $\alpha(t)$, pour t donné, suit une loi de Rayleigh et la phase $\phi(t)$ est équirépartie sur $[0, 2\pi[$. En posant $\sigma_\alpha^2 = E\left\{\alpha^2\right\}$, la densité de probabilité s'écrit :

$$p(\alpha) = \frac{2\alpha}{\sigma_\alpha^2} \exp\left(-\frac{\alpha^2}{\sigma_\alpha^2}\right) \quad \alpha \geqslant 0 \tag{2.161}$$

L'atténuation $\alpha(t)$ peut prendre des valeurs très inférieures à l'unité et, dans ce cas, le signal non bruité reçu par le récepteur est très atténué. Son niveau est alors comparable, voire inférieur, à celui du bruit. On dit que le canal de transmission présente un évanouissement profond de Rayleigh.

Si la bande B occupée par le signal modulé est supérieure à la bande de cohérence, le canal est sélectif en fréquences. Sa réponse en fréquence, sur la bande B, n'est plus plate et certaines composantes spectrales du signal modulé peuvent être très atténuées. Le canal introduit une distorsion du signal modulé qui se traduit notamment par l'apparition du phénomène de l'Interférence Entre Symboles (IES). En présence d'IES, le signal à un instant de décision est fonction de l'état du signal modulé à cet instant mais aussi des états antérieurs et postérieurs à cet instant. L'IES intervient comme un bruit qui s'ajoute au bruit additif blanc gaussien et, bien sûr, dégrade les performances de la transmission.

Temps de cohérence

Le temps de cohérence t_c d'un canal à évanouissement est défini par :

$$t_c \approx \frac{1}{B_d} \tag{2.162}$$

où B_d est la bande doppler du canal qui est bien approchée par $f^d_{\max}$ défini par :

$$f^d_{\max} = \mathrm{Max} f^d_n \tag{2.163}$$

Le temps de cohérence du canal est une mesure de sa vitesse d'évolution au cours du temps. Si t_c est très supérieur à la durée T d'un état du signal modulé, le canal est dit à évanouissement lent. Pour un canal non sélectif en fréquences et à évanouissement lent, l'atténuation $\alpha(t)$ et la phase $\phi(t)$ sont pratiquement constantes sur la durée T voire sur quelques durées T.

Un canal est non sélectif en fréquences et à évanouissement lent s'il vérifie la condition suivante :

$$T_M B_d < 1 \tag{2.164}$$

2.4.2 Transmission sur canal non sélectif en fréquences et à évanouissements lents

Performances sur canal de Rayleigh

Pour ce canal, le signal modulé subit une atténuation $\alpha(t)$ et un déphasage $\phi(t)$ aléatoires de réalisations constantes sur une durée supérieure ou égale à T. En considérant un récepteur cohérent, la probabilité d'erreur par données binaires, conditionnellement à une réalisation α de l'atténuation $\alpha(t)$, est égale à :

$$\text{Modulation MDP-2 ou MDP-4} \quad Peb(\alpha) = \frac{1}{2}\mathrm{erfc}\sqrt{\frac{\alpha^2 E_b}{N_0}} \tag{2.165}$$

$$\text{Modulation MDF-2} \quad Peb(\alpha) = \frac{1}{2}\text{erfc}\sqrt{\frac{\alpha^2 E_b}{2N_0}} \tag{2.166}$$

On obtient la probabilité d'erreur Peb en moyennant $Peb(\alpha)$ sur les différentes réalisation de $\alpha(t)$.

$$Peb = \int_0^\infty Peb(\alpha)p(\alpha)d\alpha \tag{2.167}$$

où $p(\alpha)$ est la densité de probabilité de α.

En tenant compte du fait que, pour t donné, $\alpha(t)$ est une variable aléatoire de Rayleigh de densité de probabilité

$$p(\alpha) = \frac{\alpha}{\sigma_\alpha^2}\exp\left(-\frac{\alpha^2}{2\sigma_\alpha^2}\right) \quad \alpha \geqslant 0$$

les probabilités Peb ont pour expressions :

$$\text{Modulation MDP-2 ou MDP-4} \quad Peb = \frac{1}{2}\left(1 - \sqrt{\frac{\bar{E}_b/N_0}{1+\bar{E}_b/N_0}}\right) \tag{2.168}$$

$$\text{Modulation MDF-2} \quad Peb = \frac{1}{2}\left(1 - \sqrt{\frac{\bar{E}_b/N_0}{2+\bar{E}_b/N_0}}\right) \tag{2.169}$$

où $\bar{E}_b$ est l'énergie moyenne reçue par donnée binaire transmise :

$$\bar{E}_b = \text{E}\left\{\alpha^2\frac{A^2T_b}{2}\right\} = A^2T_b\sigma_\alpha^2 \tag{2.170}$$

Pour $\bar{E}_b/N_0$ grand, les probabilités d'erreurs peuvent être approchées par :

$$\text{MDP-2 ou MDP-4} \quad Peb \approx \frac{1}{4\bar{E}/N_0} \tag{2.171}$$

$$\text{MDF-2} \quad Peb \approx \frac{1}{2\bar{E}/N_0} \tag{2.172}$$

Sur un canal à évanouissement de Rayleigh, les performances des différents récepteurs sont fortement dégradées par rapport à celles obtenues sur canal gaussien (à $E_b/N0$ identique en entrée). En effet, sur canal gaussien, les probabilités d'erreur Peb décroissent de manière exponentielle en fonction du rapport signal à bruit E_b/N_0 alors que sur canal à évanouissement de Rayleigh, la décroissance de la probabilité Peb est proportionnelle à l'inverse du rapport signal à bruit moyen $\bar{E}_b/N_0$.

Pour améliorer les performances sur canal à évanouissement de Rayleigh, on utilise deux techniques, que l'on peut éventuellement combiner, la diversité et bien sûr le codage de canal (qui est en fait de la diversité d'information).

Performances sur canal de Rayleigh avec diversité

La diversité consiste à répéter plusieurs fois un même message (ou des répliques issues d'un codage de canal) en utilisant des fréquences porteuses espacées d'une quantité supérieure ou égale à la bande de cohérence B_c du canal. Dans ce cas, on parle de diversité en fréquence. Une alternative à cette approche, consiste à transmettre plusieurs fois un même message sur une même porteuse mais sur des intervalles de temps séparés d'une quantité supérieure ou égale au temps de cohérence t_c du canal. Il s'agit de la diversité temporelle. Enfin, on peut transmettre un message une seule fois et utiliser en réception plusieurs antennes distante typiquement de quelques longueurs d'ondes de la porteuse du signal modulé. Dans ce cas il s'agit de diversité d'espace.

Supposons que l'on utilise une modulation MDP-2 pour transmettre le message d'information et une diversité d'ordre L. En se plaçant sur l'intervalle de temps $[iT,\, (i+1)T[$ et en considérant une réception cohérente, après démodulation on dispose de L observations de la forme :

$$r_i^n = \alpha_i^n \sqrt{E_b} \cos\varphi_i + b_i^n \quad n = 1,\, 2,\, \cdots,\, L \tag{2.173}$$

où α_i^n est une atténuation de Rayleigh, φ_i la phase (0 ou π) portant l'information à transmettre et b_i^n un bruit blanc gaussien, de moyenne nulle et de variance égale à $N_0/2$. Les L atténuations α_i^n sont mutuellement indépendantes ainsi que les L termes de bruit b_i^n. Ces L atténuations peuvent être vues comme L sous-canaux indépendants, appelés également branches de diversité. E_b est ainsi l'énergie utilisée pour transmettre une donnée binaire par branche de diversité.

Pour prendre une décision en présence de diversité, on construit la variable Z_i de la manière suivante :

$$Z_i = \sum_{n=1}^{L} r_i^n \cdot \alpha_i^n$$

La probabilité d'erreur par donnée binaire Peb en présence de diversité est alors égale à :

$$Peb = \frac{1}{2} \Pr\left(Z_i > 0 | \varphi_i = \pi\right) + \frac{1}{2} \Pr\left(Z_i < 0 | \varphi_i = 0\right) \tag{2.174}$$

Conditionnellement à *une* réalisation des L atténuations α_i^n, la variable de décision Z_i est gaussienne de moyenne :

$$\begin{aligned} E\left\{Z_i\right\} &= \sqrt{E_b} \sum_{n=1}^{L} (\alpha_i^n)^2 \quad \text{si} \quad \varphi_i = 0 \\ E\left\{Z_i\right\} &= -\sqrt{E_b} \sum_{n=1}^{L} (\alpha_i^n)^2 \quad \text{si} \quad \varphi_i = \pi \end{aligned} \tag{2.175}$$

et de variance :

$$\sigma_Z^2 = \frac{N_0}{2} \sum_{n=1}^{L} (\alpha_i^n)^2 \tag{2.176}$$

Sous cette hypothèse, la probabilité d'erreur $Peb(\rho)$ est égale à :

$$Peb(\rho) = \frac{1}{2}\text{erfc}\sqrt{\rho} \tag{2.177}$$

avec :

$$\rho = \frac{E_b}{N_0} \sum_{n=1}^{L} (\alpha_i^n)^2$$

En moyennant la probabilité $Peb(\rho)$ sur les différentes réalisations de la variable aléatoire ρ, on obtient la probabilité d'erreur par donnée binaire en présence de diversité d'ordre L :

$$Peb = \int_0^\infty Peb(\rho) p(\rho) d\rho$$

La variable aléatoire ρ suit une loi du χ^2 ayant pour densité de probabilité :

$$p(\rho) = \frac{1}{(L-1)! \bar{m}^L} \rho^{L-1} \exp -\frac{\rho}{\bar{m}} \tag{2.178}$$

où $\bar{m}$ est égal à :

$$\bar{m} = \frac{E_b}{N_0} \text{E}\left\{(\alpha_j^n)^2\right\}$$

Après intégration, la probabilité d'erreur par donnée binaire est égale à :

$$Peb = \left(\frac{1-\eta}{2}\right)^L \sum_{n=0}^{L-1} \left(\begin{array}{c} L-1+n \\ n \end{array} \right) \left(\frac{1+\eta}{2}\right)^n \tag{2.179}$$

avec :

$$\eta = \sqrt{\frac{\bar{E}_b / LN_0}{1 + \bar{E}_b / LN_0}} \quad \text{et} \quad \left(\begin{array}{c} L-1+n \\ n \end{array} \right) = \frac{(L-1+n)!}{n!(L-1)!}$$

où $\bar{E}_b$ est l'énergie moyenne totale utilisée pour transmettre une donnée binaire d'information ($\bar{E}_b = LE_b$).

À fort rapport signal à bruit, la probabilité d'erreur Peb est bien approchée par :

$$Peb \approx \left(\begin{array}{c} 2L-1 \\ L \end{array} \right) \left(\frac{1}{4\bar{E}_b / LN_0}\right)^L \text{ pour } \frac{\bar{E}_b}{LN_0} >> 1 \tag{2.180}$$

En présence de diversité la probabilité d'erreur par donnée binaire Peb décroît suivant l'inverse du rapport signal à bruit à la puissance L.

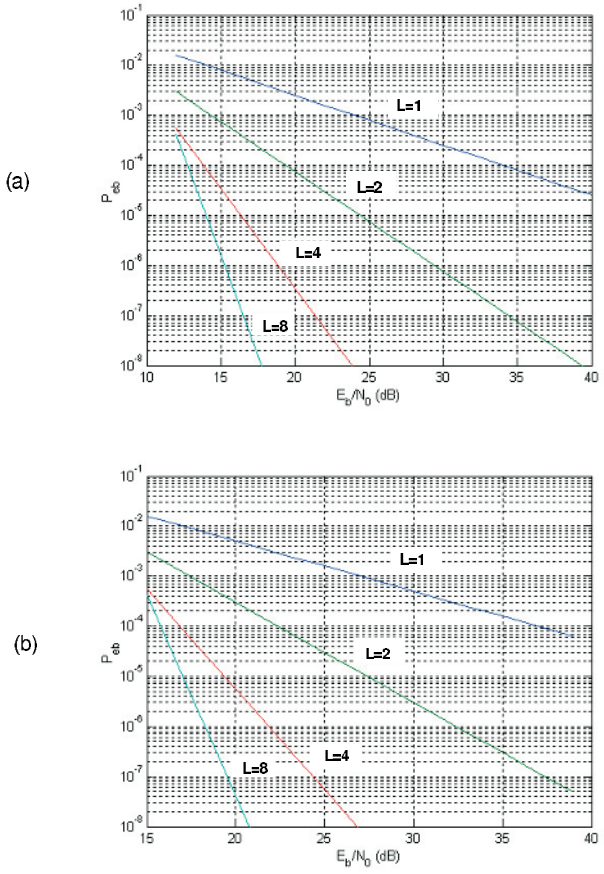


Figure 2.26 – Performance des modulations a) MDP-2 et b) MDF-2 en présence de diversité.

Pour une modulation MDF-2 le calcul de la probabilité d'erreur par donnée binaire en présence de réception cohérente est analogue à celui de la modulation MDP-2. On obtient le résultat suivant :

$$Peb = \left(\frac{1-\eta}{2}\right)^L \sum_{n=0}^{L-1} \left(\begin{array}{c} L-1+n \\ n \end{array} \right) \left(\frac{1+\eta}{2}\right)^n \tag{2.181}$$

avec cette fois-ci :

$$\eta = \sqrt{\frac{\bar{E}_b/LN_0}{2+\bar{E}_b/LN_0}}$$

À fort rapport signal à bruit, la probabilité d'erreur Peb est bien approchée

par :

$$Peb \approx \begin{pmatrix} 2L-1 \\ L \end{pmatrix} \left(\frac{1}{2\bar{E}_b/LN_0} \right)^L \quad \text{pour} \quad \frac{\bar{E}_b}{LN_0} >> 1 \tag{2.182}$$

Notons que la diversité telle que présentée ici est une forme de codage qui utiliserait un code à répétition et un décodage pondéré en réception. La figure (2.26) fournit les performances des modulations MDP-2 et MDF-2 en présence de diversité.

Transmission sur canal sélectif en fréquences et à évanouissements lents

Différentes stratégies de transmission sont possibles. On peut utiliser à l'émission une forme d'onde peu, voire pas, sensible à la sélectivité du canal ou corriger, en réception, les effets de la sélectivité du canal.

Transmission multiporteuses

La transmission multiporteuse utilise un multiplex de porteuses orthogonales modulées numériquement en phase (MDP-M), en fréquence (MDF-M) ou en amplitude et en phase (MAQ-M). Cette forme d'onde dite « parallèle », connue sous le nom de *Orthogonal Frequency Division Multiplexing* (*OFDM*), permet de s'affranchir de la sélectivité en fréquences des canaux de transmission.

Nous avons vu qu'un canal est sélectif en fréquences vis-à-vis d'un signal modulé occupant une bande B si sa bande de cohérence $B_c \approx 1/T_M$ est inférieure à B. Rappelons, d'autre part, que la bande B occupée par un signal modulé numériquement est proportionnelle à sa rapidité de modulation $1/T$. Pour construire une forme d'onde peu sensible à la sélectivité en fréquences du canal, on peut procéder de la manière suivante.

Divisons le débit $D = 1/T_b$ à transmettre en N sous débits $D' = 1/NT_b$, chaque débit élémentaire alimentant un modulateur à M états, de fréquence porteuse f_i. La rapidité de modulation des porteuses modulées est alors égale à $R = 1/T$ avec $T = NT_b \log_2(M)$. En choisissant N suffisamment grand, la rapidité de modulation des porteuses modulées peut devenir aussi petite que souhaité, et la bande B occupée par une porteuse modulée devient alors très inférieure à la bande de cohérence B_c du canal. En procédant ainsi le canal est non sélectif en fréquences vis-à-vis des porteuses modulées du multiplex. Pour un canal à trajets multiples chaque porteuse modulée se voit affectée d'une atténuation qui suit une loi de Rayleigh et d'un déphasage équiréparti sur $[0, 2\pi[$. Bien entendu, toutes les porteuses modulées ne sont pas affectées de la même façon au même instant par le canal, certaines sont fortement atténuées alors que d'autres le sont moins. Les performances en terme de probabilité d'erreur par porteuse modulée sont celles d'un canal de Rayleigh non sélectif en fréquences et à évanouissement lent.

Pour éviter en réception les gros paquets d'erreurs, on peut s'arranger pour que des données d'information qui se suivent soient transmises par des por-

teuses affectées différemment par le canal c'est-à-dire espacées d'une quantité au moins égale à la bande de cohérence B_c du canal. On peut également transmettre ces données en utilisant la même porteuse mais à des instants séparés d'une quantité au moins égale au temps de cohérence t_c du canal. Cette façon de procéder revient à faire un entrelacement fréquentiel combiné avec un entrelacement temporel.

Mise en œuvre des transmissions multiporteuses

En considérant une modulation MAQ-M sur chaque porteuse, le signal *OFDM* a pour expression :

$$s(t) = A \sum_{i=-\infty}^{+\infty} \sum_{n=0}^{N-1} \Re\left\{c_{n,i} h(t - iT) \exp(j2\pi f_n t)\right\} \tag{2.183}$$

où $c_{n,i} = a_{n,i} + jb_{n,i}$ est un symbole complexe de modulation, $h(t)$ une porte d'amplitude unité et de durée T et N est le nombre de porteuse de fréquence f_n.

Plaçons-nous sur l'intervalle de temps $[iT,\, (i+1)T[$, le signal $s(t)$ est égal à :

$$s(t) = A \sum_{n=0}^{N-1} \Re\left\{c_{n,i} \exp(j2\pi f_n t)\right\} \quad \forall\, t \in [iT,\, (i+1)T[\tag{2.184}$$

La mise en œuvre d'un signal *OFDM* nécessite de réaliser N modulateurs MAQ-M de fréquences porteuses f_n. On peut montrer que ces N modulateurs peuvent être réalisés à partir d'une transformée de Fourier discrète inverse, ce qui permet une complexité raisonnable de mise en œuvre.

En considérant N porteuses orthogonales, les fréquences f_n doivent être espacées au moins de $1/T$.

$$f_n = \frac{n}{T} \qquad n = 0,\, 1,\, \cdots,\, (N-1) \tag{2.185}$$

La densité spectrale de puissance $\gamma_{OFDM}(f)$ d'un signal *OFDM* en bande de base est proportionnelle à :

$$\gamma_{OFDM}(f) \propto \sum_{n=0}^{N-1} \left[\frac{\sin \pi(f - n/T)T}{\pi(f - n/T)T}\right]^2 \tag{2.186}$$

ce qui donne un spectre plat dans la bande de fréquences $B = (N-1)/T$.

Le signal $s(t)$ peut être échantillonné à la fréquence f_e à la condition que f_e vérifie le critère de Nyquist soit :

$$f_e \geqslant \frac{2(N-1)}{T} \tag{2.187}$$

On peut choisir $f_e = 2N/T$ et ainsi le signal $s(t)$ échantillonné aux instants lT_e avec $T_e = 1/f_e$ est égal à :

$$s_l = s(lT_e) = A \sum_{n=0}^{N-1} \Re \left\{ c_{n,i} \exp\left(j2\pi \frac{nl}{2N} \right) \right\} \quad l = 0,\, 1,\, \cdots,\, (2N-1) \quad (2.188)$$

Sous cette forme, s_l n'est pas une transformée de Fourier discrète inverse. En introduisant des symboles virtuels de la manière suivante :

$$c_{2N-n,\,i} = c^*_{n,\,i} \quad n = 1,\, \cdots,\, (N-1) c_{N,i} = \Im \{c_{0,\,i}\} \quad (2.189)$$

et en remplaçant $c_{0,i}$ par $\Re \{c_{0,i}\}$, on peut montrer que le signal échantillonné s_l peut effectivement s'écrire sous forme d'une transformée de Fourier discrète inverse :

$$s_l = A \sum_{n=0}^{2N-1} c_{n,\,i} \exp\left(j2\pi \frac{nl}{2N} \right) \quad (2.190)$$

Sur chaque intervalle de durée T on réalise une transformée de Fourier discrète inverse sur $2N$ points qui permet de produire les N signaux MAQ-M en bande de base. Un convertisseur numérique-analogique suivi d'une transposition en fréquence permet d'obtenir les signaux modulés sur fréquence porteuse.

En réception, après amplification et transposition en bande de base du signal *OFDM*, le décodage des symboles $c_{n,i} = a_{n,i} + jb_{n,i}$ de modulation peut également se faire par transformée de Fourier discrète directe.

Nous avons vu que la durée T des symboles de modulation augmente avec N et, pour de grandes valeurs de N, peut devenir comparable au temps de cohérence t_c du canal de transmission. Dans ce cas, l'hypothèse de canal à évanouissement lent n'est plus vérifiée. Il existe donc des limites au choix du paramètre N. Pour un canal à trajets multiples, si le choix de N ne permet pas d'obtenir $B < B_c$, le canal reste sélectif en fréquences vis-à-vis des porteuses modulées et de l'interférence entre symboles apparaît.

Pour s'affranchir de l'interférence entre symboles résiduelle sans augmenter N, on peut utiliser le principe de l'intervalle de garde. Pour un canal à trajets multiples, les trajets de propagation arrivent au récepteur avec des retards τ_n. En appelant τ_{Max} le plus grand de ces retards, l'intervalle de garde Δ doit vérifier l'inégalité suivante :

$$\Delta \geqslant \tau_{\text{Max}} \quad (2.191)$$

Posons :

$$T = t_s + \Delta$$

En présence d'un intervalle de garde, les symboles de modulation sont toujours de durée T mais la transformée de Fourier discrète directe en réception est réalisée sur les intervalles de temps $[iT + \Delta,\, (i+1)T[$. En procédant ainsi, on peut vérifier que sur cet intervalle de temps seul le symbole de modulation émis entre iT et $(i+1)T$ est pris en compte pour le décodage : il n'y a donc pas d'interférence entre symboles.

L'introduction d'un intervalle de garde a deux conséquences. La première est que seule une partie de l'énergie émise en émission est exploitée en réception. En effet, on émet chaque symbole de modulation sur une durée T et on récupère ce même symbole à partir d'une observation de durée $t_s = T - \Delta$. La perte, exprimée en dB, est donc égale à :

$$10 \log \left(\frac{1}{1 + \Delta / t_s} \right) \tag{2.192}$$

La seconde conséquence est que l'orthogonalité des porteuses doit être assurée de façon à pouvoir séparer ces porteuses en réception, c'est-à-dire en les espaçant d'une quantité $1/t_s$. La bande de fréquences occupée par le signal *OFDM* avec intervalle de garde est donc de :

$$B = \frac{N - 1}{t_s} \tag{2.193}$$

soit une expansion de bande par rapport à un système sans intervalle de garde de $1 + \Delta / t_s$.

En présence d'intervalle de garde on a donc intérêt à choisir Δ de façon à minimiser les dégradations du rapport signal à bruit et de l'efficacité spectrale, c'est-à-dire à choisir Δ le plus petit possible par rapport à la durée t_s.

Transmission avec égalisation en réception

En présence de canal sélectif en fréquences, on peut utiliser en émission une modulation linéaire mono-porteuse (MDP-M, MAQ-M) et corriger l'IES créée par le canal par un égaliseur. Les quelques architectures d'égaliseurs sont présentées dans le chapitre 11.

Si on compare l'approche *OFDM* et la transmission mono-porteuse avec égalisation en termes de complexité de mise en œuvre, la difficulté de réalisation de la transmission *OFDM* se situe dans le modulateur alors qu'avec l'égalisation, elle se situe au niveau du récepteur.

Chapitre 3

Limites théoriques

L'invention récente des turbocodes et la redécouverte des codes LDPC ont remis au goût du jour les limites théoriques de la transmission qui étaient jusqu'alors réputées inaccessibles. Ce chapitre fournit les bases conceptuelles nécessaires à la compréhension et au calcul de ces limites, en particulier celles qui correspondent à des situations réelles de transmission avec des messages de taille finie et des modulations binaires.

3.1 La théorie de l'information

3.1.1 Canal de transmission

Un *canal* est un milieu quelconque où des symboles peuvent être propagés (télécommunications) ou enregistrés (mémoires de masse). Par exemple, les symboles 0 et 1 de l'alphabet binaire peuvent être représentés par la polarité d'une tension appliquée à une extrémité d'une paire de fils conducteurs, en convenant par exemple que $+V$ correspond à 1 et $-V$ à 0. Alors, la mesure de la polarité à l'autre extrémité fera connaître quel symbole binaire a été émis. À l'extrémité émettrice, la polarité est changée à intervalles régulièrement espacés pour représenter les bits d'un message et permettre de reconstituer ce message à l'extrémité réceptrice. Ce schéma est bien trop simple pour illustrer les systèmes modernes de télécommunications mais, d'une façon générale, c'est le signe d'une grandeur physique réelle qui représente un symbole binaire en sortie du canal. Le plus souvent, un symbole binaire est représenté par une certaine forme d'onde et l'opération qui associe une suite de formes d'onde à la suite des symboles du message est la modulation. Les modulations ont fait l'objet du chapitre précédent.

Nous nous intéressons à une situation où le canal est peu fiable, c'est-à-dire où l'observation à l'extrémité réceptrice ne permet pas avec certitude d'identifier le bit réellement émis car une grandeur parasite, le bruit, indépendant du message émis et aléatoire, est ajoutée à la grandeur utile (il peut s'y rajouter

des effets parasites d'atténuation, comme sur le canal de Rayleigh). Le bruit thermique est bien représenté par un processus aléatoire gaussien. Quand la démodulation est effectuée de manière optimale, elle a pour résultat une variable aléatoire gaussienne dont le signe représente la meilleure hypothèse quant au symbole binaire émis. Le canal est alors caractérisé par son *rapport signal à bruit*, défini comme le rapport de la puissance du signal utile à celle du bruit perturbateur. Pour un rapport signal à bruit donné, les décisions prises quant aux symboles binaires émis sont affectées d'une probabilité d'erreur constante, ce qui conduit au modèle simple du canal binaire symétrique.

3.1.2 Un exemple : le canal binaire symétrique

C'est le modèle de canal le plus simple dont il a déjà été question à la section (1.3). Un canal peut être décrit en général par les probabilités de transition des symboles qui y entrent vers les symboles qui en sortent. Un canal binaire symétrique est ainsi représenté dans la figure 3.1. Ce canal est *sans mémoire*, en ce sens qu'il opère séparément sur les bits entrants successifs. Son symbole d'entrée X et son symbole de sortie Y sont tous deux binaires. Si $X = 0$ (respectivement $X = 1$), il existe une probabilité p que $Y = 1$ (resp. $Y = 0$). p est appelée probabilité d'erreur du canal.

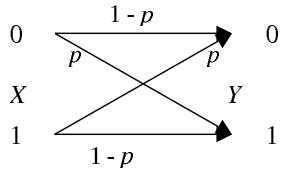


Figure 3.1 – Canal binaire symétrique de probabilité d'erreur p. Les probabilités de transition d'un symbole d'entrée vers un symbole de sortie sont égales deux à deux.

Une autre description du même canal peut être donnée de la manière suivante : soit E une variable aléatoire binaire prenant la valeur 1 avec une probabilité $p < 1/2$ et la valeur 0 avec la probabilité $1-p$. L'hypothèse que $p < 1/2$ ne restreint pas la généralité du modèle car l'échange des signes arbitraires 0 et 1 conduit à remplacer une probabilité d'erreur initiale $p > 1/2$ par $1-p < 1/2$. Le comportement du canal peut être décrit par l'expression algébrique $Y = X \oplus E$, où X et Y sont les variables binaires respectivement à l'entrée et à la sortie du canal, E une variable binaire d'erreur, et $\oplus$ représente l'addition modulo 2.

Muni de l'addition modulo 2 et de la multiplication ordinaire (avec $0 \times b = 0$ et $1 \times b = b$, où b est un symbole binaire quelconque), l'ensemble binaire $\{0, 1\}$ acquiert la structure mathématique d'un *corps* : chacun de ses éléments a un inverse pour l'addition, il existe un élément neutre pour l'addition (0), tout élément non nul (réduit ici au seul élément 1) a un inverse multiplicatif, puisque 1 est son propre inverse et constitue aussi l'élément neutre pour la

multiplication. Le corps binaire $\{0, 1\}$ est le corps qui contient le plus petit nombre possible d'éléments.

Configurations d'erreurs sur le canal binaire symétrique

Supposons maintenant que nous nous intéressions, non plus à un symbole particulier unique, mais à un ensemble de n symboles (consécutifs ou non) constituant un *mot*, noté $\mathbf{x} = (x_1 x_2 \dots x_n)$. Le fonctionnement du canal est décrit par l'addition vectorielle modulo 2 du vecteur d'information $\mathbf{y} = (y_1 y_2 \dots y_n)$ et d'un vecteur d'erreur $\mathbf{e} = (e_1 e_2 \dots e_n)$:

$$\mathbf{y} = \mathbf{x} \oplus \mathbf{e} \tag{3.1}$$

la notation $\oplus$ désignant maintenant l'addition modulo 2 de deux mots, symbole à symbole. L'hypothèse que le canal binaire symétrique est sans mémoire entraîne que les variables aléatoires e_i, $i = 1...n$, sont mutuellement indépendantes. Les configurations d'erreurs possibles sont au nombre de 2^n, et leur probabilité ne dépend, pour une probabilité d'erreur du canal p donnée, que du poids $w(e)$ de la configuration d'erreurs réalisée e, défini comme le nombre de symboles 1 qu'elle contient. Ainsi, la probabilité d'apparition d'une configuration d'erreurs particulière de poids $w(e)$ affectant un mot de longueur n vaut :

$$P_{\mathbf{e}} = p^{w(\mathbf{e})}(1-p)^{n-w(\mathbf{e})} \tag{3.2}$$

Comme p a été supposée inférieure à $1/2$, la probabilité P_e est une fonction décroissante du poids $w(\mathbf{e})$, quel que soit n.

La probabilité d'apparition d'une configuration d'erreurs quelconque de poids w vaut :

$$P_w = \left(\begin{array}{c} n \\ w \end{array} \right) p^w (1-p)^{n-w} \tag{3.3}$$

Le poids des configurations d'erreurs suit donc une distribution de Bernoulli dont l'espérance mathématique (ou moyenne) est np et la variance (l'espérance du carré de la différence entre sa valeur effective et sa moyenne) est $np(1-p)$.

Information mutuelle et capacité du canal binaire symétrique

Pour caractériser un canal, il nous faut d'abord mesurer la quantité d'information qu'un symbole Y sortant d'un canal apporte en moyenne sur le symbole correspondant qui y entre, X. Cette grandeur appelée *information mutuelle* et dont l'unité est le Shannon (Sh), est définie pour un canal à entrée et sortie discrètes par :

$$I(X;Y) = \sum_X \sum_Y \Pr(X,Y) \log_2 \frac{\Pr(X\,|Y\,)}{\Pr(X)} = \sum_X \sum_Y \Pr(X,Y) \log_2 \frac{\Pr(X,Y)}{\Pr(X)\Pr(Y)} \tag{3.4}$$

Dans cette expression, les sommes sont étendues à toutes les valeurs discrètes que peuvent prendre X et Y dans un alphabet donné. $\Pr(X,Y)$ désigne la probabilité conjointe de X et de Y, $\Pr(X|Y)$ la probabilité de X conditionnellement à Y (c'est-à-dire quand Y est donnée), $\Pr(X)$ et $\Pr(Y)$ sont les probabilités marginales de X et Y (c'est-à-dire de chacune des variables X et Y quelle que soit la valeur prise par l'autre : $\Pr(X) = \sum_Y \Pr(X,Y)$ et $\Pr(Y) = \sum_X \Pr(X,Y)$). Ces différentes probabilités sont reliées selon la règle de Bayes :

$$\Pr(X,Y) = \Pr(X\,|Y\,)\Pr(Y) = \Pr(Y\,|X\,)\Pr(X) \tag{3.5}$$

La première égalité dans (3.4) définit $I(X;Y)$ comme l'accroissement logarithmique de la probabilité de X qui résulte en moyenne de la donnée Y, c'est-à-dire la quantité d'information moyenne que la connaissance de Y apporte à celle de X. La deuxième égalité dans (3.4), déduite de la première en utilisant (3.5), montre que cette grandeur est symétrique en X et en Y. La quantité d'information que la donnée de Y apporte sur X est donc égale à celle que X apporte sur Y, ce qui justifie le nom d'information mutuelle.

L'information mutuelle ne suffit pas à caractériser le canal, car elle dépend aussi de l'*entropie* de la source, c'est-à-dire de la quantité d'information qu'elle produit en moyenne par symbole émis plus précisément le nombre moyen de bits nécessaires pour représenter chaque symbole est donné par :

$$H(X) = \sum_X \Pr(X) \log_2 (\Pr(X))$$

On définit la *capacité* d'un canal comme le maximum de l'information mutuelle de ses variables aléatoires d'entrée et de sortie par rapport à toutes les distributions de probabilité possibles de la variables d'entrée, et on montre que ce maximum est atteint pour un canal symétrique et sans mémoire lorsque la variable d'entrée du canal, X, est équiprobable (ce qui a aussi pour effet de rendre l'entropie de la source maximale). Par exemple, pour le canal binaire symétrique, la capacité est donnée par :

$$C = 1 + p \log_2 (p) + (1-p) \log_2 (1-p) \text{ (Sh)} \tag{3.6}$$

Cette capacité est maximale pour $p = 0$ (elle vaut alors 1 Sh, comme l'entropie de la source : le canal est alors « transparent ») et nulle pour $p = 1/2$, comme on pouvait s'y attendre puisqu'alors l'incertitude est totale.

3.1.3 Aperçu sur le théorème fondamental du codage

Le code le plus simple que l'on puisse imaginer est le code à répétition qui consiste à émettre un bit d'information sous la forme de plusieurs symboles identiques. Le décodage ferme s'effectue selon le principe du vote majoritaire, le décodage souple par simple addition des échantillons reçus. Si le canal est gaussien, le codage à répétition n'apporte aucun gain dans le cas du décodage

souple. Par exemple, transmettre deux fois le même symbole en allouant à chacun d'eux la moitié de l'énergie disponible et reconstituer le symbole émis par addition ne fournit pas de meilleur résultat que transmettre un seul symbole avec toute l'énergie disponible. Le vote majoritaire, quant à lui, ne peut s'envisager qu'à partir d'une émission triple et dans tous les cas, sur canal gaussien, ce procédé dégrade le bilan de liaison par rapport à la solution non codée. Il faut cependant noter que la répétition de messages est une technique largement répandue, non pas en tant que procédé de codage correcteur, mais en tant que technique de récupération de paquets de messages erronés ou perdus durant la transmission. Cette technique dite *ARQ* (*Automatic Repeat reQuest*)ne peut pas être mise en œuvre dans tous les systèmes, notamment les liaisons point à multipoint (e.g. diffusion de télévision).

Les codes ne sont idéalement efficaces que si leurs mots sont longs, en ce sens que la probabilité d'erreur ne peut être rendue arbitrairement petite que si la longueur de ces mots tend vers l'infini. Par ailleurs, un bon code doit conserver un taux d'émission ou rendement $R = k/n$ non nul quand le nombre k des bits d'information tend vers l'infini. Qu'une communication sans erreurs soit effectivement possible asymptotiquement pour un taux d'émission non nul est un résultat majeur de la théorie de l'information, appelé *théorème fondamental du codage de canal*, qui a précédé les tentatives de construire des codes pratiques, donc de longueur finie. Ce théorème a constitué une puissante incitation à la recherche de nouveaux codes, toujours plus efficaces les uns que les autres. Il posait de plus un défi aux ingénieurs dans la mesure où la preuve du théorème reposait sur le *codage aléatoire*, dont le décodage est bien trop complexe pour être envisagé en pratique.

Bien que la preuve mathématique du théorème fondamental sous sa forme la plus générale comporte des mathématiques assez difficiles, nous croyons qu'il peut être facilement compris à l'aide de la *loi des grands nombres*. Cette loi énonce simplement que les réalisations expérimentales ont des fréquences, définies comme le rapport du nombre des occurrences constatées au nombre total des épreuves, qui tendent vers les probabilités des évènements correspondants quand le nombre des épreuves tend vers l'infini. Considérons, par exemple, le jeu de pile ou face. Avec une pièce « honnête », on pourrait théoriquement aboutir, après 10000 tirages, à la suite formée exclusivement de toutes les piles (ou de toutes les faces), mais avec une probabilité qui n'est que de $2^{-10000} \approx 10^{-3010}$ (par comparaison, une seconde représente environ 10^{-18} du temps écoulé depuis la création de l'univers). En fort contraste, la probabilité que la fréquence des faces (ou des piles) soit proche de la moyenne $1/2$ (appartenant par exemple à l'intervalle 0,47-0,53) est très voisine de 1. De manière similaire, une configuration d'erreurs de poids proche de np quand n symboles sont émis sur un canal binaire symétrique de probabilité d'erreur p est très probable, à condition que le message envoyé soit suffisamment long.

3.1.4 Interprétation géométrique

Considérons l'espace fini des mots de code de n bits munis de la distance de Hamming, qui sera noté H_n. Il contient 2^n éléments qui sont dits ses points. En termes géométriques, dire que le nombre d'erreurs est voisin de np avec un grande probabilité signifie que le mot reçu est représenté par un point qui est très probablement voisin de la surface d'une hypersphère à n dimensions dans H_n, centrée sur le mot émis et dont le rayon est égal au nombre d'erreurs moyen attendu np. Si la distance minimale d du code est supérieure à deux fois ce nombre, le point à la surface de cette hypersphère est plus proche du mot effectivement émis que de toute autre mot du code et donc l'identifie sans ambiguïté. La règle de décodage optimale, qui a déjà été présentée dans le chapitre 1, peut donc être énoncée ainsi : « *Choisir le mot du code le plus proche du mot reçu* ».

La probabilité que cette règle ait un résultat erroné est d'autant plus petite que n est grand et elle tend vers 0 (en supposant que p est maintenu constant) quand n tend vers l'infini, pourvu que $d > 2np$.

Toujours dans des termes géométriques, la construction du meilleur code possible peut donc être interprétée comme consistant à choisir $M < 2^n$ points appartenant à H_n de telle sorte qu'ils soient les plus éloignés possibles les uns des autres (notons que l'inégalité $M < 2^n$ implique que le code est nécessairement redondant). Pour une valeur donnée de la probabilité d'erreur p du canal (toujours supposé binaire symétrique) il est clair qu'il existe une limite au nombre M de points qui peuvent être placés dans H_n en maintenant la distance entre ces points supérieure à $2np$. Soit $M_{\max}$ ce nombre. La grandeur

$$C = \lim_{n \to \infty} \frac{\log_2 (M_{\max})}{n}$$

mesure en shannons la plus grande quantité d'information par symbole qui peut être communiquée sans erreur à travers le canal, et elle se trouve coïncider avec la capacité du canal définie dans la section 3.1. Aucune procédure explicite permettant de déterminer $M_{\max}$ points dans H_n en maintenant la distance entre ces points supérieure à $2np$ n'est en général connue, sinon dans quelque cas simples et peu utiles.

3.1.5 Codage aléatoire

Le *codage aléatoire*, c'est-à-dire la construction d'un code en choisissant au hasard ses éléments, est un moyen de choisir M points éparpillés dans l'espace H_n. Ce moyen est optimal pour la distribution des distances, quand n tend vers l'infini. Le codage aléatoire permet que les points soient, en moyenne, également répartis dans toutes les n dimensions de H_n et il atteint un taux d'émission moyen égal à la capacité du canal. Pour un code comportant M mots de longueur n, il consiste à tirer au hasard chaque bit d'un mot indépendamment des autres avec la probabilité $1/2$ qu'il soit 0 ou 1, les M mots qui constituent

le code étant tirés de la même façon indépendamment les uns des autres. La probabilité d'un mot particulier $\boldsymbol{c}$ est $P_c = 2^{-n}$. On obtient ainsi des mots dont les poids suivent une distribution de Bernoulli et la probabilité d'obtenir un mot quelconque de poids w est donnée par (3.3) pour $p = 1/2$, soit :

$$P_w = \begin{pmatrix} n \\ w \end{pmatrix} 2^{-n} \tag{3.7}$$

L'espérance mathématique, ou moyenne, de ce poids est $n/2$ et sa variance vaut $n/4$. Pour n très grand, une bonne approximation de la distribution de poids des mots obtenus par codage aléatoire est une distribution gaussienne. Si on remplace w/n par la variable aléatoire continue X, la probabilité que $X \in (x, x + \mathrm{d}x)$ est $p_X(x)\mathrm{d}x$, où :

$$p_X(x) = \sqrt{\frac{2n}{\pi}} \exp\left[2n(x - 1/2)^2\right] \tag{3.8}$$

Cette fonction a un maximum à $x = 1/2$, donc pour $w = n/2$, et prend des valeurs décroissantes symétriques quand x s'écarte de $1/2$. Elle est centrée autour de son maximum $x = 1/2$ et la largeur de la région où elle prend des valeurs non négligeables décroît comme $1/\sqrt{n}$, et donc tend vers 0 quand n tend vers l'infini.

Cependant, le décodage d'un code obtenu par codage aléatoire est impossible en pratique puisque le décodage d'un *unique* mot reçu impliquerait de le comparer à *tous* les mots du code. Puisque des mots longs sont nécessaires à de bonnes performance, le nombre des mots du code (2^{Rn}), donc le nombre de combinaisons nécessaires, est considérable si Rn est grand, ce qui est le cas en pratique. C'est pourquoi les recherches sur les codes correcteurs d'erreurs se sont orientées vers des règles de codage non aléatoires offrant la voie à un décodage de complexité raisonnable.

Aucun moyen général n'est connu pour construire un code ayant $M_{\max}$ mots, pour une valeur arbitraire de n et une probabilité d'erreur donnée p. On sait avec certitude, ou on conjecture, qu'un petit nombre de schémas sont optimaux pour des valeurs données de M et n, pour quelques canaux simples. En l'absence d'une règle de construction générale de codes optimaux, la recherche a porté sur des codes satisfaisant à un critère plus facile : celui de distance minimale, pour lequel un code est d'autant meilleur que sa distance minimale est plus grande. La pertinence de ce critère n'a pas été remise en question avant la fin des années 1980. Ce critère ne tient pas compte du nombre des mots du code à la distance minimale d'un mot donné (ou multiplicité), alors qu'une grande valeur de ce nombre entraîne une dégradation des performances. Les turbocodes, qui seront examinés dans les chapitres suivants, n'ont pas été initialement construits pour satisfaire à ce critère. Leur distance minimale peut être petite (du moins si on la compare aux bornes connues sur la plus grande distance minimale possible et notamment la borne de Gilbert-Varshamov qui sera vue plus loin) mais leur multiplicité est aussi très petite. Ces propriétés entraînent qu'il peut exister

un « plancher d'erreur » (*error floor*) c'est-à-dire une décroissance beaucoup moins rapide de la probabilité d'erreur de leur décodage en fonction du rapport signal à bruit quand celui-ci est grand, que lorsqu'il est petit. Ce phénomène de plancher d'erreur peut également être visible avec les codes LDPC, bien que ceux-ci puissent être conçus sur le critère de la distance minimale. Quoiqu'il en soit, dans le cas des turbocodes comme dans celui des LDPC, puisque la finalité d'un codage correcteur est d'améliorer les communications quand le canal est mauvais, nous pouvons dire que ces codes sont bons quand ils sont utiles et médiocres quand ils le sont moins.

Codes imitant le codage aléatoire

Une idée simple consiste à essayer de construire des codes « imitant » le codage aléatoire, en un certain sens. Puisque les performances d'un code dépendent essentiellement de la distribution de ses distances, et celles d'un code linéaire de la distribution de ses poids, on peut se donner pour tâche de construire un code linéaire ayant une distribution de poids voisine de celle du codage aléatoire. Cette idée a été peu exploitée directement, mais on peut interpréter les turbocodes comme en étant une première mise en oeuvre. Avant de revenir à la conception de procédés de codage, nous allons faire une remarque intéressante concernant les codes qui imitent le codage aléatoire.

La probabilité d'obtenir un mot de longueur n et de poids w en tirant au hasard avec une probabilité de $1/2$ chacun des bits 0 et 1, indépendamment les uns des autres, est donnée par (3.7). En effectuant 2^k fois le tirage d'un mot, on obtient un nombre moyen de mots de poids w égal à :

$$N_{w,k} = \left(\begin{array}{c} n \\ w \end{array} \right) 2^{-(n-k)}$$

En supposant que n, k et w sont grands, nous pouvons exprimer approximativement $\left(\begin{array}{c} n \\ w \end{array} \right)$ à l'aide de la formule de Stirling, soit :

$$\left(\begin{array}{c} n \\ w \end{array} \right) = \frac{1}{\sqrt{2\pi}} \frac{n^{n+1/2}}{w^{w+1/2} \left(n-w\right)^{n-w+1/2}}$$

Le poids minimal obtenu en moyenne, soit $w_{\min}$, est le plus grand nombre tel que $N_{w_{\min},k}$ ait 1 pour meilleure approximation entière. Le nombre $N_{w_{\min},k}$ est donc petit. Il nous suffira de le poser égal à une constante λ voisine de 1, qu'il ne sera pas nécessaire de préciser davantage car elle s'éliminera du calcul. On doit donc avoir :

$$2^{-(n-k)} \frac{1}{\sqrt{2\pi}} \frac{n^{n+1/2}}{w_{\min}^{w_{\min}+1/2} \left(n-w_{\min}\right)^{n-w_{\min}+1/2}} = \lambda$$

En prenant les logarithmes de base 2 et en négligeant les constantes vis-à-vis de n, k et $w_{\min}$ qui tendent vers l'infini, on obtient :

$$1 - \frac{k}{n} \approx H_2 \left(w_{\min}/n\right)$$

où $H_2(\cdot)$ est la fonction *entropie binaire* :

$$\begin{array}{rcl} H_2 & = & -x\log_2(x) - (1-x)\log_2(1-x) \quad \text{pour } 0 < x < 1 \\ & = & 0 \quad \text{pour } x = 0 \text{ ou } x = 1 \end{array}$$

Le poids $w_{\min}$ est le poids minimal moyen d'un code obtenu par tirage aléatoire. Parmi l'ensemble de tous les codes linéaires ainsi obtenus (poids et distances étant donc confondus), il en est au moins un dont la distance minimale d est supérieure ou égale au poids moyen $w_{\min}$, de sorte que l'on a :

$$1 - \frac{k}{n} \leqslant H_2(d/n) \tag{3.9}$$

C'est la forme asymptotique de la borne de Gilbert-Varshamov qui relie la distance minimale d du code ayant la plus grande distance minimale possible à ses paramètres k et n. C'est une borne inférieure mais, sous sa forme asymptotique, elle est très proche de l'égalité. Un code dont la distance minimale vérifie cette borne avec égalité est considéré comme bon pour le critère de distance minimale. Cela montre qu'un code construit avec une distribution de poids voisine de celle du codage aléatoire est aussi bon pour ce critère.

3.2 Limites théoriques de performance

3.2.1 Canal à entrée binaire et sortie réelle

Seul le cas du canal binaire symétrique, à probabilité d'erreur p constante, a été considéré jusqu'à présent. Au lieu d'admettre une probabilité d'erreur constante, on peut considérer que la probabilité d'erreur varie en fait d'un symbole à un autre parce que l'échantillon de bruit qui affecte la grandeur reçue varie aléatoirement. Ainsi, en présence de bruit gaussien, la grandeur sortant du démodulateur optimal est une variable aléatoire gaussienne dont le signe représente la décision optimale. Nous considérons le canal qui a pour grandeur de sortie cette variable aléatoire réelle, que nous notons a. Il peut être montré que cette grandeur est liée à la décision optimale $\hat{x}$, c'est-à-dire à la meilleure hypothèse quant au bit émis x, et à la probabilité d'erreur « instantanée » p_a, selon la relation :

$$a = -(-1)^{\hat{x}} \ln\left(\frac{1-p_a}{p_a}\right) \tag{3.10}$$

ce qui entraîne, en supposant p_a inférieure à $1/2$:

$$p_a = \frac{1}{\exp(-(-1)^{\hat{x}}a) + 1} = \frac{1}{\exp(|a|) + 1} \tag{3.11}$$

Nous entendons par probabilité d'erreur instantanée la probabilité d'erreur p_a qui affecte le symbole reçu quand la grandeur réelle mesurée en sortie du canal est a. L'inégalité $p_a < 1/2$ rend $\ln\left(\frac{1-p_a}{p_a}\right)$ positif et alors la meilleure décision

est $\hat{x} = 1$ quand a est positif et $\hat{x} = 0$ quand a est négatif. Par ailleurs, la valeur absolue $|a| = \ln\left(\frac{1-p_a}{p_a}\right)$ est une fonction décroissante de la probabilité d'erreur de la décision, et elle en mesure donc la fiabilité. Elle est nulle pour $p_a = 1/2$ et tend vers l'infini quand la probabilité d'erreur p_a tend vers 0 (la décision devient alors absolument fiable). La quantité réelle que définit (3.10) est appelée *valeur relative* ou plus souvent *logarithme du rapport de vraisemblance* (LRV) du symbole binaire correspondant.

La capacité du canal ainsi défini peut être calculée comme le maximum par rapport à X de l'information mutuelle $I(X;Y)$, définie en généralisant (3.4) à $Y = a$ réel. Cette généralisation est possible mais l'expression de la capacité ainsi obtenue ne sera pas donnée ici. Nous nous contenterons de remarquer que cette capacité est supérieure à celle du canal binaire symétrique qui s'en déduit par prise de décision ferme, c'est-à-dire restreinte au symbole binaire $Y = \hat{x}$, d'un facteur qui croît quand le rapport signal à bruit du canal décroît. Il atteint $\pi/2$ quand on fait tendre ce rapport vers 0, si le bruit est gaussien. Pour un rapport signal à bruit donné, le canal binaire à l'entrée et continu en sortie est donc meilleur que le canal binaire symétrique qui s'en déduit en prenant des décisions fermes. Ce canal est aussi plus simple que le canal à décision ferme, puisqu'il ne comporte pas d'organe de prise de décision binaire en fonction de la grandeur réelle reçue. La prise de décision ferme entraîne la perte de l'information portée par les variations individuelles de cette grandeur, ce qui explique que la capacité du canal à sortie souple soit supérieure.

3.2.2 Capacité d'un canal de transmission

Nous considérons ici le cas le plus général ou l'entrée et la sortie du canal ne sont plus seulement des grandeurs scalaires mais peuvent être des vecteurs dont la dimension N est fonction du système de modulation. Par exemple, nous aurons $N = 1$ pour une modulation d'amplitude et $N = 2$ pour une modulation de phase à plus de deux états. X et Y sont donc remplacés par $\mathbf{X}$ et $\mathbf{Y}$.

Ainsi qu'elle a été introduite dans la section 3.1 pour un canal à entrée et sortie discrètes, la capacité est définie comme le maximum de l'information mutuelle de ses variables d'entrée et de sortie, par rapport à toutes les distributions de probabilité possibles des variables. Pour une dimension quelconque de l'espace des signaux, la loi demeure :

$$C = \max_{\mathbf{X},\mathbf{Y}} I\left(\mathbf{X};\mathbf{Y}\right) \tag{3.12}$$

où $I(\mathbf{X};\mathbf{Y})$ est l'information mutuelle entre $\mathbf{X}$ et $\mathbf{Y}$. Lorsque l'entrée et la sortie du canal sont des grandeurs réelles, et non plus des valeurs discrètes, les probabilités sont remplacées par des densités de probabilité et les sommes dans la relation (3.4) deviennent des intégrales. Pour des réalisations $\mathbf{x}$ et $\mathbf{y}$ des variables aléatoires $\mathbf{X}$ et $\mathbf{Y}$, on peut écrire l'information mutuelle en fonction

des probabilités de $\mathbf{x}$ et de $\mathbf{y}$:

$$I(\mathbf{X};\mathbf{Y}) = \underbrace{\int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty}}_{2N \text{ fois}} p(\mathbf{x})\, p(\mathbf{y}\,|\mathbf{x})\, \log_2 \frac{p(\mathbf{y}\,|\mathbf{x})}{p(\mathbf{y})} \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y} \tag{3.13}$$

Pour déterminer C, il faut donc maximiser (3.13) dont l'écriture est valide pour tous types d'entrées (continues, discrètes) de dimension N quelconque. D'autre part, le maximum est atteint pour des entrées équiprobables (voir section 3.1), pour lesquelles on a :

$$p(\mathbf{y}) = \frac{1}{M} \sum_{i=1}^{M} p(\mathbf{y}\,|\mathbf{x}_i)$$

où M est le nombre de symboles ou ordre de la modulation. (3.13) peut alors être écrit sous la forme :

$$C = \log_2(M) - \frac{1}{M} \sum_{i=1}^{M} \underbrace{\int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty}}_{N \text{ fois}} p(\mathbf{y}\,|\mathbf{x}_i)\, \log_2 \left(\frac{\sum_{j=1}^{M} p(\mathbf{y}\,|\mathbf{x}_j)}{p(\mathbf{y}\,|\mathbf{x}_i)} \right) \mathrm{d}\mathbf{y} \tag{3.14}$$

Selon les informations supplémentaires disponibles sur la transmission, telles que le type de bruit sur le canal, l'évanouissement éventuel, le type d'entrée et de sortie (continues, discrètes) et la modulation utilisée, (3.14) peut être particularisée.

Limite de Shannon d'un canal gaussien à entrée et sortie continues à bande limitée

Considérons simplement le cas d'un canal gaussien, avec des entrée et sortie continues. La borne de Shannon [3.1] donnant la capacité maximale C d'un tel canal est atteinte en prenant comme entrée un bruit blanc gaussien de moyenne nulle et de variance σ^2, décrit par des probabilités indépendantes sur chaque dimension, c'est-à-dire tel que :

$$p(\mathbf{x}) = \prod_{n=1}^{N} p(x_n)$$

où $\mathbf{x} = [x_1 x_2 \ldots x_N]$ est le vecteur d'entrée et $p(x_n) = N(0, \sigma^2)$. L'information mutuelle est atteinte pour des entrées équiprobables, et en notant $N_0/2$ la variance du bruit, (3.14) donne après développement :

$$C = \frac{N}{2} \log_2 \left(1 + \frac{2\sigma^2}{N_0} \right).$$

Cette formule est modifiée pour faire apparaître l'énergie moyenne E_b de chacun des bits formant le symbole $\mathbf{x}$ et par voie de conséquence le rapport signal à bruit $\frac{E_b}{N_0}$:

$$C_c = \log_2\left(1 + R\frac{E_b}{N_0}\right) \tag{3.15}$$

la capacité étant exprimée en bit/seconde/Hz. En prenant $R = 1$, cette écriture permet de borner le rapport E_b/N_0 par la limite de Shannon normalisée, de la manière représentée en figure 3.2.

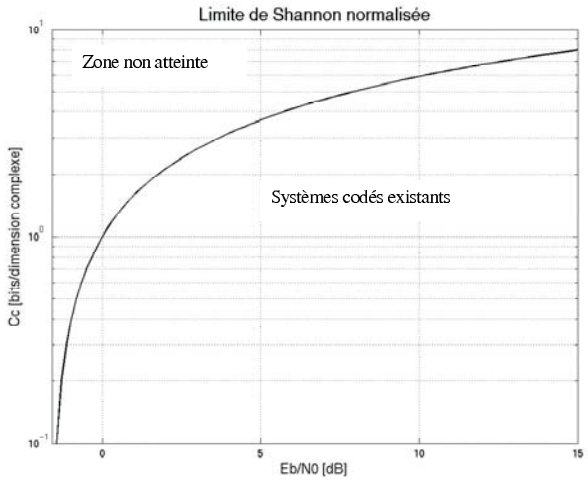


Figure 3.2 – Limite de Shannon normalisée.

Capacité d'un canal gaussien à entrée discrète

L'entrée discrète, notée $\mathbf{x} = \mathbf{x}_i\,, i = 1, \cdots, M$, est typiquement le résultat d'une modulation effectuée avant l'émission. Les entrées $\mathbf{x}_i$ appartiennent à un ensemble de M valeurs discrètes, M étant l'ordre de la modulation, et sont de dimension N, soit $\mathbf{x}_i = [x_{i1}, x_{i2}, \cdots, x_{iN}]$. La probabilité de transition du canal gaussien est :

$$p\left(\mathbf{y}\,|\mathbf{x}_i\right) = \prod_{n=1}^{N} p\left(y_n\,|x_{in}\right) = \prod_{n=1}^{N} \frac{1}{\sqrt{\pi N_0}} \exp\left(\frac{-\left(y_n - x_{in}\right)^2}{N_0}\right)$$

et nous supposons des entrées prenant les M différentes valeurs possibles de façon équiprobable. En notant $\mathbf{d}_{ij} = (\mathbf{x}_i - \mathbf{x}_j)/\sqrt{N_0}$ le vecteur de dimension N relatif à la distance entre les symboles $\mathbf{x}_i$ et $\mathbf{x}_j$, et $\mathbf{t}$ un vecteur d'intégration de dimension N, on obtient une expression simplifiée de (3.14), représentant la

capacité d'un canal gaussien à entrée discrète, pour tout type de modulation :

$$C = \log_2(M) - M^{-1}(\sqrt{\pi})^{-N} \sum_{i=1}^{M} \underbrace{\int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty}}_{N \text{ fois}} \exp\left(-|\mathbf{t}|^2\right) \log_2 \left[\sum_{j=1}^{M} \exp\left(-2\mathbf{t}\mathbf{d}_{ij} - |\mathbf{d}_{ij}|^2\right)\right] \mathrm{d}\mathbf{t} \tag{3.16}$$

C étant exprimée en bit/symbole. On remarque que $\mathbf{d}_{ij}$ augmente quand le rapport signal à bruit augmente (N_0 diminue) et la capacité tend vers $\log_2(M)$. Les différentes modulations possibles n'apparaissent que dans l'expression de $\mathbf{d}_{ij}$. Les sommes discrètes de 1 à M représentent les entrées discrètes possibles. Pour le calcul final, on exprime $\mathbf{d}_{ij}$ en fonction de E_b/N_0 selon la modulation et la capacité du canal peut être déterminée de façon numérique. La figure 3.3 donne le résultat du calcul pour quelques modulations MDP et MAQ.

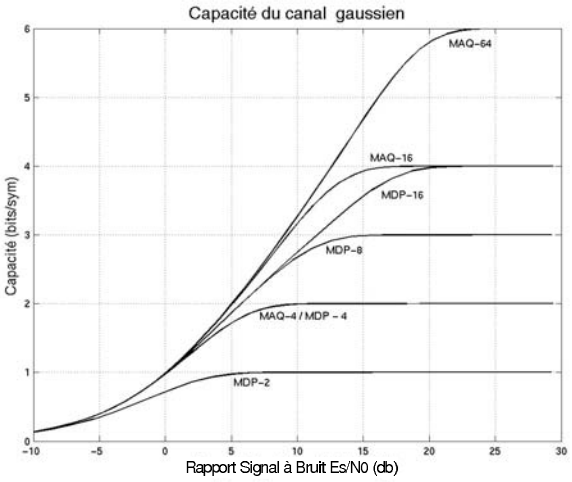


Figure 3.3 – Limite de Shannon normalisée.

Capacité du canal de Rayleigh

Soit un canal de Rayleigh dont l'atténuation est notée α. Pour des entrées discrètes et équiprobables (cas similaire au canal gaussien traité précédemment), (3.14) est toujours applicable. Deux cas se présentent, conditionnés par la connaissance ou non de l'atténuation α du canal.

Dans le cas où α n'est pas connu *a priori*, on écrit la densité de probabilité conditionnelle du canal de Rayleigh sous la forme :

$$\begin{aligned} p(\mathbf{y}\,|\mathbf{x}_i\,) &= \int_{-\infty}^{+\infty} p(\alpha) p(\mathbf{y}\,|\mathbf{x}_i, \alpha\,) \mathrm{d}\alpha \\ &= \int_{0}^{+\infty} \tfrac{1}{\sqrt{2\pi\sigma^2}} 2\alpha \exp\left(-\alpha^2\right) \exp\left(-\tfrac{|\mathbf{y}-\alpha\mathbf{x}_i|^2}{2\sigma^2}\right) \mathrm{d}\alpha \end{aligned}$$

Un développement de cette expression conduit à une écriture explicite de cette densité de probabilité conditionnelle qui se révèle indépendante de α :

$$\begin{aligned} p(\mathbf{y}\,|\mathbf{x}_i\,) &= \sqrt{\tfrac{2}{\pi}} \frac{\sigma e^{-\frac{|\mathbf{y}|^2}{2\sigma^2}}}{|\mathbf{x}_i|^2+2\sigma^2} + 2\frac{\mathbf{x}_i\mathbf{y} e^{-\frac{|\mathbf{y}|^2}{|\mathbf{x}_i|^2+2\sigma^2}}}{\left(|\mathbf{x}_i|^2+2\sigma^2\right)^{3/2}} \\ &\quad \times \left[1 - \tfrac{1}{2}\mathrm{erfc}\left(\frac{\mathbf{x}_i\mathbf{y}}{\sigma\sqrt{2\left(|\mathbf{x}_i|^2+2\sigma^2\right)}}\right)\right] \end{aligned}$$

ce qui est suffisant pour pouvoir évaluer la capacité en utilisant (3.14).

Dans le cas où l'atténuation est connue, la densité de probabilité pour une réalisation particulière de α s'écrit :

$$p\left(\mathbf{y}\,|\mathbf{x}_i\,, \alpha\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{|\mathbf{y}-\alpha\mathbf{x}_i|^2}{2\sigma^2}\right)$$

La capacité instantanée C_α pour cette réalisation particulière de α est d'abord calculée et il s'agit ensuite de moyenner C_α sur l'ensemble des réalisations de α pour obtenir la capacité du canal :

$$C_\alpha = \frac{1}{M} \sum_{i=1}^{M} \int_{-\infty}^{+\infty} p\left(\mathbf{y}\,|\mathbf{x}_i, \alpha\,\right) \log_2 \left(\frac{p\left(\mathbf{y}\,|\mathbf{x}_i, \alpha\,\right)}{p\left(\mathbf{y}\,|\alpha\,\right)}\right) \mathrm{d}\mathbf{y}$$

$$C = \int_{0}^{+\infty} C_\alpha p\left(\alpha\right) \mathrm{d}\alpha = E\left[C_\alpha\right]$$

3.3 Limites pratiques de performance

Dans les sections qui précèdent, nous avons obtenu des limites théoriques de performances qui sont soumises à certaines hypothèses non réalistes en pratique, en particulier la transmission de blocs de données de longueur infinie. Dans la grande majorité des systèmes de communications actuels, c'est une suite de blocs de données qui sont transmis, ces blocs étant d'une taille très variable suivant le système mis en oeuvre. Logiquement, une transmission par blocs de taille limitée entraîne une perte de performance par rapport à une transmission

par blocs de taille infinie, car la quantité d'information redondante contenue dans les mots de code est diminuée.

Un autre paramètre utilisé pour spécifier les performances des systèmes réels de transmissions est le taux d'erreurs paquet (TEP), qui correspond à la proportion de blocs de données faux (contenant au moins une erreur binaire après décodage).

Ce qui suit contient quelques résultats sur le canal gaussien, dans deux cas : canal gaussien à entrée binaire et sortie continue, et canal gaussien à entrée continue et sortie continue. Le cas d'entrée continue est assimilable à celui d'une modulation à nombre d'états M infini. Moins nous avons d'états pour décrire l' entrée, moins la communication est performante. En conséquence, un canal à entrée binaire donne une borne inférieure sur les performances pratiques des modulations, alors qu'un canal à entrée continue en fournit une limite supérieure.

3.3.1 Canal gaussien à entrée binaire

Les travaux initiaux sur ce canal sont dus à Gallager [3.2]. Nous notons toujours $p(\mathbf{y}|\mathbf{x})$ la probabilité de transition sur le canal, et nous considérons des messages d'information de taille k. En supposant qu'un message, choisi de façon arbitraire et équiprobable parmi 2^k, soit encodé puis transmis à travers le canal, et que l'on utilise un décodage à maximum de vraisemblance, alors le théorème du codage fournit une borne sur la probabilité d'erreur moyenne de décodage du mot de code émis. Dans [3.2], il est montré qu'il est possible de borner le TEP de la façon suivante, pour n'importe quelle valeur de la variable ρ, $0 \leqslant \rho \leqslant 1$:

$$\mathrm{TEP} \leqslant (2^k - 1)^{\rho} \sum_{\mathbf{y}} \left[\sum_{\mathbf{x}} \Pr(\mathbf{x}) p(\mathbf{y}|\mathbf{x})^{1/1+\rho} \right]^{1+\rho} \tag{3.17}$$

Dans le cas d'un canal à entrées binaires équiprobables, la probabilité de chacune des entrées est 1/2 et les vecteurs $\mathbf{x}$ et $\mathbf{y}$ peuvent être traités indépendamment en voies scalaires x et y. Considérant que (3.17) est valide pour tout ρ, afin d'obtenir la borne supérieure la plus proche possible du TEP, il faut minimiser la partie droite de (3.17) en fonction de ρ. En introduisant le rendement $R = k/n$, cela revient donc à minimiser pour $0 \leqslant \rho \leqslant 1$, l'expression :

$$\left\{ 2^{\rho R} \int_{-\infty}^{+\infty} \frac{1}{2} \left(\frac{1}{\sigma\sqrt{2\pi}} \right)^{\frac{1}{1+\rho}} \left[\exp\left(-\frac{(y-1)^2}{2\sigma^2 (1+\rho)} \right) + \exp\left(-\frac{(y+1)^2}{2\sigma^2 (1+\rho)} \right) \right] \mathrm{d}y \right\}^k$$

La valeur explicite de σ est connue pour des entrées binaires (modulations MDP-2 et MDP-4) : $\sigma = (2RE_b/N_0)^{-1/2}$. Une expression exploitable de la

borne supérieure de Gallager sur le TEP d'un canal à entrée binaire est alors :

$$e^{-k\frac{E_b}{N_0}} \min_{0\leqslant\rho\leqslant 1} \left\{ \int_0^{+\infty} 2^{\rho R+1} \frac{\exp^{-y^2}}{\sqrt{\pi}} \left(\cosh\left(\frac{y\sqrt{4RE_b/N_0}}{1+\rho} \right) \right)^{1+\rho} \mathrm{d}y \right\}^k \quad (3.18)$$

Cette expression relie le TEP, le rendement, la taille des paquets et le rapport signal à bruit E_b/N_0, pour un canal gaussien à entrée binaire. Elle fournit une borne supérieure du TEP et non pas une égalité. La formule n'est pas très bien adaptée à tous les cas de figure ; en particulier, les simulations montrent que pour un rendement proche de 1, la borne est beaucoup trop lâche et ne donne pas de résultats véritablement utiles.

Si l'on souhaite déterminer la pénalité associée à une taille de paquet donnée, on peut comparer le résultat obtenu par une évaluation de (3.18) à celui obtenu par un calcul de capacité qui considère des paquets de taille infinie.

3.3.2 Canal gaussien à entrée continue

En considérant un canal à entrée continue, nous nous plaçons dans le cas opposé au canal à entrée binaire, c'est-à-dire que nous allons obtenir une borne supérieure sur les limites pratiques de performances (toutes les modulations donnent des performances inférieures à un canal à entrée continue). Toute modulation utilisée donnera des performances bornées inférieurement par la limite obtenue par une entrée binaire et supérieurement par une entrée continue.

Les premiers résultats sont dus à Shannon [3.3] et à la méthode dite de l'enveloppement de sphère (*sphere-packing bound*) qui fournit une borne inférieure sur la probabilité d'erreur de codes aléatoires sur canal gaussien. On suppose toujours un décodage par maximum de vraisemblance. Un mot de code de longueur n est une séquence de n nombres entiers. Géométriquement, ce mot de code peut être assimilé à un point dans un espace euclidien n-dimensionnel et le bruit peut être vu comme un déplacement de ce point vers un point voisin suivant une distribution gaussienne. En notant P la puissance du signal émis, tous les mots de code sont situés sur la surface d'une sphère de rayon $\sqrt{nP}$.

En observant que nous avons un code avec 2^k points (mots de code), chacun à une distance $\sqrt{nP}$ de l'origine dans l'espace n-dimensionnel, deux points quelconques sont équidistants de l'origine, et par conséquent, la bissectrice de ces deux points (un hyperplan de dimension $n-1$) passe par l'origine. En considérant l'ensemble des 2^k points formant le code, tous les hyperplans passent par l'origine et forment des pyramides avec l'origine comme sommet. La probabilité d'erreur, après décodage, est $\Pr(e) = \frac{1}{2^k} \sum_{i=1}^{2^k} \Pr(e_i)$, où $\Pr(e_i)$ est la probabilité que le point associé au mot de code i soit déporté par le bruit en dehors de la pyramide correspondante.

Le principe de l'enveloppement de sphère de Shannon consiste en cette vision géométrique du codage. Cependant, il est très complexe de garder l'ap-

proche 'pyramidale' et la pyramide d'angle solideΩ_i, autour du mot i, est remplacée par un cône de même sommet et de même angle solide Ω_i (figure 3.4).

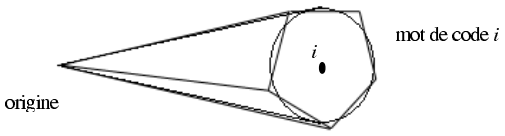


Figure 3.4 – Assimilation d'une pyramide à un cône dans l'approche dite sphere-packing de Shannon.

Il peut être montré que la probabilité que le signal reste dans le cône est supérieure à la probabilité qu'il reste dans la pyramide de même angle solide. En conséquence, la probabilité d'erreur peut être bornée de la manière suivante :

$$\Pr(e) \geqslant \frac{1}{2^k} \sum_{i=1}^{2^k} Q^*(\Omega_i) \tag{3.19}$$

en notant $Q^*(\Omega_i)$ la probabilité que le bruit transporte le point i hors du cône d'angle solide Ω_i (donc qu'une erreur de décodage soit faite sur ce point). Nous observons aussi que, si l'on considère l'ensemble des mots de code équitablement répartis sur la surface de la sphère de rayon $\sqrt{nP}$, les pyramides de décodage forment une partition de cette même sphère, et donc l'angle solide de cette sphère Ω_0 est la somme de tous les angles solides des pyramides Ω_i. Nous pouvons ainsi remplacer les angles solides Ω_i par l'angle solide moyen $\Omega_0/2^k$.

Cette progression qui conduit à une borne inférieure sur la probabilité d'erreur pour un décodage optimal de codes aléatoire sur le canal gaussien est dite de l'enveloppement de sphère car elle consiste à restreindre le codage à une sphère n-dimensionnelle et les effets du bruit à des mouvements sur cette sphère.

Des développements mathématiques donnent une forme exploitable de la borne inférieure sur la probabilité d'erreur :

$$\begin{aligned}
&\ln(\text{TEP}) \geqslant \tfrac{k}{R}\left[\ln\left(G(\theta_i, A)\sin\theta_i\right) - \tfrac{1}{2}\left(A^2 - AG(\theta_i, A)\cos\theta_i\right)\right] \\
&\theta_i \approx \arcsin\left(2^{-R}\right) \\
&G(\theta_i, A) \approx \left(A\cos\theta_i + \sqrt{A^2\cos^2\theta_i + 4}\right)/2 \\
&A = \sqrt{2RE_b/N_0}
\end{aligned} \tag{3.20}$$

Ces expressions relient sous la forme d'une borne inférieure du TEP, la taille k des paquets, le rapport signal à bruit E_b/N_0 et le rendement du code R. Pour

des valeurs de R élevées et pour des tailles de blocs k inférieures à quelques dizaines de bits, la borne inférieure est très éloignée du TEP réel.

Asymptotiquement, pour des tailles de blocs tendant vers l'infini, la borne obtenue par (3.20) tend vers la limite de Shannon pour un canal à entrée et sortie continues telle que présentée en section 3.2. De la même façon que pour le canal à entrée binaire, si l'on souhaite quantifier la perte apportée par la transmission de paquets de taille finie, il faut normaliser les valeurs obtenues par l'évaluation de (3.20) en leur retirant la limite de Shannon (3.15), la pénalité devant être nulle quand les tailles de paquets tendent vers l'infini. Les pertes liées à la transmissions de paquets de taille finie par rapport à la transmission d'un flot continu de données sont moindres dans le cas d'un canal à entrée continue que dans celui d'un canal à entrée binaire.

3.3.3 Quelques exemples de limites

La figure 3.5 ci-dessous donne un exemple de pénalités apportées par la transmission de blocs de taille k inférieure à 10000 bits, dans le cas d'une entrée continue et dans le cas d'une entrée binaire. Ces valeurs de pénalité sont à combiner avec les valeurs de capacités présentées en figure 3.3, pour obtenir les limites absolues. Comme on l'a déjà signalé, cette figure est à considérer avec précaution pour de petites valeurs de k et des TEP élevés.

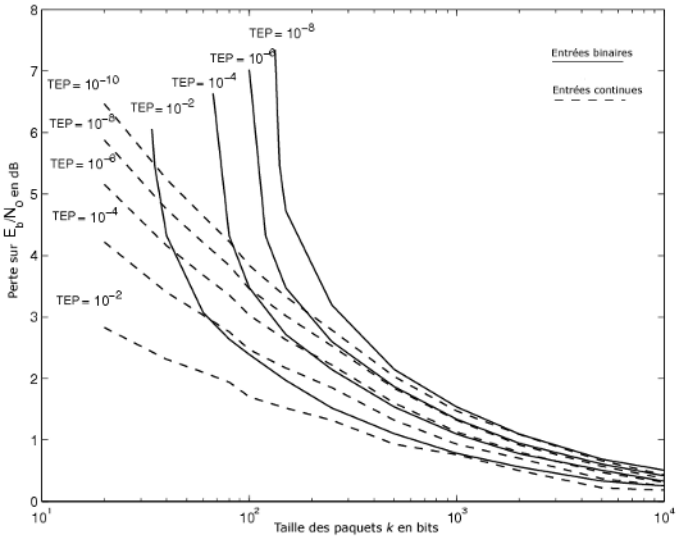


Figure 3.5 – Pénalités en $Eb/N0$ pour la transmission de paquets de taille finie pour le canal à entrée continue et le canal à entrée binaire en fonction de la taille k (bits d'information), pour un code de rendement 5/6 et différents TEP.

Les résultats obtenus concernent le canal gaussien. Il est théoriquement possible de traiter le cas des canaux avec évanouissement (Rayleigh par exemple) mais les calculs deviennent compliqués et les résultats très approximatifs.

3.4 Distances minimales requises

Dans ce qui précède, nous avons mis en évidence des limites théoriques et celles-ci ont été calculées pour le canal gaussien. Ces limites déterminent des frontières, exprimées en rapport signal à bruit, entre des canaux de transmission à la sortie desquels il est possible de corriger les erreurs et des canaux pour lesquels cette correction ne peut être envisagée. En supposant qu'existent des codes dont le décodage peut s'effectuer près de ces limites, la question se pose maintenant de savoir quelles distances minimales de Hamming (DMH) doivent posséder ces codes pour satisfaire un objectif donné de taux d'erreurs.

Nous présentons ici quelques résultats pour le canal gaussien et des modulations couramment utilisées : MDP-4 et MDP-8 ainsi que MAQ-16.

3.4.1 DMH requise avec la modulation MDP-4

Avec un décodage à maximum de vraisemblance après transmission sur canal gaussien, le taux d'erreur de paquet (TEP) possède une borne supérieure connue, dite borne de l'union :

$$\mathrm{TEP} \leqslant \frac{1}{2} \sum_{d \geqslant d_{\min}} N(d) \mathrm{erfc}\left(\sqrt{dR\frac{E_b}{N_0}}\right) \tag{3.21}$$

où $\mathrm{erfc}(x)$ désigne la fonction erreur complémentaire définie par $\mathrm{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty \exp\left(-t^2\right) \mathrm{d}t$. $d_{\min}$ est la distance minimale de Hamming du code associé à la modulation considérée, MDP-2 ou MDP-4 dans le cas présent. $N(d)$ représente les multiplicités du code considéré (voir section 1.5). Dans certains cas, ces multiplicités peuvent être déterminées de façon précise (comme par exemple les codes convolutifs simples, les codes de Reed-Solomon, les codes BCH, etc. ...), et (3.21) peut être évaluée facilement. Pour d'autres codes, en particulier les turbocodes, il n'est pas possible de déterminer facilement ces multiplicités et il faut poser quelques hypothèses réalistes afin de contourner le problème. Les hypothèses que nous adoptons pour les turbocodes ainsi que les codes LDPC sont les suivantes [3.4] :

- *Hypothèse 1 : Uniformité.* Il existe au moins un mot de code de poids[1] $d_{\min}$ ayant un bit d'information d_i égal à 1, pour toute place i de la partie systématique ($1 \leqslant i \leqslant k$).
- *Hypothèse 2 : Unicité.* Il n'y a qu'un seul mot de code de poids $d_{\min}$ tel que $d_i = 1$.

[1] Les codes étant linéaires, distance et poids ont même signification.

– *Hypothèse 3 : Non recouvrement.* Les k mots de code de poids $d_{\min}$ associés aux k bits d'information sont distincts.

En utilisant ces hypothèses et en se limitant au premier terme de la somme dans (3.21), la borne supérieure devient une approximation asymptotique (faibles TEP) :

$$\mathrm{TEP} \approx \frac{k}{2}\mathrm{erfc}\left(\sqrt{d_{\min} R \frac{E_b}{N_0}}\right) \tag{3.22}$$

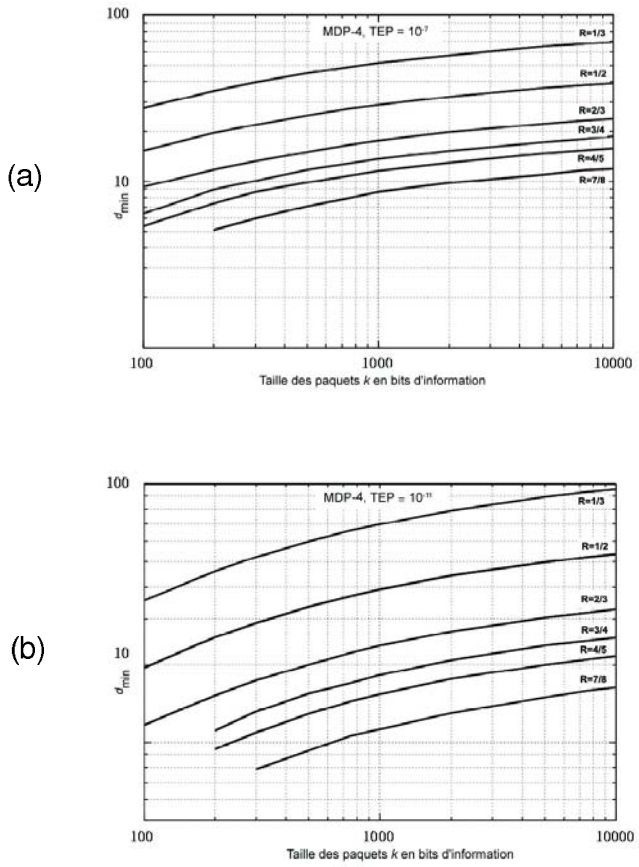


Figure 3.6 – Distances minimales requises pour une modulation MDP-4 et un canal gaussien en fonction de la taille des paquets, pour quelques rendements et taux d'erreurs de paquets (TEP).

Les trois hypothèses, prises séparément, sont plus ou moins réalistes. Les hypothèses 1 et 3 sont quelque peu pessimistes sur la quantité de mots de code à la distance minimale. L'hypothèse 2 est quant à elle légèrement optimiste. Les

trois hypothèses ensemble conviennent pour une approximation acceptable de la multiplicité, d'autant plus qu'une imprécision sur la valeur de cette multiplicité ne nuit pas à la qualité du résultat final. En effet, la distance minimale cible que l'on cherche à déterminer à partir de (3.22) y apparaît dans un argument d'exponentielle, alors que la multiplicité est un coefficient multiplicateur.

Il est alors possible de combiner (3.22) avec les résultats obtenus en section 3.3 qui fournissent les limites de rapport signal à bruit. En donnant à E_b/N_0 la valeur limite en deça de laquelle l'utilisation d'un code n'a pas d'intérêt, nous pouvons extraire de (3.22) la DMH suffisante pour atteindre un TEP à toutes les valeurs possibles de rapport signal à bruit. Sachant d'une part que (3.22) suppose un décodage idéal (maximum de vraisemblance) et d'autre part que la limite théorique n'est pas en pratique atteinte, la DMH cible peut être légèrement inférieure au résultat de cette extraction.

La figure 3.6 présente quelques résultats obtenus en utilisant cette méthode.

3.4.2 DMH requise avec la modulation MDP-8

Nous considérons ici une modulation MDP-8 sur canal gaussien mise en œuvre selon le principe de l'approche « pragmatique » et telle que présentée en figure 3.7. Cette approche consiste d'abord à encoder par paquets le flot de données pour produire des mots de code qui sont ensuite permutés par l'entrelaceur Π, avec une loi de permutation tirée au hasard. Les contenus des mots de code permutés sont ensuite agencés par groupes de 3 bits selon un codage de Gray, avant d'être modulés en MDP-8. Le démodulateur délivre les symboles reçus dont on extrait les logarithmes des rapports de vraisemblance (LRV) pour tous les bits des paquets. Enfin un entrelacement inverse et un décodage complètent la chaîne de transmission.

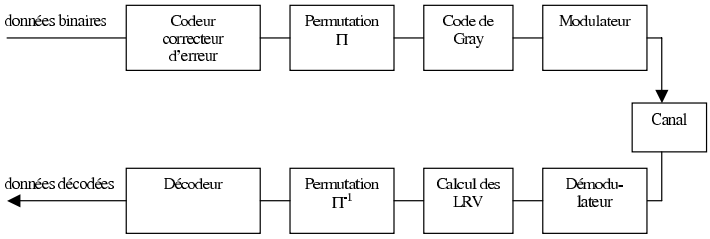


Figure 3.7 – Schéma de transmission suivant l'approche pragmatique.

La probabilité d'erreur P_e est la probabilité de décider d'un mauvais mot de code à la place du mot de code émis. Soit N_s le nombre de symboles modulés qui diffèrent entre le mot de code incorrectement décodé et le mot de code émis ; soient également $\{\phi_i\}$ et $\{\phi'_i\}$ $(1 \leqslant i \leqslant N_s)$ les séquences des phases émises pour ces symboles qui diffèrent. Il est possible d'exprimer P_e en fonction de ces

phases et du rapport signal à bruit :

$$P_e = \frac{1}{2}\mathrm{erfc}\sqrt{\frac{E_s}{N_0}\left[\sum_{i=1,N_s}\sin^2\left(\frac{\varphi'_i-\varphi_i}{2}\right)\right]} \tag{3.23}$$

où E_s est l'énergie par symbole émis et N_0 la densité monolatérale de puissance du bruit. Il n'est cependant pas possible d'exploiter (3.23) dans le cas général. Il nous faut une hypothèse supplémentaire, qui vient s'ajouter aux trois hypothèses formulées dans la section précédente et qui suppose que N_S est bien inférieure à la taille des mots de code entrelacés :

– *Hypothèse 4.* Un symbole ne contient pas plus d'un bit opposé dans le mot de code correct et dans le mot de code incorrect.

Cette hypothèse permet d'exprimer les probabilités suivantes :

$$\Pr\{\varphi_i-\varphi'_i=\pi/4\}=2/3;\quad \Pr\{\varphi_i-\varphi'_i=3\pi/4\}=1/3$$

qui signifient que deux fois sur trois en moyenne, la distance euclidienne entre les symboles concurrents est $2\sqrt{\frac{E_s}{T}}\sin(\pi/8)$ et portée, une fois sur trois, à $2\sqrt{\frac{E_s}{T}}\sin(3\pi/8)$ (figure 3.8).

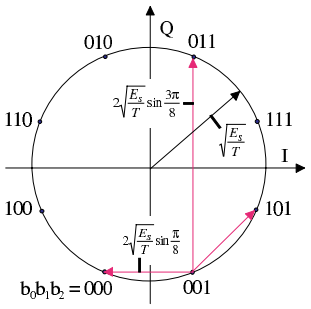


Figure 3.8 – Constellation de MDP-8 avec codage de Gray. E_s et T sont respectivement l'énergie et la durée d'un symbole.

En considérant le cas asymptotique, c'est-à-dire en posant $N_s = d_{\min}$, il vient :

$$\mathrm{TEP}_{\text{MDP-8,}\Pi\text{ aléatoire}} \approx$$
$$k\left(\tfrac{2}{3}\right)^{d_{\min}}\sum_{j=0}^{d_{\min}}\binom{d_{\min}}{j}\left(\tfrac{1}{2}\right)^{j+1}\mathrm{erfc}\sqrt{\tfrac{E_s}{N_0}\left[j\sin^2\tfrac{3\pi}{8}+(d_{\min}-j)\sin^2\tfrac{\pi}{8}\right]} \tag{3.24}$$

Cette relation permet donc d'établir une relation entre le rapport signal à bruit, la taille des blocs d'information et le TEP. De la même façon qu'en section 3.4, nous pouvons combiner ce résultat avec les limites sur le rapport signal à bruit pour obtenir les DMH cibles pour une modulation MDP-8 codée selon l'approche pragmatique. La figure 3.9 présente quelques résultats obtenus à partir de cette méthode.

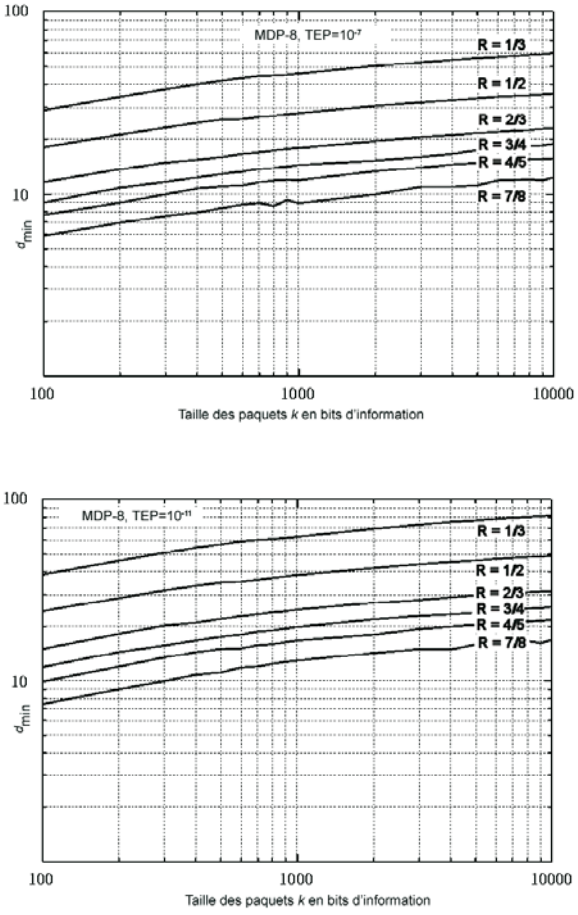


Figure 3.9 – Distances minimales requises pour une modulation MDP-8 et un canal gaussien en fonction de la taille des paquets, pour quelques rendements et taux d'erreurs de paquets (TEP).

3.4.3 DMH requise avec la modulation MAQ-16

La même méthode que précédemment, basée sur les mêmes quatre hypothèses, peut être appliquée au cas de la modulation MAQ-16 avec codage pragmatique. La constellation est une constellation de Gray classique à 16 états. Pour 75% des bits constituant les symboles, la distance euclidienne minimale est $\sqrt{2E_s/5}$ et pour les 25% restants, cette distance est $3\sqrt{2E_s/5}$. L'estimation

du TEP donne :

$$\mathrm{TEP} \approx k \left(\frac{3}{4}\right)^{d_{\min}} \sum_{j=0}^{d_{\min}} \binom{d_{\min}}{j} \left(\frac{1}{3}\right)^{j} \frac{1}{2} \mathrm{erfc} \sqrt{(8j + d_{\min}) \frac{E_s}{10N_0}} \qquad (3.25)$$

Comme pour les modulations MDP-4 et MDP-8, cette relation utilisée conjointement avec les limites en rapport signal à bruit permet d'obtenir des valeurs de DMH cibles pour la modulation MAQ-16 (figure 3.10).

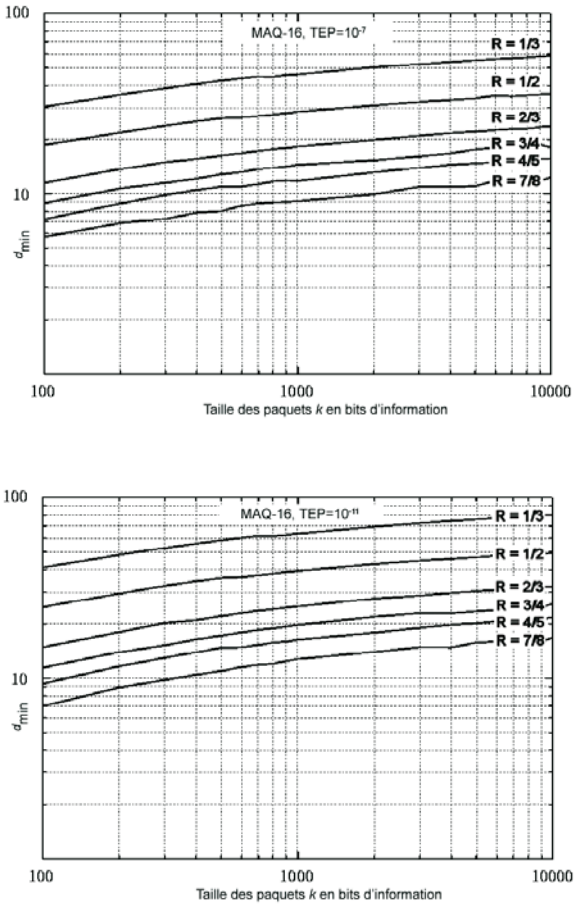


Figure 3.10 – Distances minimales requises pour une modulation MAQ-16 et un canal gaussien en fonction de la taille des paquets, pour quelques rendements et taux d'erreurs de paquets (TEP).

Quelques observations peuvent être faites à partir des résultats obtenus dans la section 3.4. Par exemple, dans le cas particulier d'une modulation MDP-4,

pour un rendement $R = 1/2$, une taille $k = 4000$ bits et un TEP de 10^{-11}, la figure 3.6 fournit une DMH cible de 50. Les codes aléatoires, à travers l'évaluation que l'on peut tirer de la borne de Gilbert-Varshamov (relation (3.9)), possèdent une distance minimale d'environ 1000. La différence est donc importante entre ce que le codage idéal (aléatoire) peut offrir et ce dont on a réellement besoin.

Un second aspect concerne la dépendance des DMH requises à la modulation utilisée, dépendance qui s'avère être minime. Ainsi, un code ayant une distance minimale suffisante pour atteindre la capacité du canal avec une modulation MDP-4 satisfera aussi le cahier des charges avec les autres modulations, à partir d'une certaine taille de message (supérieure à 1000 bits pour $R = 1/2$) ou pour des messages plus longs (au delà de 5000 bits) si le rendement est plus élevé.

3.5 Bibliographie

[3.1] C.E. Shannon, « A Mathematical Theory of Communication », *Bell Systems Technical Journal*, vol. 27, July and Oct. 1948.

[3.2] R.E. Gallager, *Information Theory and Reliable Communications*, John Wiley & Sons, 1968.

[3.3] C.E. Shannon, « Probability of Error for Optimal Codes in a Gaussian Chanel », *Bell Systems Technical Journal*, vol. 38, 1959.

[3.4] C. Berrou, E. Maury et H. Gonzalez, « Which Minimum Hamming Distance Do We Really Need ? », *Proc.* 3^{rd} *international symposium on turbo codes & related topics*, pp. 141-148, Brest, Sept. 2003.

Chapitre 4

Codes en bloc

Le codage en blocs consiste à associer à un bloc de données $\mathbf{d}$ de k symboles issus de la source d'information un bloc $\mathbf{c}$, appelé mot de code, de n symboles avec $n \geq k$. La différence $(n-k)$ représente la quantité de redondance introduite par le code. La connaissance de la règle de codage en réception permet de détecter et de corriger, sous certaines conditions, des erreurs. Le rapport k/n est appelé rendement ou taux de codage du code.

Les symboles du message d'information $\mathbf{d}$ et du mot de code $\mathbf{c}$ prennent leurs valeurs dans un corps fini F_q à q éléments, appelé corps de Galois et dont les principales propriétés sont données dans l'annexe de ce chapitre. Nous verrons que pour une majorité de codes, les symboles sont binaires et prennent leur valeur dans le corps F_2 à deux éléments (0 et 1). Ce corps est le plus petit corps de Galois.

Les opérations élémentaires d'addition et de multiplication dans le corps F_2 sont résumées dans la table 4.1.

a	b	$a+b$	ab
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Table 4.1 – Addition et multiplication dans le corps de Galois $\mathbf{F}_2$

Un code en blocs de longueur n est une application g de l'ensemble $\mathbf{F}_q^k$ vers l'ensemble F_q^n qui associe à tout bloc de données $\mathbf{d}$ un mot de code $\mathbf{c}$.

$$\begin{array}{rccc} g: & \mathbf{F}_q^k & \rightarrow & \mathbf{F}_q^n \\ & \mathbf{d} & \mapsto & \mathbf{c} = g(\mathbf{d}) \end{array}$$

L'ensemble des mots de code, au nombre de q^k, constitue généralement un sous-ensemble très réduit de $\mathbf{F}_q^n$.

Un code en blocs de paramètres (n, k), que nous noterons $C(n, k)$, est linéaire si les mots de code constituent un sous-espace vectoriel de $\mathbf{F}_q^n$, c'est-à-dire si g est une application linéaire. Une conséquence directe de la linéarité est que la somme de deux mots de code est un mot de code, et que le mot nul constitué de n symboles à zéro est toujours un mot de code.

Nous allons maintenant considérer les codes en blocs linéaires à symboles binaires. Les codes en blocs linéaires à symboles non binaires seront traités ensuite.

4.1 Les codes en blocs à symboles binaires

Dans le cas d'un code en blocs binaire, les éléments de $\mathbf{d}$ et $\mathbf{c}$ sont à valeurs dans $\mathbf{F}_2$. Comme g est une application linéaire, nous allons pouvoir décrire l'opération de codage simplement comme le résultat de la multiplication d'un vecteur de k symboles représentant les données à coder par une matrice représentative du code considéré, appelée matrice génératrice du code.

4.1.1 Matrice génératrice d'un code en blocs binaire

Notons $\mathbf{d} = [d_0 \cdots d_j \cdots d_{k-1}]$ et $\mathbf{c} = [c_0 \cdots c_j \cdots c_{n-1}]$ le bloc de données et le mot de code associé. En exprimant le vecteur $\mathbf{d}$ à partir d'une base $(\mathbf{e}_1, \ldots, \mathbf{e}_j, \ldots, \mathbf{e}_{k-1})$ de $\mathbf{F}_2^k$, nous pouvons écrire :

$$\mathbf{d} = \sum_{j=0}^{k-1} d_j \mathbf{e}_j \tag{4.1}$$

En tenant compte du fait que l'application g est linéaire, le mot $\mathbf{c}$ associé à $\mathbf{d}$ est égal à :

$$\mathbf{c} = g(\mathbf{d}) = \sum_{j=0}^{k-1} d_j g(\mathbf{e}_j) \tag{4.2}$$

En exprimant le vecteur $g(\mathbf{e}_j)$ à partir d'une base $(\mathbf{e'}_1, \cdots, \mathbf{e'}_l, \cdots, \mathbf{e'}_{n-1})$ de $\mathbf{F}_2^n$, nous obtenons :

$$g(\mathbf{e}_j) = \sum_{l=0}^{n-1} g_{jl} \mathbf{e'}_l \tag{4.3}$$

Les vecteurs $g(\mathbf{e}_j) = \mathbf{g}_j = (g_{j0} \cdots g_{jl} \cdots g_{j,n-1})$, $0 \leq j \leq k-1$ représentent les k lignes de la matrice $\mathbf{G}$ associée à l'application linéaire g.

$$\mathbf{G} = \begin{bmatrix} g_0 \\ \vdots \\ g_j \\ \vdots \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} g_{0,0} & \cdots & g_{0,l} & \cdots & g_{0,n-1} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ g_{j,0} & \cdots & g_{j,l} & \cdots & g_{j,n-1} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ g_{k-1,0} & \cdots & g_{k-1,l} & \cdots & g_{k-1,n-1} \end{bmatrix} \tag{4.4}$$

La matrice $\mathbf{G}$ à k lignes et à n colonnes, ayant pour éléments $g_{jl} \in \mathbf{F}_2$ est appelée matrice génératrice du code $C(n,k)$. Au bloc de données $\mathbf{d}$, elle associe le mot de code $\mathbf{c}$ par la relation matricielle :

$$\mathbf{c} = \mathbf{dG} \tag{4.5}$$

La matrice génératrice d'un code en blocs n'est pas unique. En effet, en permutant les vecteurs de la base $(\mathbf{e'}_1, \ldots, \mathbf{e'}_l, \ldots, \mathbf{e'}_{n-1})$ ou de la base $(\mathbf{e}_1, \ldots, \mathbf{e}_j, \ldots, \mathbf{e}_{k-1})$, on obtient une nouvelle matrice génératrice $\mathbf{G}$ dont les colonnes ou les lignes ont aussi été permutées. Bien entendu, la permutation des colonnes ou des lignes de la matrice génératrice engendre toujours le même ensemble de mots de code ; ce qui change, c'est l'association entre les mots de code et les k-uplets de données.

Signalons que les lignes de la matrice génératrice d'un code en blocs linéaire sont des mots de code indépendants, et qu'ils constituent une base du sous-espace vectoriel engendré par le code. La matrice génératrice d'un code en blocs linéaire est donc de rang k. Une conséquence directe est que toute famille composée de k mots de code indépendants peut être utilisée pour définir une matrice génératrice du code considéré.

Exemple 4.1

Considérons un code en blocs linéaire de parité $C(n,k)$ avec $k = 2$ et $n = k+1 = 3$ (pour un code de parité la somme des symboles d'un mot de code est égale à zéro). Nous avons quatre mots de code :

Bloc de données	Mot de code
00	000
01	011
10	101
11	110

Pour écrire une matrice génératrice de ce code, considérons, par exemple, la base canonique de $\mathbf{F}_2^2$:

$$\mathbf{e}_0 = \begin{bmatrix} 1 & 0 \end{bmatrix}, \mathbf{e}_1 = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

et la base canonique de $\mathbf{F}_2^3$:

$$\mathbf{e'}_0 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \mathbf{e'}_1 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}, \mathbf{e'}_2 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

Nous pouvons écrire :

$$g(\mathbf{e}_0) = [101] = 1.\mathbf{e'}_0 + 0.\mathbf{e'}_1 + 1.\mathbf{e'}_2$$
$$g(\mathbf{e}_1) = [011] = 0.\mathbf{e'}_0 + 1.\mathbf{e'}_1 + 1.\mathbf{e'}_2$$

Une matrice génératrice du code de parité est donc égale à :

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

En permutant les deux premiers vecteurs de la base canonique de $\mathbf{F}_2^3$, nous obtenons une nouvelle matrice génératrice du même code de parité :

$$\mathbf{G}' = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

Nous venons de voir, sur cet exemple, que la matrice génératrice d'un code en blocs n'est pas unique. En permutant les lignes ou les colonnes d'une matrice génératrice ou encore en ajoutant à une ligne une ou plusieurs autres lignes, ce qui revient à considérer une nouvelle base dans $\mathbf{F}_2^n$, il est toujours possible d'écrire une matrice génératrice d'un code en blocs sous la forme suivante :

$$\mathbf{G} = \begin{bmatrix} \mathbf{I}_k & \mathbf{P} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 & p_{0,1} & \cdots & p_{0,l} & \cdots & p_{0,n-k} \\ 0 & 1 & \cdots & 0 & p_{1,1} & \cdots & p_{1,l} & \cdots & p_{1,n-k} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ 0 & 0 & \ldots & 1 & p_{k-1,1} & \cdots & p_{k-1,l} & \cdots & p_{k-1,n-k} \end{bmatrix} \tag{4.6}$$

où $\mathbf{I}_k$ est la matrice identité $k \times k$ et $\mathbf{P}$ une matrice $k \times (n-k)$ utilisée pour calculer les $(n-k)$ symboles de redondance.
Ainsi écrite, la matrice génératrice $\mathbf{G}$ est sous forme réduite et engendre des mots de code de la forme :

$$\mathbf{c} = \begin{bmatrix} \mathbf{d} & \mathbf{d}P \end{bmatrix} \tag{4.7}$$

Dans chaque mot de code, les k bits de poids fort coïncident avec les symboles de données. Le code est donc systématique.

Plus généralement, le code sera dit systématique lorsqu'il existe k indices $i_0, i_1, \ldots, i_{k-1}$, tels que pour tout mot de données $\mathbf{d}$, le mot de code $\mathbf{c}$ associé vérifie la relation :

$$c_{i_q} = d_q, \qquad q = 0, 1, \cdots, k-1.$$

4.1.2 Code dual et matrice de contrôle de parité

Avant d'aborder la notion de code dual, définissons l'orthogonalité entre deux vecteurs constitués de n symboles. Deux vecteurs $\mathbf{x} = [x_0 \cdots x_j \cdots x_{n-1}]$ et $\mathbf{y} = [y_0 \cdots y_j \cdots y_{n-1}]$ sont orthogonaux ($\mathbf{x} \perp \mathbf{y}$) si leur produit scalaire noté $\langle \mathbf{x}, \mathbf{y} \rangle$ est nul.

$$\mathbf{x} \perp \mathbf{y} \Leftrightarrow \langle \mathbf{x}, \mathbf{y} \rangle = \sum_{j=0}^{n-1} x_j y_j = \mathbf{0}$$

À chaque code en blocs linéaire $C(n,k)$, on peut associer un code en blocs linéaire dual, qui vérifie que tout mot du code dual est orthogonal à tout mot du code $C(n,k)$. Le dual du code $C(n,k)$ est donc un sous-espace vectoriel de $\mathbf{F}_2^n$ constitué de 2^{n-k} mots de code de n symboles. Ce sous-espace vectoriel est l'orthogonal du sous-espace vectoriel constitué des 2^k mots du code $C(n,k)$. Il en résulte que tout mot $\mathbf{c}$ du code $C(n,k)$ est orthogonal aux lignes de la matrice génératrice $\mathbf{H}$ de son code dual.

$$\mathbf{c}\mathbf{H}^\mathbf{T} = \mathbf{0} \tag{4.8}$$

où T désigne la transposition.

Un vecteur $\mathbf{y}$ appartenant à $\mathbf{F}_2^n$ est donc un mot du code $C(n,k)$ si, et seulement si, il est orthogonal aux mots de son code dual c'est-à-dire si :

$$\mathbf{y}\mathbf{H}^\mathbf{T} = \mathbf{0}$$

Le décodeur d'un code $C(n,k)$ peut utiliser cette propriété pour vérifier que le mot reçu est un mot de code et détecter ainsi la présence d'erreurs. C'est pourquoi la matrice $\mathbf{H}$ est appelée la matrice de contrôle de parité du code $C(n,k)$.

Il est facile de voir que les matrices $\mathbf{G}$ et $\mathbf{H}$ sont orthogonales ($\mathbf{G}\mathbf{H}^\mathbf{T} = \mathbf{0}$). Ainsi, lorsque le code est systématique et que sa matrice génératrice est de la forme $\mathbf{G} = [\mathbf{I}_k \ \ \mathbf{P}]$, nous avons :

$$\mathbf{H} = [\mathbf{P}^\mathbf{T} \ \mathbf{I}_{n-k}] \tag{4.9}$$

4.1.3 Distance minimale

Avant de rappeler ce que représente la distance minimale d'un code en bloc linéaire (voir section 1.2), revenons sur la notion de distance de Hamming qui mesure la différence entre deux mots de code. La distance de Hamming, notée d_H, est égale au nombre d'emplacements où les deux mots de code possèdent des symboles différents.

On peut également définir le poids de Hamming, noté w_H, d'un mot de code comme le nombre de symboles non nuls de ce mot de code. Ainsi, la distance de Hamming entre deux mots de code est aussi égale au poids de leur somme.

Exemple 4.2
Soit deux mots $\mathbf{u} = [1101001]$ et $\mathbf{v} = [0101101]$. La distance de Hamming entre $\mathbf{u}$ et $\mathbf{v}$ est de 2. Leur somme $\mathbf{u}+\mathbf{v} = [1000100]$ a pour poids de Hamming 2.

La distance minimale $d_{\min}$ d'un code en blocs est égale à la plus petite distance de Hamming entre ses mots de code.

$$d_{\min} = \min_{\mathbf{c}\neq\mathbf{c}'} d_H(\mathbf{c},\mathbf{c}'), \qquad \forall \mathbf{c},\mathbf{c}' \in C(n,k) \tag{4.10}$$

En tenant compte du fait que la distance entre deux mots de code est égale au poids de leur somme, la distance minimale d'un code en blocs est aussi égale au poids minimal de ses mots de code non nuls.

$$d_{\min} = \min_{\mathbf{c}\neq 0, \mathbf{c}\in C(n,k)} w_H(\mathbf{c}) \tag{4.11}$$

Lorsque le nombre de mots de code est très élevé, la recherche de la distance minimale peut s'avérer laborieuse. Une première solution pour contourner cette difficulté est de déterminer la distance minimale à partir de la matrice de contrôle de parité.

Nous avons vu que $d_{\min}$ est égal au poids de Hamming minimal des mots de code non nuls. Considérons un mot de code de poids $d_{\min}$. La propriété d'orthogonalité $\mathbf{c}\mathbf{H}^{\mathrm{T}} = \mathbf{0}$ implique que la somme de $d_{\min}$ colonnes de la matrice de contrôle de parité est nulle. Ainsi $d_{\min}$ correspond au nombre minimal de colonnes de la matrice de contrôle de parité linéairement dépendantes.

Une seconde solution pour évaluer $d_{\min}$ est d'utiliser des bornes supérieures de la distance minimale. Une première borne peut s'exprimer en fonction des paramètres k et n du code. Pour un code en blocs linéaire dont la matrice génératrice est écrite sous la forme systématique $\mathbf{G} = [\mathbf{I}_k \;\; \mathbf{P}]$, les $(n-k)$ colonnes de la matrice unité $\mathbf{I}_{n-k}$ de la matrice de contrôle de parité ($\mathbf{H} = \left[\mathbf{P}^{\mathrm{T}} \;\; \mathbf{I}_{n-k}\right]$) étant linéairement indépendantes, toute colonne de $\mathbf{P}^T$ peut s'exprimer comme au plus une combinaison de ces $(n-k)$ colonnes. La distance minimale est donc bornée supérieurement par :

$$d_{\min} \leq n - k + 1 \tag{4.12}$$

Une autre borne de la distance minimale, appelée borne de Plotkin, peut être obtenue en remarquant que la distance minimale est nécessairement inférieure au poids moyen des mots de code non nuls. Si on considère l'ensemble des mots de code, il est facile de voir qu'il y a autant de symboles à 0 que de symboles à 1. Ainsi la somme des poids de tous les mots de code est égale à $n2^{k-1}$. Le nombre de mots de code non nuls étant de $2^k - 1$, la distance minimale peut être bornée par :

$$d_{\min} \leq \frac{n2^{k-1}}{2^k - 1} \tag{4.13}$$

4.1.4 Codes étendus et codes raccourcis

À partir d'un code en blocs $C(n,k)$ de distance minimale $d_{\min}$ on peut construire un code linéaire $C(n+1,k)$ en ajoutant à la fin de chaque mot de code un symbole égal à 1 (respectivement à 0) si le mot de code comprend un nombre impair (respectivement pair) de 1. Ce code est appelé code étendu et sa distance minimale est égale à $d_{\min}+1$ si $d_{\min}$ est un nombre impair.

La matrice de contrôle de parité $\mathbf{H}_e$ d'un code étendu est de la forme :

$$\mathbf{H}_e = \begin{bmatrix} & & & 0 \\ & \mathbf{H} & & \vdots \\ & & & 0 \\ 1 & \cdots & 1 & 1 \end{bmatrix}$$

où $\mathbf{H}$ est la matrice de contrôle de parité du code $C(n,k)$.

Un code en blocs systématique $C(n,k)$ de distance minimale $d_{\min}$ peut être raccourci en mettant à zéro $s < k$ symboles de données. On obtient ainsi un code linéaire systématique $C(n-s,k-s)$. Bien entendu les s symboles mis à zéro ne sont pas transmis, mais ils sont conservés pour calculer les $(n-k)$ symboles de redondance. La distance minimale d'un code raccourci est toujours supérieure ou égale à la distance du code $C(n,k)$.

4.1.5 Codes produits

Un code produit est un code à plusieurs dimensions construit à partir de codes élémentaires. Pour illustrer ces codes, considérons un code produit construit à partir de deux codes en blocs systématiques $C_1(n_1,k_1)$ et $C_2(n_2,k_2)$.

Soit un tableau à n_2 lignes et n_1 colonnes. Les k_2 premières lignes sont remplies par des mots de code de longueur n_1 produits par le code $C_1(n_1,k_1)$. Les (n_2-k_2) lignes restantes sont complétées par des symboles de redondance produits par le code $C_2(n_2,k_2)$; les k_2 symboles de chacune des n_1 colonnes étant les données d'information du code $C_2(n_2,k_2)$. On peut montrer que les (n_2-k_2) lignes du tableau sont des mots de code du code $C_1(n_1,k_1)$. Il en résulte que toutes les lignes du tableau sont des mots de code de $C_1(n_1,k_1)$ et toutes les colonnes du tableau sont des mots de code de $C_2(n_2,k_2)$.

Les paramètres du code produit à deux dimensions $C(n,k)$ de distance minimale $d_{\min}$ sont égaux au produit des paramètres des codes élémentaires.

$$n = n_1 n_2 \qquad k = k_1 k_2 \qquad d_{\min} = d^1_{\min} d^2_{\min}$$

où $d^1_{\min}$ et $d^2_{\min}$ sont respectivement les distances minimales des codes $C_1(n_1,k_1)$ et $C_2(n_2,k_2)$.

Un code produit à deux dimensions peut être vu comme une concaténation doublement série de deux codes élémentaires (chapitre 6). Un codeur C_1 est alimenté par k_2 blocs de données de longueur k_1 et produit k_2 mots de code de longueur n_1 qui sont écrits en ligne dans une matrice. La matrice est lue

par colonnes et produit n_1 blocs de symboles de longueur k_2 qui alimentent un codeur C_2. Celui-ci produit à son tour n_1 mots de code de longueur n_2. La figure 4.1 illustre la mise en œuvre d'un code produit à deux dimensions construit à partir de deux codes en blocs systématiques.

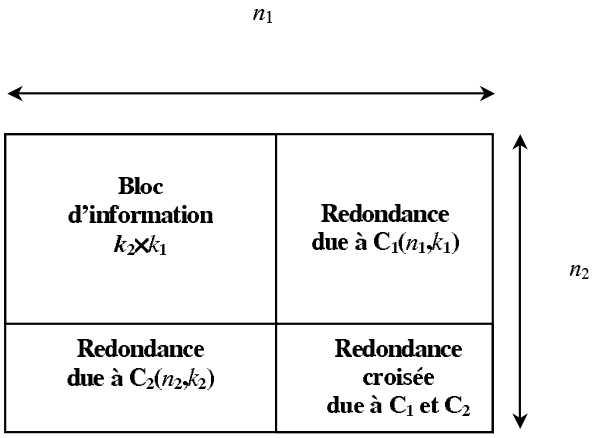


Figure 4.1 – Code produit réalisé à partir de la concaténation série de deux codes en blocs systématiques.

4.1.6 Exemples de codes en blocs binaires

Code de parité

Ce code utilise un symbole binaire de redondance ($n = k+1$) déterminé de façon à assurer la nullité modulo 2 de la somme des symboles de chaque mot de code.

$$\mathbf{c} = \left[\begin{array}{ccccccc} d_0 & d_1 & \cdots & d_{k-2} & d_{k-1} & c_{n-1} \end{array} \right] \quad \text{avec} \quad c_{n-1} = \sum_{j=0}^{k-1} d_j$$

où $\mathbf{d} = \left[\begin{array}{cccc} d_0 & d_1 & \cdots & d_{k-1} \end{array} \right]$ représente les symboles de données. La distance minimale de ce code est de 2.

Exemple 4.3
Une matrice génératrice $\mathbf{G}$ de ce code pour $n = 5$, $k = 4$ est égale à :

$$\mathbf{G} = \left[\begin{array}{ccccc} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{array} \right] = \left[\begin{array}{cc} \mathbf{I}_4 & \mathbf{P} \end{array} \right]$$

et la matrice de contrôle de parité $\mathbf{H}$ se réduit à un vecteur.

$$\mathbf{H} = \left[\begin{array}{ccccc} 1 & 1 & 1 & 1 & 1 \end{array} \right] = \left[\begin{array}{cc} \mathbf{P}^{\mathrm{T}} & \mathbf{I}_1 \end{array} \right]$$

Code à répétition

Pour ce code de paramètres $k = 1$ et $n = 2m + 1$, chaque donnée binaire issue de la source d'information est répétée un nombre impair de fois. La distance minimale de ce code est $2m + 1$. Le code à répétition $C(2m + 1, 1)$ est le code dual du code de parité $C(2m + 1, 2m)$.

Exemple 4.4

La matrice génératrice et la matrice de contrôle de parité de ce code, pour $k = 1$, $n = 5$, peuvent être les suivantes :

$$\mathbf{G} = \left[\begin{array}{ccccc} 1 & 1 & 1 & 1 & 1 \end{array} \right] = \left[\begin{array}{cc} \mathbf{I}_1 & \mathbf{P} \end{array} \right]$$

$$\mathbf{H} = \left[\begin{array}{ccccc} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{array} \right] = \left[\begin{array}{cc} \mathbf{P}^{\mathrm{T}} & \mathbf{I}_4 \end{array} \right]$$

Code de Hamming

Pour un code de Hamming les colonnes de la matrice de contrôle de parité sont les représentations binaires des nombres de 1 à n. Chaque colonne étant constituée de $m = (n - k)$ symboles binaires, les paramètres du code de Hamming sont donc :

$$n = 2^m - 1 \qquad k = 2^m - m - 1$$

Les colonnes de la matrice de contrôle de parité étant constituées par toutes les combinaisons possibles de $(n - k)$ symboles binaires sauf $(00 \cdots 0)$, la somme de deux colonnes est égale à une colonne. Le nombre minimal de colonnes linéairement dépendantes est de 3. La distance minimale d'un code de Hamming est donc égale à 3, quelle que soit la valeur des paramètres n et k.

Exemple 4.5

Soit un code de Hamming de paramètre $m = 3$. Les mots de code et les blocs de données sont alors respectivement constitués de $n = 7$ et $k = 4$ symboles binaires. La matrice de contrôle de parité peut être la suivante :

$$\mathbf{H} = \left[\begin{array}{ccccccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{array} \right] = \left[\begin{array}{cc} \mathbf{P}^{\mathrm{T}} & \mathbf{I}_3 \end{array} \right]$$

et la matrice génératrice correspondante est égale à :

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{I}_4 & \mathbf{P} \end{bmatrix}$$

Code à longueur maximale

Les colonnes de la matrice génératrice d'un code à longueur maximale sont les représentations binaires des nombres de 1 à n. Les paramètres de ce code sont donc $n = 2^m - 1, k = m$ et on peut montrer que sa distance minimale est de 2^{k-1}. Le code à longueur maximale de paramètres $n = 2^m - 1, k = m$ est le code dual du code de Hamming de paramètres $n = 2^m - 1, k = 2^m - m - 1$, c'est-à-dire que la matrice génératrice de l'un est la matrice de contrôle de parité de l'autre.

Code de Hadamard

Les mots de code d'un code de Hadamard sont constitués par les lignes d'une matrice de Hadamard et de sa matrice complémentaire. Une matrice de Hadamard comprend n lignes et n colonnes (n entier pair) dont les éléments sont des 1 et des 0. Chaque ligne diffère des autres lignes sur $n/2$ positions. La première ligne de la matrice est constituée uniquement de 0, les autres lignes possédant $n/2$ 0 et $n/2$ 1.
Pour $n = 2$, la matrice de Hadamard est de la forme :

$$\mathbf{M}_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

À partir d'une matrice $\mathbf{M}_n$ on peut générer une matrice $\mathbf{M}_{2n}$.

$$\mathbf{M}_{2n} = \begin{bmatrix} \mathbf{M}_n & \mathbf{M}_n \\ \mathbf{M}_n & \overline{\mathbf{M}}_n \end{bmatrix}$$

où $\overline{\mathbf{M}}_n$ est la matrice complémentaire de $\mathbf{M}_n$ c'est-à-dire où chaque élément à 1 (respectivement à 0) de $\mathbf{M}_n$ devient un élément à 0 (respectivement à 1) pour $\overline{\mathbf{M}}_n$.

Exemple 4.6
Si $n = 4$ $\mathbf{M}_4$ et $\overline{\mathbf{M}}_4$ sont de la forme :

$$\mathbf{M}_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad \overline{\mathbf{M}}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Les lignes de $\mathbf{M}_4$ et $\overline{\mathbf{M}}_4$ sont les mots de code d'un code de Hadamard de paramètres $n = 4$, $k = 3$ et de distance minimale égale à 2. Dans ce cas particulier le code de Hadamard est un code de parité.

Plus généralement, les lignes des matrices $\mathbf{M}_n$ et $\overline{\mathbf{M}}_n$ sont les mots d'un code de Hadamard de paramètres $n = 2^m, k = m + 1$ et de distance minimale $d_{\min} = 2^{m-1}$.

Codes de Reed-Muller

Un code de Reed-Muller (RM) d'ordre r et de paramètre m, noté $\mathrm{RM}_{r,m}$, possède des mots de code de longueur $n = 2^m$ et les blocs de données sont constitués de k symboles avec :

$$k = 1 + \left(\begin{array}{c} m \\ 1 \end{array} \right) + \cdots + \left(\begin{array}{c} m \\ r \end{array} \right), \quad \text{avec} \quad \left(\begin{array}{c} N \\ q \end{array} \right) = \frac{N!}{q!\,(N-q)!}$$

où $r < m$. La distance minimale d'un code RM est $d_{\min} = 2^{m-r}$.
La matrice génératrice d'un code RM d'ordre r se construit à partir de la matrice génératrice d'un code de RM d'ordre $r - 1$ et si $\mathbf{G}^{(r,m)}$ désigne la matrice génératrice du code de Reed-Muller d'ordre r et de paramètre m, elle s'obtient à partir de $\mathbf{G}^{(r-1,m)}$ par la relation :

$$\mathbf{G}^{(r,m)} = \left[\begin{array}{c} \mathbf{G}^{(r-1,m)} \\ \mathbf{Q}_r \end{array} \right]$$

où $\mathbf{Q}_r$ est une matrice de dimensions $\left(\begin{array}{c} m \\ r \end{array} \right) \times n$.

Par construction, $\mathbf{G}^{(0,m)}$ est un vecteur ligne de longueur n dont les éléments sont égaux à 1. La matrice $\mathbf{G}^{(1,m)}$ est obtenue en écrivant sur chaque colonne la représentation binaire de l'indice des colonnes (de 0 à $n - 1$). Par exemple pour $m = 4$, la matrice $\mathbf{G}^{(1,m)}$ est donnée par :

$$G^{(1,4)} = \left[\begin{array}{cccccccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{array} \right].$$

La matrice $\mathbf{Q}_r$ s'obtient simplement en considérant toutes les combinaisons de r lignes de $\mathbf{G}^{(1,m)}$ et en effectuant le produit composante à composante de ces vecteurs. Le résultat de cette multiplication constitue une ligne de $\mathbf{Q}_r$. Par exemple, pour la combinaison comportant les lignes de $\mathbf{G}^{(1,m)}$ d'indices i_1, i_2, …, i_r, le j-ième coefficient de la ligne ainsi obtenue est égal à $G^{(1,m)}_{i_1,j} G^{(1,m)}_{i_2,j} \cdots G^{(1,m)}_{i_r,j}$, la multiplication étant menée dans le corps $\mathbf{F}_2$. Par

exemple, pour $r = 2$, nous obtenons :

$$\mathbf{Q}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

On peut montrer que le code $\mathrm{RM}_{m-r-1,m}$ est le code dual du code $\mathrm{RM}_{r,m}$ c'est-à-dire que la matrice génératrice du code $\mathrm{RM}_{m-r-1,m}$ est la matrice de contrôle de parité du code $\mathrm{RM}_{r,m}$. Pour certaines valeurs de r et de m, la matrice génératrice du code $\mathrm{RM}_{r,m}$ est aussi sa matrice de contrôle de parité. On dit alors que le code $\mathrm{RM}_{r,m}$ est *self dual*. Le code $\mathrm{RM}_{1,3}$, par exemple, est un code *self dual*.

4.1.7 Les codes cycliques

Les codes cycliques représentent la classe la plus importante des codes en blocs linéaires. Leur mise en œuvre relativement aisée, à partir de registres à décalage et d'opérateurs logiques, en a fait des codes attractifs et très utilisés.

Définition et représentation polynomiale

Un code en blocs linéaire $C(n, k)$ est cyclique si pour tout mot de code $\mathbf{c} = \begin{bmatrix} c_0 & c_1 & \cdots & c_{n-1} \end{bmatrix}$, $\mathbf{c}_1 = \begin{bmatrix} c_{n-1} & c_0 & \cdots & c_{n-2} \end{bmatrix}$, obtenu par permutation circulaire à droite d'un symbole de $\mathbf{c}$, est aussi un mot de code. Cette définition des codes cycliques entraîne que toute permutation circulaire à droite de j symboles d'un mot de code, redonne un mot de code.
Pour les codes cycliques, on utilise une représentation polynomiale des mots de code et des blocs de données. Ainsi, au mot de code $\mathbf{c}$ on associe le polynôme $c(x)$ de degré $n-1$.

$$c(x) = c_0 + c_1 x + \cdots + c_j x^j + \cdots + c_{n-1} x^{n-1}$$

et au bloc de données $\mathbf{d}$ le polynôme $d(x)$ de degré $k-1$.

$$d(x) = d_0 + d_1 x + \cdots + d_j x^j + \cdots + d_{k-1} x^{k-1}$$

où d_j et c_j prennent leurs valeurs dans F_2.
Multiplions $c(x)$ par x,

$$xc(x) = c_0 x + c_1 x^2 + \cdots + c_j x^{j+1} + \cdots + c_{n-1} x^n$$

puis divisons $xc(x)$ par $x^n + 1$, nous obtenons :

$$xc(x) = (x^n + 1)c_{n-1} + c_1(x)$$

où $c_1(x)$ est le reste de la division de $xc(x)$ par $x^n + 1$ avec :

$$c_1(x) = c_{n-1} + c_0(x) + \cdots c_j x^{j+1} + \cdots c_{n-2} x^{n-1}$$

On peut noter que $c_1(x)$ correspond au mot de code $\mathbf{c}_1 = (c_0 \ldots c_j \ldots c_{n-2})$. En suivant la même démarche que précédemment, on obtient :

$$x^j c(x) = (x^n + 1)q(x) + c_j(x) \tag{4.14}$$

où $c_j(x)$ est aussi un mot de code obtenu par j permutations circulaires à droite des symboles de $c(x)$.
Les mots de code d'un code cyclique sont des multiples d'un polynôme $g(x)$ normalisé de degré $(n-k)$ appelé polynôme générateur.

$$g(x) = g_0 + g_1 x + \cdots + g_j x^j + \cdots + x^{n-k}$$

où g_j prend ses valeurs dans F_2.
Le polynôme générateur d'un code cyclique est un diviseur de $x^n + 1$. Il existe un polynôme $h(x)$ de degré k tel que l'équation (4.15) soit vérifiée.

$$g(x)h(x) = x^n + 1 \tag{4.15}$$

Le produit $d(x)g(x)$ est un polynôme de degré inférieur ou égal à $n-1$, il peut en conséquence représenter un mot de code. Le polynôme $d(x)$ possédant 2^k réalisations, $d(x)g(x)$ permet de générer 2^k mots de code. Notons $d_l(x)$ la l-ième réalisation de $d(x)$ et $c_l(x)$ la représentation polynomiale du mot de code associé. Nous pouvons écrire :

$$c_l(x) = d_l(x)g(x) \tag{4.16}$$

Nous allons maintenant montrer que les mots de code vérifiant la relation (4.16) satisfont aux propriétés des codes cycliques. Pour cela récrivons la relation (4.14) sous la forme :

$$c_j(x) = (x^n + 1)q(x) + x^j c(x) \tag{4.17}$$

Comme $c(x)$ représente un mot de code, il existe un polynôme $d(x)$ de degré au plus $k-1$, tel que $c(x) = d(x)g(x)$. En utilisant (4.15), on peut donc exprimer autrement (4.17) :

$$c_j(x) = g(x)[h(x)q(x) + x^j d(x)] \tag{4.18}$$

Les mots de code $c_j(x)$ sont donc des multiples du polynôme générateur, ils peuvent être générés à partir des données $d_j(x)$ en appliquant la relation (4.16).

• *Polynôme générateur du code dual de $C(n,k)$*

Le code dual d'un code en bloc cyclique est également cyclique. Le polynôme $h(x)$ de degré k peut être utilisé pour construire le code dual de $C(n,k)$. Le polynôme réciproque $\tilde{h}(x)$de $h(x)$ est défini de la manière suivante :

$$\tilde{h}(x) = x^k h(x^{-1}) = 1 + h_{k-1}x + h_{k-2}x^2 + \cdots + h_1 x^{k-1} + x^k$$

Nous pouvons écrire différemment (4.15) :

$$[x^{n-k}g(x^{-1})][x^k h(x^{-1})] = x^n + 1 \tag{4.19}$$

Le polynôme $\tilde{h}(x)$ est aussi un diviseur de x^n+1 ; il est le polynôme générateur d'un code $C^\perp = C(n, n-k)$ qui est le code dual de $C(n,k)$.

Remarque : le code de polynôme générateur $h(x)$ est équivalent au code dual $C^\perp$. La représentation vectorielle des mots engendrés par $h(x)$ correspond à la représentation vectorielle retournée des mots de code de $C^\perp$.

$$\begin{array}{lcl} C^\perp, \text{ engendré par } \tilde{h}(x) & \leftrightarrow & \text{Code engendré par } h(x) \\ \tilde{\mathbf{c}} = \left[\begin{array}{cccc} c_0 & c_1 & \cdots & c_{n-1} \end{array}\right] & \leftrightarrow & \mathbf{c} = \left[\begin{array}{cccc} c_{n-1} & \cdots & c_1 & c_0 \end{array}\right] \end{array}$$

• *Matrice génératrice d'un code cyclique*

À partir du polynôme générateur $g(x)$ il est possible de construire une matrice génératrice $\mathbf{G}$ du code $C(n,k)$. On se rappelle que les k lignes de la matrice $\mathbf{G}$ sont constituées de k mots de code linéairement indépendants. Ces k mots de code peuvent être obtenus à partir d'un ensemble de k polynômes indépendants de la forme :

$$x^j g(x) \qquad j = k-1, k-2, \ldots, 1, 0.$$

Soit $d(x)$ la représentation polynomiale d'un bloc de données quelconque. Les k mots de code engendrés par les polynômes $x^j g(x)$ ont pour expression :

$$\mathbf{c}_j(x) = x^j g(x) d(x) \qquad j = k-1, k-2, \cdots, 1, 0$$

et les k lignes de la matrice $\mathbf{G}$ ont pour éléments les coefficients binaires des monômes de $\mathbf{c}_j(x)$.

Exemple 4.7

Soit le code $C(7,4)$ de polynôme générateur $g(x) = 1 + x^2 + x^3$. Prenons pour bloc de données $d(x) = 1$. Les 4 lignes de la matrice génératrice $\mathbf{G}$ sont obtenues à partir des 4 mots de code $\mathbf{c}_j(x)$.

$$\begin{array}{l} \mathbf{c}_3(x) = x^3 + x^5 + x^6 \\ \mathbf{c}_2(x) = x^2 + x^4 + x^5 \\ \mathbf{c}_1(x) = x + x^3 + x^4 \\ \mathbf{c}_0(x) = 1 + x^2 + x^3 \end{array}$$

Une matrice génératrice du code $C(7,4)$ est égale à :

$$\mathbf{G} = \left[\begin{array}{ccccccc} 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{array}\right]$$

Code cyclique sous forme systématique

Lorsque les mots de code sont sous forme systématique, les données issues de la source d'information sont séparées des symboles de redondance. Le mot de code $c(x)$ associé au bloc de données $d(x)$ est alors de la forme :

$$c(x) = x^{n-k}d(x) + v(x) \tag{4.20}$$

où $v(x)$ est le polynôme de degré au plus égal à $n-k-1$ associé aux symboles de redondance.

En tenant compte du fait que $c(x)$ est un multiple du polynôme générateur et que l'addition et la soustraction peuvent être confondues dans F_2, alors on peut écrire :

$$x^{n-k}d(x) = q(x)g(x) + v(x)$$

$v(x)$ est donc le reste de la division de $x^{n-k}d(x)$ par le polynôme générateur $g(x)$. Le mot de code associé au bloc de données $d(x)$ est égal à $x^{n-k}d(x)$ augmenté du reste de la division de $x^{n-k}d(x)$ par le polynôme générateur.

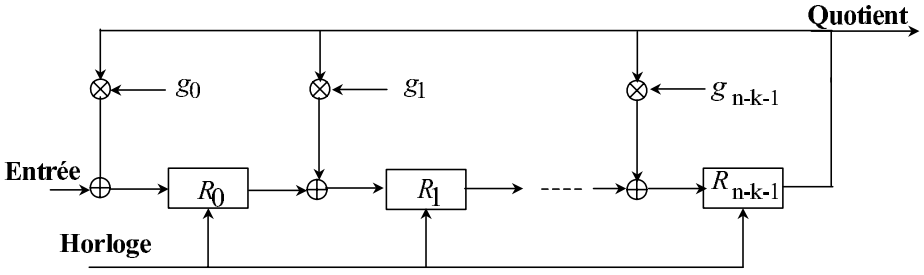


Figure 4.2 – Schéma de principe d'un circuit diviseur par $g(x)$.

Exemple 4.8

Pour illustrer le calcul d'un mot de code écrit sous forme systématique, prenons l'exemple d'un code $C(7,4)$ de polynôme générateur $g(x) = 1 + x + x^3$ et déterminons le mot de code $c(x)$ associé au message $d(x) = 1 + x^2 + x^3$, soit :

$$c(x) = x^3 d(x) + v(x)$$

Le reste de la division de $x^3 d(x)$ par $g(x) = 1 + x + x^3$ étant égal à 1, le mot de code $\mathbf{c}(x)$ associé au bloc de données $d(x)$ est :

$$c(x) = 1 + x^3 + x^5 + x^6$$

Ainsi au bloc de données $\mathbf{d}$, constitué de 4 symboles binaires d'information est associé le mot de code $\mathbf{c}$ avec :

$$\mathbf{d} = \begin{bmatrix} 1 & 0 & 1 & 1 \end{bmatrix} \rightarrow \mathbf{c} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Pour obtenir la matrice génératrice, il suffit de coder $d(x) = \quad 1, \quad x, \quad x^2, \quad x^3$. On obtient :

$d(x)$	$c(x)$
1	$1 + x + x^3$
x	$x + x^2 + x^4$
x^2	$1 + x + x^2 + x^5$
x^3	$1 + x^2 + x^6$

D'où la matrice génératrice sous une forme systématique :

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

On peut vérifier que pour $\mathbf{d} = \begin{bmatrix} 1 & 0 & 1 & 1 \end{bmatrix}$, le produit matriciel $\mathbf{dG}$ fournit bien $\mathbf{c} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$.

Mise en œuvre d'un codeur

Nous venons de voir que le codeur doit effectuer la division de $x^{n-k}d(x)$ par le polynôme générateur $g(x)$ puis ajouter le reste $v(x)$ de cette division à $x^{n-k}d(x)$. Cette opération peut être réalisée en utilisant uniquement des registres à décalage et des additionneurs dans le corps F_2. L'opération la plus difficile à réaliser étant la division de $x^{n-k}d(x)$ par $g(x)$, examinons d'abord le schéma de principe d'un diviseur par $g(x)$ représenté sur la figure 4.2. Le circuit diviseur est réalisé à partir d'un registre à décalage à $(n-k)$ mémoires notées R_i et autant d'additionneurs. Le registre à décalage est initialisé à zéro et les k coefficients du polynôme $x^{n-k}d(x)$ sont introduits dans le circuit diviseur au rythme d'une horloge. Après k impulsions d'horloge, on peut vérifier que le résultat de la division est disponible en sortie du circuit diviseur, ainsi que le reste $v(x)$ qui se trouve dans les mémoires du registre à décalage.

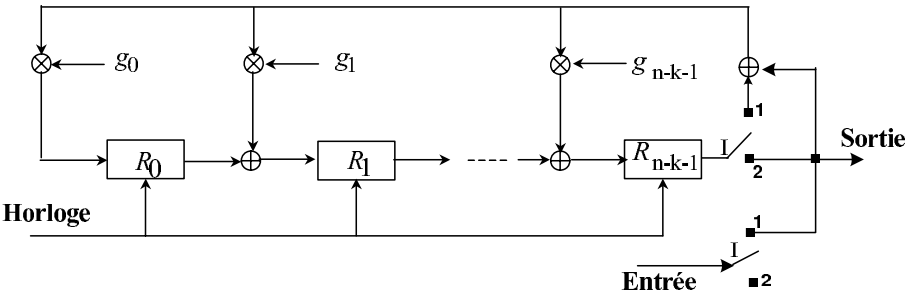


Figure 4.3 – Schéma de principe d'un codeur pour un code cyclique.

Le schéma de principe du codeur représenté dans la figure 4.3, utilise le circuit diviseur de la figure 4.2. La multiplication de $d(x)$ par x^{n-k}, correspondant

à un simple décalage, est réalisée en introduisant le polynôme $d(x)$ en sortie du registre à décalage du diviseur.

Les k données issues de la source d'information sont introduites au rythme d'une horloge dans le codeur (interrupteur I en position 1) qui réalise la division de $x^{n-k}d(x)$ par $g(x)$. Simultanément, les k données issues de la source d'information sont aussi émises en ligne. Cette opération terminée, le reste $v(x)$ de la division se trouve dans les $(n-k)$ mémoires du registre à décalage. L'interrupteur I passe alors en position 2, et les $(n-k)$ symboles de redondance sont envoyés vers la sortie du codeur.

Les codes BCH

Les codes de Bose-Chaudhuri-Hocquenghem, appelés codes BCH permettent de construire de manière systématique des codes cycliques corrigeant au moins t erreurs dans un bloc de n symboles c'est-à-dire, des codes dont la distance minimale $d_{\min}$ est au moins égale à $2t+1$.

Pour construire un code BCH, on se fixe t ou de manière équivalente d, appelée distance construite du code et on détermine son polynôme générateur $g(x)$. Le code obtenu possède une distance minimale $d_{\min}$ qui est toujours supérieure ou égale à la distance construite.

Code BCH primitif

Le polynôme générateur $g(x)$ d'un code BCH primitif construit sur un corps de Galois F_q à $q=2^m$ éléments, de distance construite d possède $(d-1)$ racines de la forme : $\alpha^l, \cdots, \alpha^{l+j}, \cdots, \alpha^{l+d-2}$, où $2t+1$ est un élément primitif du corps de Galois F_q et l un entier. Le code BCH est dit primitif car les racines de son polynôme générateur sont des puissances de α, élément primitif de F_q. Nous verrons ultérieurement qu'il est possible de construire des codes BCH non primitifs.

Généralement le paramètre l est fixé à 0 ou à 1 et on montre que pour un code BCH primitif l'exposant $(l+d-2)$ de la racine α^{j+d-2} doit être pair. Lorsque $l=0$, la distance construite est donc nécessairement paire, c'est-à-dire égale à $2t+2$ pour un code corrigeant t erreurs. Lorsque $l=1$, la distance construite est impaire, soit égale à $2t+1$ pour un code corrigeant t erreurs.

- *Code BCH primitif avec $l=1$*
  Le polynôme générateur d'un code BCH primitif corrigeant au moins t erreurs (distance construite $2t+1$) a donc pour racines $\alpha, \cdots, \alpha^j, \cdots, \alpha^{2t}$. On montre que le polynôme générateur $g(x)$ d'un code BCH primitif est égal à :

  $$g(x) = \text{P.P.C.M.}\,(m_\alpha(x), \cdots, m_{\alpha^i}(x), \cdots, m_{\alpha^{2t}}(x))$$

  où $m_{\alpha^i}(x)$ est le polynôme minimal à coefficients dans le corps F_2 associé à α^j et P.P.C.M. désigne le Plus Petit Commun Multiple.

Il est détaillé dans l'annexe 2 qu'un polynôme à coefficients dans F_2 ayant α^j comme racine possède aussi α^{2j} comme racine. Ainsi, les polynômes minimaux $m_{\alpha^i}(x)$ et $m_{\alpha^{2i}}(x)$ ont même racines. Cette remarque permet de simplifier l'écriture du polynôme générateur $g(x)$.

$$g(x) = \text{P.P.C.M.}\,(m_\alpha(x), m_{\alpha^3}(x), \cdots, m_{\alpha^{2t-1}}(x)) \tag{4.21}$$

Le degré d'un polynôme minimal étant inférieur ou égal à m, le degré $(n-k)$ du polynôme générateur d'un code BCH primitif corrigeant au moins t erreurs, est donc inférieur ou égal à mt. En effet, $g(x)$ est au plus égal au produit de t polynômes de degré inférieur ou égal à m.
Les paramètres d'un code BCH primitif construit sur un corps de Galois F_q de distance construite $d = 2t+1$ sont donc les suivants :

$$n = 2^m - 1; k \geq 2^m - 1 - mt; d_{\min} \geq 2t+1$$

Lorsque $t = 1$ un code BCH primitif est un code de Hamming. Le polynôme générateur d'un code de Hamming, égal à $m_\alpha(x)$, est donc un polynôme primitif.

Exemple 4.9
Déterminons le polynôme générateur d'un code BCH ayant pour paramètres $m = 4$ et $n = 15$, $t = 2$ et $l = 1$. Pour cela, nous allons utiliser un corps de Galois à $q = 2^4$ éléments construit à partir d'un polynôme primitif de degré $m = 4(\alpha^4 + \alpha + 1)$. Les éléments de ce corps sont donnés dans l'annexe 2.
Nous devons d'abord déterminer les polynômes minimaux $m_\alpha(x)$ et $m_{\alpha^3}(x)$ associés respectivement aux éléments α et α^3 du corps F_{16}.
Nous avons vu dans l'annexe 2 que si α est racine du polynôme $m_\alpha(x)$ alors $\alpha^2, \alpha^4, \alpha^8$ sont aussi des racines de ce polynôme (l'élévation de α aux puissances 16, 32 etc... redonne, modulo $\alpha^4 + \alpha + 1$, les éléments $\alpha, \alpha^2, \alpha^4, \alpha^8$). Nous pouvons donc écrire :

$$m_\alpha(x) = (x+\alpha)(x+\alpha^2)(x+\alpha^4)(x+\alpha^8)$$

En développant l'expression de $m_\alpha(x)$ on obtient :

$$m_\alpha(x) = [x^2 + x(\alpha^2+\alpha) + \alpha^3][x^2 + x(\alpha^8+\alpha^4) + \alpha^{12}]$$

En utilisant les représentations binaires des éléments du corps F_{16}, on peut montrer que $\alpha^2 + \alpha = \alpha^5$ et que $\alpha^4 + \alpha^8 = \alpha^5$ (on rappelle que les additions binaires sont faites modulo 2 dans le corps de Galois). On poursuit alors le développement de $m_\alpha(x)$ et il vient finalement :

$$m_\alpha(x) = x^4 + x + 1$$

Pour le calcul de $m_{\alpha^3}(x)$, les racines à prendre en compte sont $\alpha^3, \alpha^6, \alpha^{12}, \alpha^{24} = \alpha^9$ ($\alpha^{15} = 1$), les autres puissances de α^3 ($\alpha^{48}, \alpha^{96}, \cdots$) redonnent les

racines précédentes. Le polynôme minimal $m_{\alpha^3}(x)$ est donc égal à :

$$m_{\alpha^3}(x) = (x+\alpha^3)(x+\alpha^6)(x+\alpha^{12})(x+\alpha^9)$$

ce qui après développement et simplification donne :

$$m_{\alpha^3}(x) = x^4 + x^3 + x^2 + x + 1$$

Le P.P.C.M. des polynômes $m_\alpha(x)$ et $m_{\alpha^3}(x)$ est évidemment égal au produit de ces deux polynômes puisqu'ils sont irréductibles et ainsi, le polynôme générateur est égal à :

$$g(x) = (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)$$

En développant, on obtient :

$$g(x) = x^8 + x^7 + x^6 + x^4 + 1$$

Finalement les paramètres de ce code BCH sont :

$$m = 4; n = 15; n - k = 8; k = 7; t = 2$$

Les valeurs numériques des paramètres (n, k, t) des principaux codes BCH ainsi que les polynômes générateurs associés ont été tabulés et peuvent être trouvés dans [4.1]. À titre d'exemple, nous donnons dans la table 4.2 les paramètres et les polynômes générateurs, exprimés en octal, de quelques codes BCH de pouvoir de correction $t = 1$ (codes de Hamming).

n	**k**	**t**	**g(x)**
7	4	1	13
15	11	1	23
31	26	1	45
63	57	1	103
127	120	1	211
255	247	1	435
511	502	1	1021
1023	1013	1	2011
2047	2036	1	4005
4095	4083	1	10123

Table 4.2 – Paramètres de quelques codes de Hamming.

Note : $g(x) = 13$ en octal donne 1011 en binaire soit $g(x) = x^3 + x + 1$

- *Code BCH primitif avec $l = 0$*
  Le polynôme générateur d'un code BCH primitif corrigeant au moins t erreurs (distance construite $d = 2t + 2$) possède $(2t + 1)$ racines de la forme : $\alpha^0, \alpha^1, \cdots, \alpha^j, \cdots, \alpha^{2t}$; soit une racine de plus (α^0) que lorsque

$l = 1$. En tenant compte du fait que les polynômes minimaux $m_{\alpha^j}(x)$ et $m_{\alpha^{2j}}(x)$ ont même racines, le polynôme générateur $g(x)$ est égal à :

$$g(x) = \text{P.P.C.M.}(m_{\alpha^0}(x), m_{\alpha^1}(x), m_{\alpha^3}(x), \cdots, m_{\alpha^{2t-1}}(x))$$

1. Code de parité

   Considérons un code BCH avec $l = 0$ et $t = 0$. Son polynôme générateur, $g(x) = (x + 1)$ possède une racine unique $\alpha^0 = 1$. Ce code utilise un seul symbole de redondance et les mots $c(x)$ de ce code vérifient la condition :

   $$c(\alpha^0) = c(1) = 0$$

   Ce code qui est cyclique puisque $(x + 1)$ divise $(x^n + 1)$ est un code de parité de paramètres $n = k + 1, k, t = 0$. Ainsi, à chaque fois que l'on construit un code BCH en choisissant $l = 0$, on introduit dans le polynôme générateur un terme en $(x+1)$ et les mots de code sont de poids pair.

2. Code *CRC* (*Cycle Redundancy Code*)

   Un autre exemple de code BCH pour lequel $l = 0$ est le code *CRC* utilisé pour la détection d'erreurs. Un code *CRC* possède une distance construite de 4 ($t = 1$) et son polynôme générateur, d'après ce qui précède, est donc égal à :

   $$g(x) = (x + 1)m_\alpha(x)$$

   α étant un élément primitif, $m_\alpha(x)$ est un polynôme primitif et ainsi le polynôme générateur d'un code *CRC* est égal au produit de $(x+1)$ par le polynôme générateur d'un code de Hamming.

   $$g_{\text{CRC}}(x) = (x + 1)g_{\text{Hamming}}(x)$$

   Les paramètres d'un code *CRC* sont donc :

   $$n = 2^m - 1; (n - k) = m + 1; k = 2^m - m - 2$$

   Les codes *CRC* les plus utilisés ont pour paramètres $m = 12$, 16, 32 et leurs polynômes générateurs sont donnés, en octal, dans la table 4.3.

   Note : $g(x) = 14017$ en octal correspond à 1 100 000 001 111 en binaire soit :

   $$g(x) = x^{12} + x^{11} + x^3 + x^2 + x + 1$$

Code	m	$g(x)$
CRC-12	12	14017
CRC-16	16	300005
CRC-CCITT	16	210041
CRC-32	32	40460216667

Table 4.3 – Polynômes générateurs de quelques codes *CRC*.

Code BCH non primitif

Le polynôme générateur d'un code BCH non primitif (avec $l = 1$) corrigeant au moins t erreurs (distante construite $d = 2t + 1$) possède $2t$ racines de la forme : $\beta, \beta^2, \beta^3, \ldots, \beta^{2t}$ où β est un élément non primitif d'un corps de Galois F_q. En tenant compte du fait que les polynômes minimaux $m_{\beta^j}(x)$ et $m_{\beta^{2j}}(x)$ ont les mêmes racines, le polynôme générateur d'un code BCH non primitif est égal à :

$$g(x) = \text{P.P.C.M.}(m_\beta(x), m_{\beta^3}(x).....m_{\beta^{2t-1}}(x))$$

On peut montrer que la longueur n des mots d'un code BCH non primitif n'est plus de la forme $2^m - 1$ mais est égale à p, où p est l'exposant de β tel que $\beta^p = 1$ (p est l'ordre de β). Un corps de Galois F_q possède des éléments non primitifs si $2^m - 1$ n'est pas premier. Les éléments non primitifs sont alors de la forme $\beta = \alpha^\lambda$ où λ est un diviseur de $2^m - 1$ et α est un élément primitif du corps.

Exemple 4.10

Soit un corps de Galois F_q avec $m = 6$ et $q = 64$. La quantité $2^m - 1 = 63$ n'est pas égale à un nombre premier, elle est divisible par 3, 7, 9, 21 et 63. Les éléments non primitifs de ce corps sont donc $\alpha^3, \alpha^7, \alpha^9, \alpha^{21}, \alpha^{63} = 1$. Construisons par exemple un code BCH non primitif ayant un pouvoir de correction au moins égal à $t = 2$ sur le corps F_{64} et prenons pour élément non primitif $\beta = \alpha^3$. Nous avons deux polynômes minimaux à calculer $m_\beta(x)$ et $m_{\beta^3}(x)$. En tenant compte du fait que $\beta^{21} = \alpha^{63} = 1$, les racines de ces polynômes sont :

$$m_\beta(x) : \text{racines } \beta, \beta^2, \beta^4, \beta^8, \beta^{16}, \beta^{32} = \beta^{11}$$
$$m_{\beta^3}(x) : \text{racines } \beta^3, \beta^6, \beta^{12}$$

Le polynôme générateur de ce code est égal à :

$$g(x) = m_\beta(x) m_{\beta^3}(x)$$

ce qui après développement et simplification donne :

$$g(x) = x^9 + x^8 + x^7 + x^5 + x^4 + x + 1$$

Les paramètres de ce code BCH non primitif sont :

$$n = 21; (n - k) = 9; k = 12$$

- *Code de Golay*
  Parmi les codes BCH non primitifs, le plus connu est certainement le code de Golay construit sur un corps de Galois F_q avec $m = 11, q = 2048$. En remarquant que $2^m - 1 = 2047 = 23 \times 89$, l'élément non primitif utilisé pour construire un code de Golay est $\beta = \alpha^{89}$. Le calcul du polynôme générateur de ce code construit sur le corps $\mathbf{F}_{2048}$ conduit à l'expression suivante :
  $$g(x) = x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1$$
  On peut montrer que la distance minimale $d_{\min}$ d'un code de Golay est de 7 et ainsi, son pouvoir de correction est de 3 erreurs dans un bloc de 23 symboles binaires ($\beta^{23} = \alpha^{2047} = 1$). Les paramètres d'un code de Golay sont donc :
  $$n = 23; (n-k) = 11; k = 12; t = 3$$
  Notons que le polynôme réciproque de $g(x)$, égal à $\tilde{g}(x) = x^{11} g(x^{-1})$ permet également d'engendrer un code de Golay.
  $$\tilde{g}(x = x^{11} + x^{10} + x^6 + x^5 + x^4 + x^2 + 1$$

4.2 Les codes en blocs à symboles non binaires

4.2.1 Les codes de Reed-Solomon

Les codes de Reed-Solomon, désignés par le sigle RS, sont les codes à symboles non binaires les plus connus et surtout les plus utilisés. Pour les codes à symboles non binaires les coefficients c_j des mots de code et d_j des blocs de données prennent leur valeur dans un corps de Galois F_q à $q = 2^m$ éléments. Ainsi, chaque symbole de ces codes peut être codé sur m symboles binaires. Les codes de Reed-Solomon étant des codes cycliques, ils sont engendrés par un polynôme générateur $g(x)$ diviseur de $x^n + 1$ dont les coefficients g_j $j = 0, 1, \cdots, n-k-1$ prennent également leur valeur dans le corps de Galois F_q.

Le polynôme générateur d'un code de Reed-Solomon, de distance construite d possède $(d-1)$ racines $\alpha^l, \cdots, \alpha^{l+j}, \cdots, \alpha^{l+d-2}$ où α est un élément primitif du corps de Galois F_q. Il a donc pour expression :

$$g(x) = \text{P.P.C.M.}(m_{\alpha^l}(x), \cdots, m_{\alpha^{l+j}}(x), \cdots, m_{\alpha^{l+d-2}}(x))$$

où $m_{\alpha^{l+j}}$ est le polynôme minimal associé à l'élément α^{l+j} du corps F_q.

En utilisant les résultats de l'annexe 2 sur les polynômes minimaux à coefficients dans F_q, le polynôme minimal $m_{\alpha^{l+j}}$ a pour unique racine α^{l+j}.

$$m_{\alpha^{j+i}}(x) = (x + \alpha^{j+i})$$

Le polynôme générateur d'un code de Reed-Solomon est donc de la forme :

$$g(x) = (x + \alpha^j)(x + \alpha^{j+1})...(x + \alpha^{j+i})...(x + \alpha^{j+d-2})$$

En général, le paramètre j est fixé à 0 ou à 1 comme pour les codes BCH binaires. Le polynôme générateur d'un code de Reed Solomon, de degré $(n-k)$ possède $(d-1)$ racines soit $n-k=d-1$. Sa distance construite est donc égale à :

$$d = n - k + 1$$

Pour un code en bloc $C(n,k)$ la distance minimale $d_{\min}$ étant inférieure ou égale à $n-k+1$, la distance minimale d'un code de Reed-Solomon est donc toujours égale à sa distance construite. Un code dont la distance minimale est égale à $n-k+1$ est appelé un code à distance maximale.
Les paramètres d'un code de Reed-Solomon corrigeant t erreurs dans un bloc de n symboles q-aires sont donc :

$$n = q-1; \quad n-k = d_{\min} - 1 = 2t; \quad k = n-2t$$

Exemple 4.11

Déterminons le polynôme générateur d'un code de Reed-Solomon construit à partir d'un corps de Galois à 16 éléments ayant un pouvoir de correction de $t=2$ erreurs. La distance minimale de ce code est donc de $d_{\min}=5$. En prenant par exemple $l=1$, le polynôme générateur de ce code est donc de la forme :

$$g(x) = (x+\alpha)(x+\alpha^2)(x+\alpha^3)(x+\alpha^4)$$

En développant l'expression précédente, on obtient :

$$g(x) = \lfloor x^2 + x(\alpha+\alpha^2) + \alpha^3 \rfloor \lfloor x^2 + x(\alpha^3+\alpha^4) + \alpha^7 \rfloor$$

En utilisant les représentations binaires des éléments du corps F_{16} (annexe 2), le polynôme $g(x)$ après développement et simplification est égal à :

$$g(x) = x^4 + \alpha^3 x^3 + \alpha^6 x^2 + \alpha^3 x + \alpha^{10}$$

4.2.2 Mise en oeuvre du codeur

Le schéma de principe d'un codeur pour code de Reed-Solomon est tout à fait analogue à celui d'un codeur pour code cyclique à symboles binaires. Simplement le codeur doit désormais réaliser des multiplications entre symboles q-aires et mémoriser des symboles q-aires.

À titre d'exemple nous avons représenté dans la figure 4.4, le schéma de principe du codeur pour le code de Reed-Solomon traité dans l'exemple précédent.

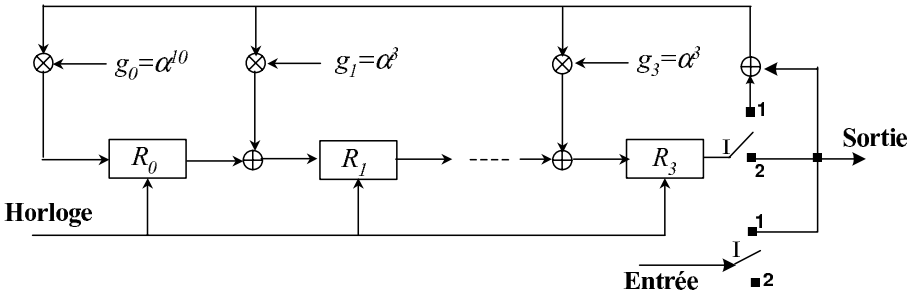


Figure 4.4 – Schéma de principe du codeur du code de RS(15,11).

4.3 Décodage et performances des codes à symboles binaires

4.3.1 Détection d'erreur

En considérant un canal de transmission binaire symétrique, le décodeur reçoit des symboles binaires supposés en synchronisme parfait avec le codeur. Cela signifie que le découpage en mots de n symboles à l'entrée du décodeur correspond à celui utilisé par le codeur. Ainsi, en l'absence d'erreurs le décodeur voit à son entrée des mots de code.

Supposons que le mot de code $\mathbf{c}$ soit émis par le codeur et soit $\mathbf{r}$ le mot de n symboles reçu à l'entrée du décodeur. Le mot $\mathbf{r}$ peut toujours s'écrire sous la forme :

$$\mathbf{r} = \mathbf{c} + \mathbf{e}$$

où $\mathbf{e}$ est un mot dont les symboles non nuls représentent les erreurs. Un symbole non nul de $\mathbf{e}$ indique la présence d'une erreur sur la position correspondante de $\mathbf{c}$.

La détection des erreurs se fait en utilisant la propriété d'orthogonalité de la matrice de contrôle de parité avec les mots de code et en calculant la quantité $\mathbf{s}$ appelée syndrome d'erreur.

$$\mathbf{s} = \mathbf{r}\mathbf{H}^{\mathbf{T}} = (\mathbf{c} + \mathbf{e})\mathbf{H}^{\mathbf{T}} = \mathbf{e}\mathbf{H}^{\mathbf{T}}$$

Le syndrome $\mathbf{s}$ est nul si, et seulement si, $\mathbf{r}$ est un mot de code. Un syndrome non nul implique la présence d'erreurs. Notons toutefois qu'un syndrome nul ne signifie pas nécessairement l'absence d'erreurs car le mot $\mathbf{r}$ peut appartenir aux mots du code tout en étant différent de $\mathbf{c}$. Il suffit pour cela que le mot $\mathbf{e}$ soit un mot de code. En effet, pour un code en blocs linéaire, la somme de deux mots de code est encore un mot de code.

Retenons finalement que pour tout code en blocs linéaire, il existe des configurations d'erreurs non détectables.

Pouvoir de détection

Soit $\mathbf{c}_j$ le mot de code émis et $\mathbf{c}_l$ son plus proche voisin, nous avons l'inégalité suivante :

$$d_H(\mathbf{c}_j, \mathbf{c}_l) \geqslant d_{\min}$$

En introduisant le mot reçu $\mathbf{r}$, nous pouvons écrire :

$$d_{\min} \leqslant d_H(\mathbf{c}_j, \mathbf{c}_l) \leqslant d_H(\mathbf{c}_j, \mathbf{r}) + d_H(\mathbf{c}_l, \mathbf{r})$$

et ainsi toutes les erreurs pourront être détectées si la distance de Hamming entre $\mathbf{r}$ et $\mathbf{c}_l$ est supérieure ou égale à 1, c'est-à-dire si $\mathbf{r}$ ne se confond pas avec $\mathbf{c}_l$.

Le pouvoir de détection d'un code $C(n, k)$ de distance minimale $d_{\min}$ est donc égal à $d_{\min} - 1$.

Probabilité de non détection d'erreur

En considérant un code en blocs $C(n, k)$ et un canal binaire symétrique de probabilité d'erreur p, la probabilité de non détection des erreurs P_{nd} est égale à :

$$P_{nd} = \sum_{j=d_{\min}}^{n} A_j p^j (1-p)^{n-j} \tag{4.22}$$

où A_j est le nombre de mots de code de poids j.

En se plaçant dans l'hypothèse d'une transmission complètement dégradée c'est-à-dire d'une probabilité d'erreur sur le canal de $p = 1/2$, et en tenant compte du fait que pour tout code en blocs on a :

$$\sum_{j=d_{\min}}^{n} A_j = 2^k - 1$$

(le -1 dans l'expression ci-dessus correspond au mot de code nul), la probabilité P_{nd} est égale à :

$$P_{nd} = \frac{2^k - 1}{2^n} \cong 2^{-(n-k)}$$

La détection des erreurs reste donc efficace quelle que soit la probabilité d'erreur sur le canal de transmission si le nombre de symboles de redondance $(n - k)$ est suffisamment grand. La détection d'erreurs est donc peu sensible à la statistique des erreurs.

Lorsque des symboles erronés sont détectés, le destinataire demande généralement à la source de les émettre de nouveau. Pour transmettre cette demande de réémission, il est alors nécessaire de disposer d'une liaison destinataire source, appelée voie de retour. Le débit sur la voie de retour étant faible (a priori les demandes de retransmission sont courtes et peu nombreuses), on peut toujours s'arranger pour que la probabilité d'erreur sur cette voie soit

très inférieure à la probabilité d'erreur sur le canal de transmission. Ainsi, les performances d'un système de transmission utilisant la détection d'erreurs et la répétition ne dépendent que très faiblement de la voie de retour.

En cas de détection d'erreurs, l'émission de la source peut être interrompue pour permettre la retransmission de l'information erronée. Le débit d'information n'est donc pas constant, ce qui peut, dans certain cas, poser des problèmes.

4.3.2 Correction des erreurs

La correction d'erreurs consiste à rechercher le mot de code émis $\mathbf{c}$ c'est-à-dire à réaliser un décodage de $\mathbf{c}$ à partir du mot reçu $\mathbf{r}$. Deux stratégies sont possibles. Une première correspond à un mot reçu $\mathbf{r}$ à l'entrée du décodeur constitué de symboles binaires (cas d'un canal binaire symétrique) et une seconde, à un mot reçu $\mathbf{r}$ constitué de symboles analogiques (cas d'un canal gaussien). Dans le premier cas, on parle de décodage à entrée ferme alors que dans le second cas on parle de décodage à entrée pondérée. Nous allons maintenant examiner ces deux types de décodage dont il a déjà été question dans le chapitre 1.

Décodage ferme

• *Décodage à maximum de vraisemblance a posteriori*

Pour le décodage ferme le mot reçu $\mathbf{r}$ est de la forme :

$$\mathbf{r} = \mathbf{c} + \mathbf{e}$$

où $\mathbf{c}$ et $\mathbf{e}$ sont des mots à symboles binaires.

Le décodage selon le maximum de vraisemblance *a posteriori* consiste à rechercher le mot de code $\hat{\mathbf{c}}$ tel que :

$$\Pr\{\hat{\mathbf{c}}/\mathbf{r}\} > \Pr\{\mathbf{c}_i/\mathbf{r}\} \quad \forall\, \mathbf{c}_i \neq \hat{\mathbf{c}} \in C(n,k)$$

où Pr désigne « la probabilité d'avoir ». En utilisant la règle de Bayes et en supposant que tous les mots de code sont équiprobables, la règle de décision précédente peut encore s'écrire :

$$\hat{\mathbf{c}} = \mathbf{c}_i \Leftrightarrow \Pr(\mathbf{r}\,|\mathbf{c} = \mathbf{c}_i) > \Pr(\mathbf{r}\,|\mathbf{c} = \mathbf{c}_j), \forall \mathbf{c}_j \neq \mathbf{c}_i \in C(n,k)$$

En reprenant l'exemple d'un canal binaire symétrique de probabilité d'erreur p et en notant $d_H(\mathbf{r}, \hat{\mathbf{c}})$ la distance de Hamming entre $\mathbf{r}$ et $\hat{\mathbf{c}}$, la règle de décision est :

$$\hat{\mathbf{c}} = \mathbf{c}_i \Leftrightarrow p^{d_H(\mathbf{c_i},\mathbf{r})}(1-p)^{n-d_H(\mathbf{c}_i,\mathbf{r})} > p^{d_H(\mathbf{c}_j,\mathbf{r})}(1-p)^{n-d_H(\mathbf{c}_j,\mathbf{r})},\ \forall \mathbf{c}_j \neq \mathbf{c}_i$$

En prenant le logarithme des deux membres de l'inégalité précédente et en considérant $p < 0.5$, la règle de décision du maximum de vraisemblance *a posteriori* peut finalement s'écrire :

$$\hat{\mathbf{c}} = \mathbf{c}_i \Leftrightarrow d_H(\mathbf{r}, \mathbf{c}_i) \leqslant d_H(\mathbf{r}, \mathbf{c}_j), \quad \forall\, \mathbf{c}_j \neq \mathbf{c}_i \in C(n,k)$$

Si deux ou plusieurs mots de code sont à la même distance de $\mathbf{r}$, le mot de code $\hat{\mathbf{c}}$ est choisi arbitrairement parmi les mots de code équidistants de $\mathbf{r}$.

Cette procédure de décodage qui est optimale, c'est-à-dire qui minimise la probabilité de décodage erroné, devient difficile à mettre en œuvre dès lors que le nombre de mots de code devient important, ce qui est souvent le cas pour les codes en blocs les plus utilisés.

• *Décodage à partir du syndrome*

Pour contourner cette difficulté, il est possible de réaliser le décodage en utilisant le syndrome $\mathbf{s}$. Rappelons que le syndrome est un vecteur de dimension $(n-k)$ qui dépend uniquement de la configuration d'erreur $\mathbf{e}$. Pour un code en blocs à symboles binaires, le syndrome possède 2^{n-k} configurations ce qui est généralement très inférieur aux 2^k mots de code.

Pour le décodage à partir du syndrome on utilise un tableau à n lignes et deux colonnes. On porte respectivement dans chaque ligne de la première colonne le syndrome $\mathbf{s}$ nul (tous les symboles sont à zéro, pas d'erreur) puis les syndromes $\mathbf{s}$ correspondant à une configuration d'une erreur puis de deux erreurs etc... jusqu'à ce que les n lignes soient remplies. Toutes les configurations des syndromes de la première colonne doivent être différentes. En deuxième colonne, on porte la configuration d'erreur associée à chaque syndrome de la première colonne.

Pour un mot reçu $\mathbf{r}$ on calcule le syndrome $\mathbf{s}$ puis, à l'aide de la table, on en déduit le mot d'erreur $\mathbf{e}$. Finalement on ajoute le mot $\mathbf{e}$ à $\mathbf{r}$ et on obtient le mot de code le plus vraisemblable.

Exemple 4.12 Considérons un code $C(7,4)$ de matrice de contrôle de parité $\mathbf{H}$ avec :

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Ce code possède 16 mots de code mais seulement 8 configurations pour le syndrome comme indiqué dans la table 4.4.

Syndrome $\mathbf{s}$	Mot d'erreur $\mathbf{e}$
000	0000000
001	0000001
010	0000100
011	0001000
100	0000100
101	0010000
110	0100000
111	1000000

Table 4.4: Syndromes et mots d'erreur correspondants pour un code $C(7,4)$.

Supposons que le mot de code émis soit $\mathbf{c} = [0101101]$ et que le mot reçu $\mathbf{r} = [0111101]$ soit entaché d'une erreur en position 3 . Le syndrome est alors égal à $\mathbf{s} = [101]$ et, d'après la table, $\mathbf{e} = [0010000]$. Le mot de code décodé est $\hat{\mathbf{c}} = \mathbf{r} + \mathbf{e} = [0101101]$ et l'erreur est corrigée.

Si le nombre de configurations du syndrome est encore trop élevé pour appliquer cette procédure de décodage, on utilise des algorithmes de décodage spécifiques à certaines classes de codes mais qui, malheureusement, n'exploitent pas toujours toute la puissance de correction du code. Ces algorithmes sont présentés dans un paragraphe ultérieur.

• *Pouvoir de correction*

Soit $\mathbf{c}_j$ le mot de code émis et $\mathbf{c}_l$ son plus proche voisin, nous avons l'inégalité suivante :

$$d_H(\mathbf{c}_j, \mathbf{c}_l) \geqslant d_{\min}$$

En introduisant le mot reçu $\mathbf{r}$ et en supposant que la distance minimale $d_{\min}$ soit égale à $2t + 1$ (t entier), nous pouvons écrire :

$$2t + 1 \leqslant d_H(\mathbf{c}_j, \mathbf{c}_l) \leqslant d_H(\mathbf{c}_j, \mathbf{r}) + d_H(\mathbf{c}_l, \mathbf{r})$$

Nous voyons que si le nombre d'erreurs est inférieur ou égal à t, $\mathbf{c}_j$ est le mot de code le plus vraisemblable car plus voisin de $\mathbf{r}$ que de $\mathbf{c}_l$ et ainsi les t erreurs pourront être corrigées. Si maintenant la distance minimale est de $(2t + 2)$, en utilisant le même raisonnement, on aboutit au même pouvoir de correction. En conclusion, le pouvoir de correction d'un code en blocs linéaire de distance minimale $d_{\min}$ avec décodage ferme est égale à :

$$t = \lfloor \frac{d_{\min} - 1}{2} \rfloor \tag{4.23}$$

où $\lfloor x \rfloor$ est la partie entière de x par valeur inférieure (par exemple $\lfloor 2.5 \rfloor = 2$).

• *Probabilité de décodage erroné d'un mot de code*

Pour un code en blocs linéaire $C(n, k)$ de pouvoir de correction t, le mot de code émis sera mal décodé si il y a $t + j$ erreurs, $j = 1, 2, \cdots, n - t$, dans le mot reçu $\mathbf{r}$. Pour un canal binaire symétrique de probabilité p, la probabilité $P_{e,\text{mot}}$ de faire un décodage erroné du mot de code émis est égale à :

$$P_{e,\text{mot}} < \sum_{j=t+1}^{n} \left(\begin{array}{c} n \\ j \end{array} \right) p^j (1-p)^{n-j} \tag{4.24}$$

On peut également déterminer la probabilité d'erreur binaire $P_{e,\text{bit}}$ sur les données d'information après décodage. En présence de décodage erroné, le décodeur à maximum de vraisemblance *a posteriori* rajoute au plus t erreurs en choisissant le mot de code à la distance minimale du mot reçu. La probabilité d'erreur est donc bornée par :

$$P_{e,\text{bit}} < \frac{1}{n} \sum_{j=t+1}^{n} (j + t) \left(\begin{array}{c} n \\ j \end{array} \right) p^j (1-p)^{n-j} \tag{4.25}$$

Si la transmission est réalisée avec une modulation de phase binaire (MDP-2, MDP-4), la probabilité p est égale à :

$$p = \frac{1}{2}\text{erfc}\sqrt{\frac{RE_b}{N_0}}$$

où R est le rendement du code, E_b l'énergie reçue par donnée d'information transmise et N_0 la densité spectrale de puissance mono-latérale du bruit. La figure 4.5 représente la probabilité d'erreur par bit et par mot après un décodage algébrique pour le code BCH (15,7). La modulation est une MDP-4 et le canal est gaussien. Les bornes supérieures exprimées respectivement par (4.24) et (4.25) sont également tracées.

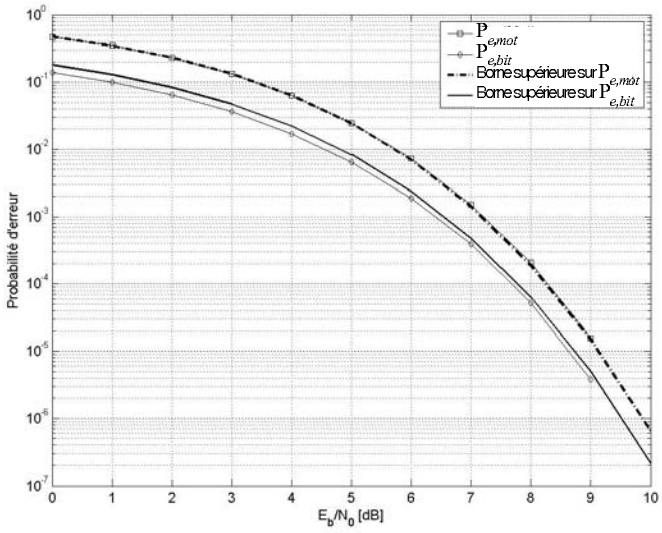


Figure 4.5 – Performances du décodage algébrique du code BCH (15,7). Transmission MDP-4 sur canal gaussien.

Décodage pondéré

En considérant un canal à bruit additif blanc gaussien et une transmission en modulation de phase à symboles binaires (MDP-2 ou MDP-4), les composantes $r_j, \quad j = 0, 1, \cdots, n-1$ du mot reçu $\mathbf{r}$ sont de la forme :

$$r_j = \sqrt{E_s}\tilde{c}_j + b_j, \quad \tilde{c}_j = 2c_j - 1$$

où $c_j = 0, 1$ est le symbole en position j du mot de code $\mathbf{c}$, $\tilde{c}_j$ est le symbole binaire associé à c_j, E_s est l'énergie reçue par symbole transmis du mot de code

et b_j un bruit blanc, gaussien, centré, de variance égale à σ_b^2.

• *Décodage à maximum de vraisemblance a posteriori*

Le décodage selon le critère du maximum de vraisemblance *a posteriori* consiste à rechercher le mot de code $\hat{\mathbf{c}}$ tel que :

$$\hat{\mathbf{c}} = \mathbf{c} \Leftrightarrow \Pr\{\mathbf{c}/\mathbf{r}\} > \Pr\{\mathbf{c}'/\mathbf{r}\}, \quad \forall \mathbf{c} \neq \mathbf{c}' \in C(n,k)$$

En utilisant la règle de Bayes et en supposant tous les mots de code équiprobables, l'inégalité précédente peut encore s'écrire :

$$\hat{\mathbf{c}} = \mathbf{c} \text{ si } p(\mathbf{r}/\mathbf{c}) > p(\mathbf{r}/\mathbf{c}'), \qquad \forall\, \mathbf{c} \neq \mathbf{c}' \in C(n,k) \tag{4.26}$$

où $p(\mathbf{r}/\mathbf{c})$ est la densité de probabilité de l'observation $\mathbf{r}$ conditionnellement au mot de code $\mathbf{c}$.

Pour un canal gaussien, la densité de probabilité $p(\mathbf{r}/\mathbf{c})$ est égale à :

$$p(\mathbf{r}/\mathbf{c}) = \left(\frac{1}{\sqrt{2\pi}\sigma_b}\right)^n \exp\left(-\frac{1}{2\sigma_b^2}\sum_{j=0}^{n-1}(r_j - \sqrt{E_s}\tilde{c}_j)^2\right)$$

où σ_b^2 est la variance du bruit.

En remplaçant les deux densités de probabilité par leurs expressions respectives dans l'inégalité (4.26) et après quelques calculs élémentaires, on obtient :

$$\hat{\mathbf{c}} = \mathbf{c} \Leftrightarrow \sum_{j=0}^{n-1} r_j c_j > \sum_{j=0}^{n-1} r_j c'_j, \quad \forall \mathbf{c} \neq \mathbf{c}' \in C(n,k)$$

Le mot de code décodé est celui qui maximise le produit scalaire $\langle \mathbf{r}, \mathbf{c} \rangle$. On pourrait aussi montrer que le mot de code décodé est celui qui minimise le carré de la distance euclidienne $\left\|\mathbf{r} - \sqrt{E_s}\tilde{\mathbf{c}}\right\|^2$.

Cette procédure de décodage est applicable dans la mesure où le nombre de mots de code n'est pas trop élevé. En présence d'un grand nombre de mots de code on peut utiliser un algorithme dû à Chase dont le principe est d'appliquer la procédure précédente de décodage en restreignant l'espace de recherche à un sous-ensemble de mots de code.

• *Algorithme de Chase*

L'algorithme de Chase est une procédure de décodage sous-optimale qui utilise le critère du maximum de vraisemblance *a posteriori* mais en considérant un sous-ensemble très réduit des mots de code. Pour déterminer ce sous-ensemble des mots de code, l'algorithme de Chase procède de la manière suivante.

– Étape 1 : Le mot reçu $\mathbf{r}$, constitué de symboles analogiques, est transformé en un mot à symboles binaires $\mathbf{z}_0 = (z_{00} \cdots z_{0j} \cdots z_{0n-1})$ par seuillage.

$$z_{0j} = \text{sgn}(r_j)$$

avec la convention suivante :

$$\begin{aligned} \text{sgn}(x) &= 1 \text{ si } x \geqslant 0 \\ &= 0 \text{ si } x < 0 \end{aligned}$$

Le mot binaire $\mathbf{z}_0$ est ensuite décodé par un algorithme à décision ferme autre que le maximum de vraisemblance *a posteriori* (nous présenterons ultérieurement des algorithmes de décodage pour codes en blocs). Soit $\mathbf{c}_0$ le mot de code obtenu.

– Étape 2 : Soient $j_1, j_2, \cdots, j_t$ les positions des symboles les moins fiables c'est-à-dire telles que les amplitudes $|r_j|$ sont les plus faibles.
$2^t - 1$ mots $\mathbf{e}_i$ sont construits en formant toutes les combinaisons binaires possibles non nulles sur les positions $j_1, j_2, \cdots, j_t$. Sur les positions autres que $j_1, j_2, \cdots, j_t$, les symboles de $\mathbf{e}_i$ sont mis à zéro. Rappelons que t est le pouvoir de correction du code.
– Étape 3 : Chacun des $2^t - 1$ mots $\mathbf{e}_i$ sert à définir des mots $\mathbf{z}_i$ avec :

$$\mathbf{z}_i = \mathbf{z}_0 + \mathbf{e}_i$$

Un décodeur ferme traite les mots $\mathbf{z}_i$ pour obtenir au plus $2^t - 1$ mots de code $\mathbf{c}_i$. Notons que le mot en sortie du décodeur algébrique n'est pas toujours un mot de code et seuls les mots de code seront considérés dans l'application du critère de décision.
– Étape 4 : La règle du maximum de vraisemblance *a posteriori* est appliquée sur le sous ensemble des mots de code $\mathbf{c}_i$ créé dans l'étape précédente.

Exemple 4.13

Soit un code $C(n, k)$ de pouvoir de correction $t = 3$. Le sous ensemble des mots de code est constitué de 8 mots de code dont 7 sont élaborés à partir des mots $\mathbf{e}_i$. Les mots $\mathbf{e}_i$ sont des mots de longueur n dont les composantes sont nulles sauf éventuellement aux places j_1, j_2 et j_3 (voir tableau ci-dessous).

i	$e_{i,j_1}\ e_{i,j_2}\ e_{i,j_3}$
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

• *Probabilité de décodage erroné d'un mot de code*

Supposons que le mot de code émis soit $\mathbf{c}_0 = (c_{01} \cdots c_{0j} \cdots c_{0n-1})$ et soit $\mathbf{r}_0 = (r_0 \cdots r_j \cdots r_{n-1})$ le mot reçu avec :

$$r_j = \sqrt{E_s}\tilde{c}_{0j} + b_j$$

Le mot de code $\mathbf{c}_0$ sera mal décodé si :

$$\sum_{j=0}^{n-1} r_j c_{0,j} < \sum_{j=0}^{n-1} r_j c_{l,j} \qquad \forall\, \mathbf{c}_l \neq \mathbf{c}_0 \in C(n,k)$$

Le code étant linéaire, on peut, sans nuire à la généralité, supposer que le mot de code émis est le mot nul soit $c_{0,j} = 0$ pour $j = 0, 1, \cdots, n-1$.

La probabilité de décodage erroné $P_{e,\text{mot}}$ d'un mot de code est alors égale à :

$$P_{e,\text{mot}} = \Pr\left(\sum_{j=0}^{n-1} r_j c_{1,j} > 0 \text{ ou } \ldots \sum_{j=0}^{n-1} r_j c_{l,j} > 0 \text{ ou } \ldots\right)$$

La probabilité $P_{e,\text{mot}}$ peut être bornée supérieurement par une somme de probabilités et, après quelques calculs classiques, elle peut s'écrire sous la forme :

$$P_{e,\text{mot}} \leqslant \frac{1}{2} \sum_{j=2}^{2^k} \text{erfc}\sqrt{w_j \frac{E_s}{N_0}}$$

où w_j est le poids de Hamming du j-ème mot de code.

En supposant que le code $C(n,k)$ possède A_w mots de code de poids w, la probabilité $P_{e,\text{mot}}$ peut encore se mettre sous la forme :

$$P_{e,\text{mot}} < \frac{1}{2} \sum_{w=d_{\min}}^{n} A_w \text{erfc}\sqrt{w \frac{E_s}{N_0}} \tag{4.27}$$

En introduisant l'énergie E_b reçue par donnée d'information transmise, la probabilité $P_{e,\text{mot}}$ peut finalement s'écrire :

$$P_{e,\text{mot}} < \frac{1}{2} \sum_{w=d_{\min}}^{n} A_w \text{erfc}\sqrt{w \frac{RE_b}{N_0}} \tag{4.28}$$

où R est le rendement du code.

On peut également établir une borne supérieure de la probabilité d'erreur binaire sur les symboles d'information après décodage.

$$P_{e,\text{bit}} < \frac{1}{2} \sum_{w=d_{\min}}^{n} \frac{w}{n} A_w \text{erfc}\sqrt{w \frac{RE_b}{N_0}} \tag{4.29}$$

Pour calculer les probabilités $P_{e,\text{mot}}$ et $P_{e,\text{bit}}$ il faut connaître le nombre A_w de mots de code de poids w. Pour les codes BCH étendus les quantités A_w sont données dans [4.2].

À titre d'exemple, la table 4.5 donne les quantités A_w pour trois codes de Hamming étendus.

Pour le code (32,26) les quantités A_w manquantes sont obtenues à partir de la relation $A_w = A_{n-w}$ pour $0 \leqslant w \leqslant n/2$, $n/2$ pair.

n	k	$d_{\min}$	A_4	A_6	A_8	A_{10}	A_{12}	A_{14}	A_{16}
8	4	4	14	-	1	-	-	-	-
16	11	4	140	448	870	448	140	-	1
32	26	4	1240	27776	330460	2011776	7063784	14721280	18796230

Table 4.5 – A_w pour trois codes de Hamming étendus.

Les quantités A_w pour les codes de Hamming non étendus se déduisent de celles des codes étendus en résolvant le système d'équations suivant :

$$(n+1)A_{w-1} = wA_w^{\text{étendu}}$$
$$wA_w = (n+1-w)A_{w-1}$$

où n est la longueur des mots du code non étendu.

Pour le code de Hamming (7,4), par exemple, les quantités A_w sont :

$$\begin{array}{lcll} 8A_3 & = & 4A_4^{\text{étendu}} & A_3 = 7 \\ 4A_4 & = & 4A_3 & A_4 = 7 \\ 8A_7 & = & 8A_8^{\text{étendu}} & A_7 = 1 \end{array}$$

Pour les codes de Golay et Golay étendu, les quantités A_w sont données dans la table 4.6.

Poids	A_w (23,12)	A_w (24,12)
0	1	1
7	253	0
8	506	759
11	1288	0
12	1288	2576
15	506	0
16	253	759
23	1	0
24	0	1

Table 4.6 – A_w pour les codes de Golay et Golay étendu.

À fort rapport signal à bruit, la probabilité d'erreur $P_{e,\text{mot}}$ est bien approchée par le premier terme de la série :

$$P_{e,\text{mot}} \cong \frac{1}{2} A_{d_{\min}} \text{erfc} \sqrt{\frac{R d_{\min} E_b}{N_0}} \quad \text{si} \quad \frac{E_b}{N_0} >> 1 \tag{4.30}$$

Il en va de même pour la probabilité d'erreur $P_{e,\text{bit}}$ sur les symboles d'information.

$$P_{e,\text{bit}} \cong \frac{d_{\min}}{n} P_{e,\text{mot}} \quad \text{si} \quad \frac{E_b}{N_0} >> 1 \tag{4.31}$$

En l'absence de codage, la probabilité d'erreur sur les symboles binaires est égale à :

$$p = \frac{1}{2} \text{erfc} \sqrt{\frac{E_b}{N_0}}$$

Comme cela a été vu dans la section 1.5, en comparant les deux expressions de la probabilité d'erreur binaire avec et sans codage, on observe que le rapport signal à bruit E_b/N_0 est multiplié par $Rd_{\min}$ en présence de codage. Si ce coefficient multiplicateur est supérieur à 1, le codage agit comme un amplificateur du rapport signal à bruit dont le gain asymptotique est approché par

$$G_a = 10\log(Rd_{\min})(\text{dB})$$

Pour illustrer ces bornes, reprenons l'exemple du code BCH(15,7) transmis sur un canal gaussien avec une modulation MDP-4. Sur la figure 4.6, nous avons représenté l'évolution de la probabilité d'erreur binaire et par mot obtenues par simulation à partir de l'algorithme sous-optimal de Chase (4 positions non fiables). Nous avons également représenté les deux premiers termes des sommes intervenant dans les bornes données par (4.28) et (4.29). À titre de référence, nous avons également tracé la courbe de probabilité d'erreur binaire d'une modulation MDP-4 sans codage.

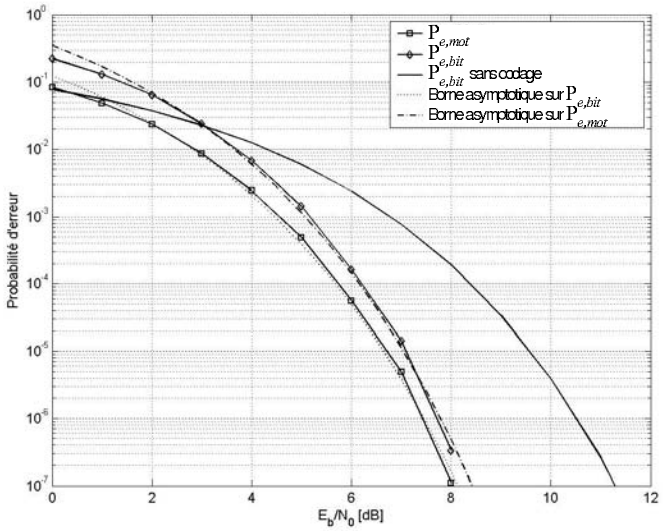


Figure 4.6 – Performances du décodage pondéré du code BCH (15,7). Transmission MDP-4 sur un canal gaussien.

4.4 Décodage et performances des codes à symboles non binaires

4.4.1 Décodage à entrée ferme des codes de Reed-Solomon

Les algorithmes de décodage à entrée ferme permettent de décoder les codes de Reed-Solomon (RS) et les codes BCH à symboles binaires. Nous commencerons par présenter le principe du décodage des codes RS puis nous traiterons le cas des codes BCH à symboles binaires comme un cas particulier du décodage des codes RS.

En supposant que $c(x)$ est le mot de code émis alors, pour un canal à entrée et sortie discrètes, le mot reçu peut toujours se mettre sous la forme :

$$r(x) = c(x) + e(x)$$

où $e(x)$ est le polynôme d'erreur avec :

$$e(x) = e_0 + e_1 x + \cdots + e_j x^j + \cdots + e_{n-1} x^{n-1}, \quad e_j \in \mathbf{F}_q \quad \forall\, j$$

Lorsque $e_j \neq 0$ une erreur se trouve en position j.

Il a été vu précédemment que le polynôme générateur d'un code RS ou d'un code BCH (avec $l = 1$) corrigeant t erreurs avait pour racines $\alpha, \cdots, \alpha^j, \cdots, \alpha^{2t}$ et, que les mots de code étaient des multiples du polynôme générateur. Ainsi, pour tout mot de code nous pouvons écrire :

$$c(\alpha^i) = 0; \forall\, i = 1, 2, \cdots, 2t$$

Le décodage des codes RS et des codes BCH binaires peut être réalisé à partir d'un vecteur à $2t$ composantes $\mathbf{S} = [S_1 \cdots S_j \cdots S_{2t}]$, appelé *syndrome*.

$$S_j = r(\alpha^j) = e(\alpha^j), \quad j = 1, 2, \cdots, 2t \tag{4.32}$$

Lorsque les composantes du vecteur $\mathbf{S}$ sont toutes nulles, il n'y a pas d'erreur ou du moins d'erreurs détectables. Lorsque certaines composantes du vecteur $\mathbf{S}$ sont non nulles, des erreurs sont présentes qui, sous certaines conditions, peuvent être corrigées.

En présence de t erreurs de transmission, le polynôme d'erreur $e(x)$ est de la forme :

$$e(x) = e_{n_1} x^{n_1} + e_{n_2} x^{n_2} + \cdots + e_{n_t} x^{n_t}$$

où les e_{n_l} sont des coefficients non nuls prenant leur valeur dans le corps $\mathbf{F}_q$. Les composantes S_j du syndrome $\mathbf{S}$ sont égales à :

$$S_j = e_{n_1} (\alpha^j)^{n_1} + \cdots + e_{n_l} (\alpha^j)^{n_l} + \cdots + e_{n_t} (\alpha^j)^{n_t}$$

En posant $Z_l = \alpha^{n_l}$ et, pour simplifier les notations $e_{n_l} = e_l$, la composante S_j du syndrome est encore égale à :

$$S_j = e_1 Z_1^j + \cdots + e_l Z_l^j + \cdots + e_t Z_t^j \tag{4.33}$$

Pour déterminer la position des erreurs de transmission il suffit donc de connaître la valeur des quantités $Z_l; j = 1, 2, \cdots, t$ puis, pour corriger les erreurs d'évaluer les coefficients $e_l; l = 1, 2, \cdots, t$.

La principale difficulté dans le décodage des codes RS ou des codes BCH binaires est de déterminer la position des erreurs. Deux méthodes sont principalement utilisées pour décoder les codes RS ou les codes BCH binaires : la méthode directe due à Peterson et la méthode itérative utilisant l'algorithme de Berlekamp-Massey ou l'algorithme d'Euclide.

4.4.2 Méthode directe de Peterson

Description de l'algorithme pour des codes à symboles non binaires

Cette méthode est bien adaptée pour le décodage des codes RS ou des codes BCH binaires corrigeant un faible nombre d'erreurs, typiquement de 1 à 3. En effet, la complexité de cette méthode augmente comme le carré du pouvoir de correction du code alors que pour la méthode itérative, la complexité augmente seulement linéairement avec le pouvoir de correction du code.

Pour déterminer la position des erreurs introduisons un polynôme $\sigma_d(x)$ appelé polynôme *localisateur d'erreur* dont les racines sont précisément les quantités Z_l.

$$\sigma_d(x) = \prod_{l=1}^{t} (x + Z_l)$$

En développant cette expression, le polynôme $\sigma_d(x)$ est encore égal à :

$$\sigma_d(x) = x^t + \sigma_1 x^{t-1} + \cdots + \sigma_j x^{t-j} + \cdots + \sigma_t$$

où les coefficients σ_j sont des fonctions des quantités Z_l.

À partir de l'expression de S_j on peut former un système non linéaire de $2t$ équations.

$$S_j = \sum_{i=1}^{t} e_i Z_i^j, \quad j = 1, 2, \cdots, 2t$$

Les quantités $Z_l, l = 1, \cdots, t$ étant les racines du polynôme localisateur d'erreur $\sigma_d(x)$, on peut écrire :

$$\sigma_d(Z_l) = Z_l^t + \sum_{j=1}^{t} \sigma_j Z_l^{t-j} = 0, \quad l = 1, 2, \cdots, t \tag{4.34}$$

En multipliant les deux membres de cette expression par un même terme $e_l Z_l^q$, on obtient :

$$e_l Z_l^{t+q} + \sum_{j=1}^{t} \sigma_j e_l Z_l^{t+q-j} = 0, \quad l = 1, 2, \cdots, t \tag{4.35}$$

En sommant les relations (4.35) pour l allant de 1 à t et en tenant compte de la définition de la composante S_j du syndrome $\mathbf{S}$, nous pouvons écrire :

$$S_{t+q} + \sigma_1 S_{t+q-1} + \cdots + \sigma_j S_{t+q-j} + \cdots + \sigma_t S_q = 0, \quad \forall\, q \tag{4.36}$$

Pour un code RS corrigeant une erreur ($t = 1$) dans un bloc de n symboles, le syndrome $\mathbf{S}$ possède deux composantes S_1 et S_2. Le coefficient σ_1 du polynôme localisateur d'erreur est déterminé à partir de la relation (4.36) en faisant $t = 1$ et $q = 1$.

$$S_2 + \sigma_1 S_1 = 0 \rightarrow \sigma_1 = \frac{S_2}{S_1} \tag{4.37}$$

De la même façon, pour un code RS corrigeant deux erreurs ($t = 2$) dans un bloc de n symboles, le syndrome possède quatre composantes S_1, S_2, S_3, S_4. En utilisant la relation (4.36) avec $t = 2$ et $q = 1, 2$ on obtient le système de deux équations suivant :

$$\begin{aligned} \sigma_1 S_2 + \sigma_2 S_1 &= S_3 \\ \sigma_1 S_3 + \sigma_2 S_2 &= S_4 \end{aligned}$$

La résolution de ce système à deux équations permet de déterminer les coefficients σ_1 et σ_2 du polynôme localisateur d'erreur.

$$\begin{aligned} \sigma_1 &= \tfrac{1}{\Delta_2}\left[S_1 S_4 + S_2 S_3\right] \\ \sigma_2 &= \tfrac{1}{\Delta_2}\left[S_2 S_4 + S_3^2\right] \end{aligned} \tag{4.38}$$

où Δ_2 est le déterminant du système à deux équations.

$$\Delta_2 = S_2^2 + S_1 S_3$$

Enfin, pour un code RS corrigeant trois erreurs ($t = 3$), la relation (4.36) avec $t = 3$ et $q = 1, 2, 3$ conduit au système de trois équations suivant :

$$\begin{aligned} \sigma_1 S_3 + \sigma_2 S_2 + \sigma_3 S_1 &= S_4 \\ \sigma_1 S_4 + \sigma_2 S_3 + \sigma_3 S_2 &= S_5 \\ \sigma_1 S_5 + \sigma_2 S_4 + \sigma_3 S_3 &= S_6 \end{aligned}$$

La résolution de ce système permet de déterminer les coefficients σ_1, σ_2 et σ_3 du polynôme localisateur d'erreur.

$$\begin{aligned} \sigma_1 &= \tfrac{1}{\Delta_3}\left[S_1 S_3 S_6 + S_1 S_4 S_5 + S_2^2 S_6 + S_2 S_3 S_5 + S_2 S_4^2 + S_3^2 S_4\right] \\ \sigma_2 &= \tfrac{1}{\Delta_3}\left[S_1 S_4 S_6 + S_1 S_5^2 + S_2 S_3 S_6 + S_2 S_4 S_5 + S_3^2 S_5 + S_3 S_4^2\right] \\ \sigma_3 &= \tfrac{1}{\Delta_3}\left[S_2 S_4 S_6 + S_2 S_5^2 + S_3^2 S_6 + S_4^3\right] \end{aligned} \tag{4.39}$$

où Δ_3 est le déterminant du système à trois équations.

$$\Delta_3 = S_1 S_3 S_5 + S_1 S_4^2 + S_2^2 S_5 + S_3^3$$

Mise en œuvre du décodeur de Peterson pour un code RS de paramètre $t = 3$

1. Calcul des $2t$ syndromes S_j : $S_j = r(\alpha^j)$
2. Détermination du nombre d'erreurs :
   - Cas (a) $S_j = 0$, $\forall\, j$: aucune erreur détectable.
   - Cas (b) $\Delta_3 \neq 0$: présence de trois erreurs.
   - Cas (c) $\Delta_3 = 0$ et $\Delta_2 \neq 0$: présence de deux erreurs.
   - Cas (d) $\Delta_3 = \Delta_2 = 0$ et $S_1 \neq 0$: présence d'une erreur.
3. Calcul du polynôme localisateur d'erreur $\sigma_d(x)$
   - Cas (b) Utiliser (4.39)
   - Cas (c) Utiliser (4.38)
   - Cas (d) Utiliser (4.37)
4. Recherche des racines de $\sigma_d(x)$ dans le corps $\mathbf{F}_q$
5. Calcul des coefficients d'erreur e_i
   - Cas (b)

$$e_i = \tfrac{1}{\Delta}\left[S_1(Z_k^2 Z_p^3 + Z_k^3 Z_p^2) + S_2(Z_k^3 Z_p + Z_k Z_p^3) + S_3(Z_k^2 Z_p + Z_k Z_p^2)\right], \quad k \neq p \neq i, (i,k,p) \in \{1,2,3\}^3$$

$$\Delta = \sum_{\substack{1 \leqslant i_1, i_2, i_3 \leqslant 3 \\ i_1 + i_2 + i_3 = 6 \\ i_1 \neq i_2 \neq i_3}} Z_1^{i_1} Z_2^{i_2} Z_3^{i_3}$$

   - Cas (c)

$$e_i = \frac{S_1 Z_p + S_2}{Z_i(Z_1 + Z_2)},\ p \neq i,\ (i,p) \in \{1,2\}^2$$

   - Cas (d)

$$e_1 = \frac{S_1^2}{S_2}$$

6. Correction des erreurs : $\hat{c}(x) = r(x) + e(x)$

Exemple 4.14

Pour illustrer le décodage d'un code RS suivant la méthode directe, nous allons maintenant traiter un exemple en considérant un code RS corrigeant jusqu'à trois erreurs ($t = 3$) et ayant les paramètres suivants :

$$m = 4 \quad q = 16 \quad n = 15 \quad n - k = 6$$

Supposons par exemple que le mot du code émis soit $c(x) = 0$ et que le mot reçu soit entaché de deux erreurs.

$$r(x) = \alpha^7 x^3 + \alpha^3 x^6$$

1. Calcul des composantes du syndrome

$$\begin{array}{ll} S_1 = \alpha^{10} + \alpha^9 = \alpha^{13} & S_4 = \alpha^{19} + \alpha^{27} = \alpha^6 \\ S_2 = \alpha^{13} + \alpha^{15} = \alpha^6 & S_5 = \alpha^{22} + \alpha^{33} = \alpha^4 \\ S_3 = \alpha^{16} + \alpha^{21} = \alpha^{11} & S_6 = \alpha^{25} + \alpha^{39} = \alpha^{13} \end{array}$$

2. Détermination du nombre d'erreurs

$$\Delta_3 = 0 \qquad \Delta_2 = \alpha^8$$

Δ_3 étant nul et $\Delta_2 \neq 0$, on est en présence de deux erreurs.

3. Calcul des coefficients σ_1 et σ_2 du polynôme localisateur d'erreur.

$$\begin{aligned} \sigma_1 &= \frac{1}{\Delta_2}\left[S_1S_4 + S_2S_3\right] = \frac{\alpha^{19}+\alpha^{17}}{\alpha^8} = \alpha^{11} + \alpha^9 = \alpha^2 \\ \sigma_2 &= \frac{1}{\Delta_2}\left[S_2S_4 + S_3^2\right] = \frac{\alpha^{12}+\alpha^{22}}{\alpha^8} = \alpha^4 + \alpha^{14} = \alpha^9 \end{aligned}$$

Le polynôme localisateur d'erreur est donc égal à :

$$\sigma_d(x) = x^2 + \alpha^2 x + \alpha^9$$

4. Recherche des deux racines du polynôme localisateur d'erreur.
En passant en revue les éléments du corps $\mathbf{F}_{16}$ on trouve que α^3 et α^6 annulent le polynôme $\Lambda(x)$. Les erreurs portent donc sur les termes en x^3 et en x^6 du mot $r(x)$.

5. Calcul des coefficients d'erreur e_1 et e_2.

$$\begin{aligned} e_1 &= \frac{S_1Z_2+S_2}{Z_1Z_2+Z_1^2} = \frac{\alpha^{19}+\alpha^6}{\alpha^9+\alpha^6} = \frac{\alpha^{12}}{\alpha^5} = \alpha^7 \\ e_2 &= \frac{S_1Z_1+S_2}{Z_1Z_2+Z_2^2} = \frac{\alpha^{16}+\alpha^6}{\alpha^9+\alpha^{12}} = \frac{\alpha^{11}}{\alpha^8} = \alpha^3 \end{aligned}$$

6. Correction des erreurs

$$c(x) = (\alpha^7x^3 + \alpha^3x^6) + (\alpha^7x^3 + \alpha^3x^6) = 0$$

Le mot de code émis est le mot nul, les deux erreurs sont donc bien corrigées.

Simplification de l'algorithme de Peterson pour des codes binaires

Pour les codes BCH à symboles binaires il n'est pas nécessaire de calculer les coefficients e_j. En effet, ces coefficients étant binaires, ils valent nécessairement 1 en présence d'une erreur en position j. Le calcul des coefficients σ_j peut aussi être simplifié en tenant compte du fait que pour un code à symboles binaires on a :

$$S_{2j} = e(\alpha^{2j}) = \left[e(\alpha^j)\right]^2 = S_j^2$$

Pour un code BCH à symboles binaires corrigeant jusqu'à $t = 3$ erreurs, en tenant compte de la remarque précédente et en utilisant les expressions des trois coefficients σ_j du polynôme localisateur, on obtient :

$$\begin{aligned} \sigma_1 &= S_1 \\ \sigma_2 &= \frac{S_1^2S_3+S_5}{S_1^3+S_3} \\ \sigma_3 &= (S_1^3 + S_3) + S_1\frac{S_1^2S_3+S_5}{S_1^3+S_3} \end{aligned} \tag{4.40}$$

Pour un code BCH à symboles binaires corrigeant jusqu'à $t = 2$ erreurs, en tenant compte également de la remarque précédente et en utilisant les expressions des deux coefficients σ_j du polynôme localisateur d'erreur, on obtient :

$$\begin{array}{rcl} \sigma_1 & = & S_1 \\ \sigma_2 & = & \frac{S_3+S_1^3}{S_1} \end{array} \tag{4.41}$$

Finalement, en présence d'une erreur $\sigma_2 = \sigma_3 = 0$ et $\sigma_1 = S_1$.

Exemple 4.15

Considérons un code BCH corrigeant deux erreurs ($t = 2$) dans un bloc de $n = 15$ symboles de polynôme générateur égal à :

$$g(x) = x^8 + x^7 + x^6 + x^4 + 1$$

Supposons que le mot de code émis soit $c(x) = 0$ et que le mot reçu $r(x)$ soit entaché de deux erreurs.

$$r(x) = x^8 + x^3$$

Le décodage va se faire en trois étapes : calcul du syndrome $\mathbf{S}$, détermination des coefficients σ_l du polynôme localisateur d'erreur et recherche de ses racines dans le corps $\mathbf{F}_{16}$.

1. Calcul du syndrome $\mathbf{S}$: nous avons uniquement besoin de calculer les composantes d'indice impair S_1 et S_3 du syndrome $\mathbf{S}$. En utilisant les représentations binaires des éléments du corps $\mathbf{F}_{16}$ données dans l'annexe 2, et en tenant compte du fait que $\alpha^{15} = 1$, nous avons :

$$\begin{array}{l} S_1 = r(\alpha) = \alpha^8 + \alpha^3 = \alpha^{13} \\ S_3 = r(\alpha^3) = \alpha^{24} + \alpha^9 = \alpha^9 + \alpha^9 = 0 \end{array}$$

2. Détermination des coefficients σ_1 et σ_2 du p olynôme localisateur d'erreur. En utilisant les expressions des coefficients σ_1 et σ_2, on obtient :

$$\begin{array}{l} \sigma_1 = S_1 = \alpha^{13} \\ \sigma_2 = \frac{S_3+S_1^3}{S_1} = S_1^2 = \alpha^{26} = \alpha^{11} \quad (\alpha^{15} = 1) \end{array}$$

   et le polynôme localisateur d'erreur est égal à :

$$\sigma_d(x) = x^2 + \alpha^{13}x + \alpha^{11}$$

3. Recherche des racines du polynôme localisateur d'erreur dans le corps $\mathbf{F}_{16}$. En essayant tous les éléments du corps $\mathbf{F}_{16}$, on peut vérifier que les racines du polynôme localisateur d'erreur sont α^3 et α^8. En effet, nous avons

$$\sigma(\alpha^3) = \alpha^6 + \alpha^{16} + \alpha^{11} = \alpha^6 + \alpha + \alpha^{11} = 1100 + 0010 + 1110 = 0000$$

$$\sigma(\alpha^8) = \alpha^{16} + \alpha^{21} + \alpha^{11} = \alpha + \alpha^6 + \alpha^{11} = 0010 + 1100 + 1110 = 0000$$

Les erreurs de transmission portent sur les termes x^8 et x^3 du mot reçu $r(x)$. Le mot de code émis est donc $c(x) = 0$ et les deux erreurs sont bien corrigées.
Le lecteur pourra vérifier qu'en présence d'une seule erreur, $r(x) = x^j$; $0 \leqslant j \leqslant (n-1)$, la correction est encore réalisée correctement puisque :

$$S_1 = \alpha^j;\ S_3 = \alpha^{3j};\ \sigma_1 = \alpha^j;\ \sigma_2 = 0;\ \sigma_d(x) = x(x + \sigma_1)$$

et le polynôme localisateur d'erreur possède une racine unique $\sigma_1 = \alpha^j$.

Algorithme de Chien

Pour rechercher les racines du polynôme localisateur d'erreur pour des codes à symboles binaires, on peut éviter de passer en revue tous les éléments du corps $\mathbf{F}_q$ en utilisant un algorithme itératif dû à Chien.
En divisant le polynôme $\sigma_d(x)$ par x^t, nous obtenons :

$$\tilde{\sigma}_d(x) = \frac{\sigma_d(x)}{x^t} = 1 + \sigma_1 x^{-1} + \cdots + \sigma_j x^{-j} + \cdots + \sigma_t x^{-t}$$

Les racines du polynôme $\sigma_d(x)$ qui sont aussi les racines de $\tilde{\sigma}_d(x)$ sont de la forme α^{n-j} où $j = 1, 2, \ldots, n-1$ et $n = q - 1$.
Ainsi α^{n-j} est racine de $\tilde{\sigma}_d(x)$ si :

$$\sigma_1 \alpha^{-n+j} + \cdots + \sigma_p x^{-np+jp} + \cdots + \sigma_t x^{-nt+jt} = 1$$

En tenant compte du fait que $\alpha^n = 1$, la condition à satisfaire pour que α^{n-j} soit racine du polynôme localisateur d'erreur est :

$$\sum_{p=1}^{t} \sigma_p \alpha^{jp} = 1; \quad j = 1, 2, \cdots, (n-1) \tag{4.42}$$

L'algorithme de Chien vient tester si la condition (4.42) est vérifiée en utilisant le circuit représenté sur la figure 4.7.
Ce circuit comprend un registre à t mémoires initialisé avec les t coefficients σ_j du polynôme localisateur d'erreur et un registre à n mémoires qui stocke les symboles r_j ; $j = 0, 1, \cdots, (n-1)$ du mot $r(x)$. À la première impulsion d'horloge, le circuit réalise le calcul du membre de gauche de l'expression (4.42) pour $j = 1$. Si le résultat de ce calcul est égal à 1, α^{n-1} est racine du polynôme localisateur d'erreur et l'erreur qui portait sur le symbole r_{n-1} est alors corrigée. Si le résultat de ce calcul est égal à 0, aucune correction n'est effectuée. À l'issue de cette première phase, les coefficients σ_j contenus dans les t mémoires du registre sont remplacés par $\sigma_j \alpha^j$. À la deuxième impulsion d'horloge le circuit réalise de nouveau le calcul du membre de gauche de l'expression (4.42) pour $j = 2$. Si le résultat de ce calcul est égal à 1, α^{n-2} est racine du polynôme localisateur d'erreur et l'erreur qui portait sur le symbole r_{n-2} est alors corrigée. L'algorithme se poursuit de la même manière pour les impulsions d'horloge suivantes.

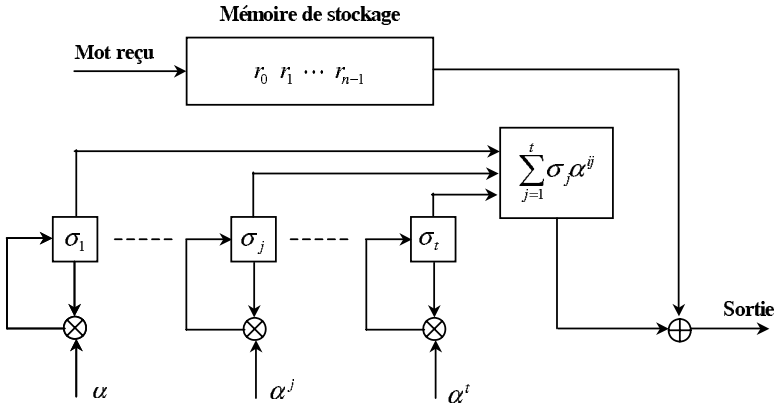


Figure 4.7 – Schéma de principe du circuit réalisant l'algorithme de Chien.

4.4.3 Méthode itérative

Le décodage des codes RS ou des codes BCH binaires selon la méthode itérative utilise deux polynômes, le polynôme localisateur d'erreur $\Lambda(x)$ et le polynôme évaluateur d'erreur $\Gamma(x)$. Ces deux polynômes sont respectivement définis par :

$$\Lambda(x) = \prod_{j=1}^{t} (1 + Z_j x) \tag{4.43}$$

$$\Gamma(x) = \sum_{i=1}^{t} e_i Z_i x \frac{\Lambda(x)}{1 + Z_i x} \tag{4.44}$$

Le polynôme localisateur d'erreur dont les racines sont Z_j^{-1} permet de déterminer la position des erreurs et le polynôme évaluateur d'erreur de déterminer la valeur de l'erreur e_j. En effet, en tenant compte du fait que $\Lambda(Z_j^{-1}) = 0$, le polynôme $\Gamma(x)$ pris en Z_j^{-1} est égal à :

$$\begin{aligned} \Gamma(Z_j^{-1}) &= e_j \prod_{p \neq j} (1 + Z_p Z_j^{-1}) \\ &= e_j Z_j^{-1} \Lambda'(Z_j^{-1}) \end{aligned}$$

où $\Lambda'(x) = \frac{d\Lambda}{dx}(x)$.
La valeur de l'erreur e_j est alors donnée par l'algorithme de Forney :

$$e_j = Z_j \frac{\Gamma(Z_j^{-1})}{\Lambda'(Z_j^{-1})} \tag{4.45}$$

En introduisant le polynôme $S(x)$ défini par :

$$S(x) = \sum_{j=1}^{2t} S_j x^j \tag{4.46}$$

on peut montrer que :

$$\Lambda(x)S(x) \equiv \Gamma(x) \text{ modulo } x^{2t+1} \tag{4.47}$$

Cette relation est appelée *l'équation clé* du décodage d'un code cyclique.
Pour déterminer les polynômes $\Lambda(x)$ et $\Gamma(x)$ deux algorithmes itératifs sont principalement utilisés, l'algorithme de Berlekamp-Massey et l'algorithme d'Euclide.

Algorithme de Berlekamp-Massey pour des codes à symboles non binaires

Le calcul des polynômes $\Lambda(x)$ et $\Gamma(x)$ selon l'algorithme de Berlekamp-Massey est réalisé de manière itérative. Il fait appel à deux polynômes intermédiaires notés $\Theta(x)$ et $\Omega(x)$. L'algorithme comporte $2t$ itérations. Une fois l'algorithme déroulé, il faut mettre en œuvre l'algorithme de Chien pour déterminer les racines Z_j^{-1}de $\Lambda(x)$ et par conséquent la position des erreurs. Ensuite l'algorithme de Forney exprimé par (4.45) permet de calculer la valeur des erreurs e_j.

Conditions initiales :

$$\begin{array}{rclcrcl} L_0 & = & 0 & & & & \\ \Lambda^{(0)}(x) & = & 1 & & \Theta^{(0)}(x) & = & 1 \\ \Gamma^{(0)}(x) & = & 0 & & \Omega^{(0)}(x) & = & 1 \end{array}$$

Récursion : $1 \leqslant p \leqslant 2t$

$$\Delta_p = \sum_j \Lambda_j^{(p-1)} S_{p-j}$$

$$\begin{array}{rcl} \delta_p & = & 1 \text{ si } \Delta_p \neq 0 \text{ et } 2L_{p-1} \leqslant p-1 \\ & = & 0 \text{ sinon} \end{array}$$

$$L_p = \delta_p(p - L_{p-1}) + (1-\delta_p)L_{p-1}$$

$$\begin{bmatrix} \Lambda^{(p)} & \Gamma^{(p)} \\ \Theta^{(p)} & \Omega^{(p)} \end{bmatrix} = \begin{bmatrix} 1 & \Delta_p x \\ \Delta_p^{-1}\delta_p & (1-\delta_p)x \end{bmatrix} \begin{bmatrix} \Lambda^{(p-1)} & \Gamma^{(p-1)} \\ \Theta^{(p-1)} & \Omega^{(p-1)} \end{bmatrix}$$

Terminaison :

$$\begin{array}{l} \Lambda(x) = \Lambda^{(2t)}(x) \\ \Gamma(x) = \Gamma^{(2t)}(x) \end{array}$$

Exemple 4.16

Pour illustrer le décodage d'un code RS à partir de l'algorithme de Berlekamp-Massey, considérons un code RS corrigeant jusqu'à deux erreurs ($t = 2$) et ayant les paramètres suivants :

$$m = 4; \quad q = 16; \quad n = 15; \quad n - k = 4$$

Supposons par exemple que le mot de code émis soit $c(x) = 0$ et que le mot reçu soit entaché de deux erreurs.

$$r(x) = \alpha^7 x^3 + \alpha^3 x^6$$

L'ensemble des calculs effectués pour décoder ce code RS sera fait dans le corps $\mathbf{F}_{16}$ dont les éléments sont donnés dans l'annexe 2.

1. Calcul du syndrome $\mathbf{S} = (S_1, S_2, S_3, S_4)$

$$\begin{array}{ll} S_1 = \alpha^{10} + \alpha^9 = \alpha^{13} & S_3 = \alpha^{16} + \alpha^{21} = \alpha^{11} \\ S_2 = \alpha^{13} + \alpha^{15} = \alpha^6 & S_4 = \alpha^{19} + \alpha^{27} = \alpha^6 \end{array}$$

   Le polynôme $S(x)$ est donc égal à :

$$S(x) = \alpha^{13} x + \alpha^6 x^2 + \alpha^{11} x^3 + \alpha^6 x^4$$

2. Calcul des polynômes $\Lambda(x)$ et $\Gamma(x)$ à partir de l'algorithme de Berlekamp-Massey

p	Δ_p	δ_p	L_p	$\Lambda^p(x)$	$\Theta^p(x)$	$\Gamma^p(x)$	$\Omega^p(x)$
0			0	1	1	0	1
1	α^{13}	1	1	$1 + \alpha^{13}x$	α^2	$\alpha^{13}x$	0
2	α	0	1	$1 + \alpha^8 x$	$\alpha^2 x$	$\alpha^{13}x$	0
3	α^{10}	1	2	$1 + \alpha^8 x + \alpha^{12}x^2$	$\alpha^5 + \alpha^{13}x$	$\alpha^{13}x$	$\alpha^3 x$
4	α^{10}	0	2	$1 + \alpha^2 x + \alpha^9 x^2$	$\alpha^5 x + \alpha^{13}x^2$	$\alpha^{13}x + \alpha^{13}x^2$	$\alpha^3 x^2$

   Dans le tableau ci-dessus, tous les calculs sont faits dans le corps $\mathbf{F}_{16}$ et en tenant compte du fait que $\alpha^{15} = 1$.

   Les polynômes localisateur d'erreur et évaluateur d'erreur sont :

$$\begin{array}{rcl} \Lambda(x) & = & 1 + \alpha^2 x + \alpha^9 x^2 \\ \Gamma(x) & = & \alpha^{13} x + \alpha^{13} x^2 \end{array}$$

   On peut vérifier que l'équation clé du décodage est bien satisfaite. En effet, nous avons bien :

$$\Lambda(x)S(x) = \alpha^{13}x + \alpha^{13}x^2 + \alpha^4 x^5 + x^6 \equiv \alpha^{13}x + \alpha^{13}x^2 \text{ modulo } x^5 = \Gamma(x) \text{ modulo } x^5$$

3. Recherche des racines du polynôme localisateur d'erreur

   En passant en revue tous les éléments du corps $\mathbf{F_{16}}$ on trouve que α^{12} et α^9 sont des racines du polynôme $\Lambda(x)$. Les erreurs sont donc en position $x^3(\alpha^{-12} = \alpha^3)$ et $x^6(\alpha^{-9} = \alpha^6)$ et le polynôme d'erreur $e(x)$ est égal à :

$$e(x) = e_3 x^3 + e_6 x^6$$

4. Calcul des coefficients d'erreur e_j (4.45).

$$\begin{array}{rcl} e_3 & = & \alpha^3 \frac{\alpha^6}{\alpha^2} = \alpha^7 \\ e_6 & = & \alpha^6 \frac{\alpha^{14}}{\alpha^2} = \alpha^3 \end{array}$$

Le polynôme d'erreur $e(x)$ est donc égal à :

$$e(x) = \alpha^7 x^3 + \alpha^3 x^6$$

et le mot de code estimé est $\hat{c}(x) = r(x) + e(x) = 0$. Les deux erreurs de transmission sont corrigées.

Algorithme d'Euclide

L'algorithme d'Euclide permet de résoudre l'équation clé du décodage c'est-à-dire de déterminer les polynômes $\Lambda(x)$ et $\Gamma(x)$.

Conditions initiales :

$$R_{-1}(x) = x^{2t};\ R_0(x) = S(x);\ U_{-1}(x) = 0;\ U_0(x) = 1$$

Récursion :

Calculer $Q_j(x)$, $R_{j+1}(x)$ et $U_{j+1}(x)$ à partir des deux expressions suivantes :

$$\begin{array}{l} \frac{R_{j-1}(x)}{R_j(x)} = Q_j(x) + \frac{R_{j+1}(x)}{R_j(x)} \\ U_{j+1}(x) = Q_j(x)U_j(x) + U_{j-1}(x) \end{array}$$

Lorsque $\deg(U_j) \leqslant t$ et $\deg(R_j) \leqslant t$ alors :

$$\begin{array}{l} \Lambda(x) = U_{j+1}(x) \\ \Gamma(x) = R_{j+1}(x) \end{array}$$

Exemple 4.17

Reprenons le code RS utilisé pour illustrer l'algorithme de Berlekamp-Massey. En supposant que le mot reçu est toujours $r(x) = \alpha^7 x^3 + \alpha^3 x^6$ lorsque le mot du code émis est $c(x) = 0$, l'algorithme de décodage est le suivant :

1. Calcul du syndrome $\mathbf{S} = (S_1, S_2, S_3, S_4)$

$$\begin{array}{ll} S_1 = \alpha^{10} + \alpha^9 = \alpha^{13} & S_3 = \alpha^{16} + \alpha^{21} = \alpha^{11} \\ S_2 = \alpha^{13} + \alpha^{15} = \alpha^6 & S_4 = \alpha^{19} + \alpha^{27} = \alpha^6 \end{array}$$

Le polynôme $S(x)$ est donc égal à :

$$S(x) = \alpha^{13}x + \alpha^6 x^2 + \alpha^{11} x^3 + \alpha^6 x^4$$

2. Calcul des polynômes $\Lambda(x)$ et $\Gamma(x)$ à partir de l'algorithme d'Euclide (les calculs sont effectués dans le corps $\mathbf{F_{16}}$ dont les éléments sont donnés dans l'annexe 2).

$$\begin{array}{rcl} j & = & 0 \\ R_{-1}(x) & = & x^5 \\ R_0(x) & = & S(x) \\ Q_0(x) & = & \alpha^9 x + \alpha^{14} \\ R_1(x) & = & \alpha^5 x^3 + \alpha^{13} x^2 + \alpha^{12} x \\ U_1(x) & = & \alpha^9 x + \alpha^{14} \end{array} \qquad \begin{array}{rcl} j & = & 1 \\ R_0(x) & = & S(x) \\ R_1(x) & = & \alpha^5 x^3 + \alpha^{13} x^2 + \alpha^{12} x \\ Q_1(x) & = & \alpha x + \alpha^5 \\ R_2(x) & = & \alpha^{14} x^2 + \alpha^{14} x \\ U_2(x) & = & \alpha^{10} x^2 + \alpha^3 x + \alpha \end{array}$$

On peut vérifier que $\deg(U_2(x)) = 2$ est inférieur ou égal à t ($t = 2$) et que $\deg(R_2(x)) = 2$ est inférieur ou égal à t. L'algorithme est donc terminé et les polynômes $\Lambda(x)$ et $\Gamma(x)$ ont respectivement pour expression :

$$\begin{array}{l} \Lambda(x) = U_2(x) = \alpha + \alpha^3 x + \alpha^{10} x^2 = \alpha(1 + \alpha^2 x + \alpha^9 x^2) \\ \Gamma(x) = R_2(x) = \alpha^{14} x + \alpha^{14} x^2 = \alpha(\alpha^{13} x + \alpha^{13} x^2) \end{array}$$

On peut vérifier que l'équation clé du décodage est bien satisfaite et que les deux polynômes obtenus sont identiques, à un coefficient α près à ceux déterminés avec l'algorithme de Berlekamp-Massey.

Les racines du polynôme $\Lambda(x)$ sont donc $1/\alpha^3$ et $1/\alpha^6$ et le polynôme d'erreur $e(x)$ est égal à :

$$e(x) = \alpha^7 x^3 + \alpha^3 x^6$$

Calcul des coefficients e_j par transformée

Il est possible de calculer les coefficients e_j; $j = 0, 1, \cdots, (n-1)$ du polynôme d'erreur $e(x)$ sans déterminer les racines du polynôme localisateur d'erreur $\Lambda(x)$. Pour cela on introduit le *syndrome prolongé* $S^*(x)$ défini par :

$$S^*(x) = \Gamma(x)\frac{1 + x^n}{\Lambda(x)} = \sum_{j=1}^{n} S_j x^j \tag{4.48}$$

Le coefficient e_j est nul (pas d'erreur) si α^{-j} n'est pas racine du polynôme localisateur d'erreur $\Lambda(x)$. Dans ce cas, nous avons $S^*(\alpha^{-j}) = 0$ puisque $\alpha^{-jn} = 1$ (se rappeler que $n = q - 1$ et $\alpha^{q-1} = 1$).

A contrario si α^{-j} est racine du polynôme localisateur, le coefficient e_j est non nul (présence d'une erreur) et $S^*(\alpha^{-j})$ est de la forme $0/0$. Cette indétermination peut être levée en calculant la dérivée du numérateur et du dénominateur de l'expression (4.48).

$$S^*(\alpha^{-i}) = \Gamma(\alpha^{-i})\frac{n\alpha^{-j(n-1)}}{\Lambda'(\alpha^{-j})}$$

En utilisant l'équation (4.45) et en tenant compte du fait que $\alpha^{-j(n-1)} = \alpha^j$ et que $na = a$ pour n impair dans un corps de Galois, le coefficient e_j est égal à :

$$e_j = S^*(\alpha^{-j}) \tag{4.49}$$

Le calcul du syndrome prolongé peut se faire à partir des polynômes $\Lambda(x)$ et $\Gamma(x)$ en utilisant la relation suivante déduite de l'expression (4.48).

$$\Lambda(x)S^*(x) = \Gamma(x)(1 + x^n) \tag{4.50}$$

Les coefficients S_j du syndrome prolongé sont identiques à ceux du syndrome $S(x)$ pour j allant de 1 à $2t$ et sont déterminés en annulant les coefficients des termes x^j dans le produit $\Lambda(x)S^*(x)$, pour j allant de $2t + 1$ à n.

Exemple 4.18

En reprenant l'exemple du code RS ($q = 16$; $n = 15$; $k = 11$; $t = 2$) utilisé pour illustrer l'algorithme de Berlekamp-Massey, déterminons le syndrome prolongé.

$$S^*(x) = \sum_{j=1}^{15} S_j x^j$$

avec :

$$\begin{array}{ll} S_1 = \alpha^{13} & S_3 = \alpha^{11} \\ S_2 = \alpha^6 & S_4 = \alpha^6 \end{array}$$

L'équation (4.50) nous fournit la relation suivante :

$$S(x) + \alpha^2 x S(x) + \alpha^9 x^2 S(x) = \alpha^{13}(x + x^2 + x^{16} + x^{17})$$

$$\begin{array}{l} S_1 x + (\alpha^2 S_1 + S_2)x^2 \\ + \sum_{k=3}^{15} (\alpha^9 S_{k-2} + \alpha^2 S_{k-1} + S_k)x^k \\ +(\alpha^2 S_{15} + \alpha^9 S_{14})x^{16} + \alpha^9 S_{15} x^{17} \quad = \quad \alpha^{13}(x + x^2 + x^{16} + x^{17}) \end{array}$$

Il en résulte la relation de récurrence :

$$S_k = \alpha^2 S_{k-1} + \alpha^9 S_{k-2}, \quad k = 3, 4, \cdots, 15$$

On obtient ainsi les coefficients du syndrome prolongé :

$$\begin{array}{l} S_5 = \alpha^4, S_6 = \alpha^{13}, S_7 = \alpha^6, S_8 = \alpha^{11}, S_9 = \alpha^6, S_{10} = \alpha^4 \\ S_{11} = \alpha^{13}, S_{12} = \alpha^6, S_{13} = \alpha^{11}, S_{14} = \alpha^6, S_{15} = \alpha^4 \end{array}$$

Une autre façon d'obtenir le polynôme prolongé consiste à diviser $\Gamma(x)(1+x^n)$ par $\Lambda(x)$ selon les puissances croissantes.

Les erreurs étant au niveau des monômes x^3 et x^6, calculons les coefficients e_3 et e_6.

$$\begin{array}{rcl} e_3 & = & S^*(\alpha^{12}) = \alpha^2 + \alpha^4 + \alpha^{10} + \alpha^7 = \alpha^7 \\ e_6 & = & S^*(\alpha^9) = \alpha^4 + \alpha^7 = \alpha^3 \end{array}$$

Les valeurs trouvées pour les coefficients e_3 et e_6 sont évidemment identiques à celles obtenues dans l'exemple 4.16. On pourra vérifier que les autres coefficients e_j sont tous nuls.

Algorithme de Berlekamp-Massey pour les codes cycliques binaires

Pour les codes BCH binaires l'algorithme de Berlekamp-Massey peut se simplifier puisqu'il n'est plus nécessaire de déterminer le polynôme évaluateur d'erreur, et qu'il est possible de montrer que les termes Δ_j sont nuls pour j pair. Ceci implique :

$$\begin{array}{l} \delta_{2p} = 0 \\ L_{2p} = L_{2p-1} \\ \Lambda^{(2p)}(x) = \Lambda^{(2p-1)}(x) \\ \Theta^{(2p)}(x) = x\Theta^{(2p-1)}(x) \end{array}$$

D'où l'algorithme en t itérations :
Conditions initiales :

$$\begin{array}{rcl} L_{-1} & = & 0 \\ \Lambda^{(-1)}(x) & = & 1 \quad \Theta^{(-1)}(x) = x^{-1} \end{array}$$

Récursion : $0 \leqslant p \leqslant t-1$

$$\Delta_{2p+1} = \sum_j \Lambda_j^{(2p-1)} S_{2p+1-j}$$

$$\begin{array}{rcl} \delta_{2p+1} & = & 1 \quad \text{si } \Delta_{2p+1} \neq 0 \text{ et } L_{2p-1} \leqslant p \\ & = & 0 \quad \text{sinon} \end{array}$$

$$L_{2p+1} = \delta_{2p+1}(2p+1-L_{2p-1}) + (1-\delta_{2p+1})L_{2p-1}$$

$$\begin{bmatrix} \Lambda^{(2p+1)} \\ \Theta^{(2p+1)} \end{bmatrix} = \begin{bmatrix} 1 & \Delta_{2p+1}x^2 \\ \Delta_{2p+1}^{-1}\delta_{2p+1} & (1-\delta_{2p+1})x^2 \end{bmatrix} \begin{bmatrix} \Lambda^{(2p-1)} \\ \Theta^{(2p-1)} \end{bmatrix}$$

Terminaison :

$$\Lambda(x) = \Lambda^{(2t-1)}(x)$$

Exemple 4.19

Reprenons le code BCH qui a été utilisé pour illustrer le calcul du polynôme localisateur d'erreur avec la méthode directe. Supposons que le mot reçu soit $r(x) = x^8 + x^3$ lorsque le mot de code émis est $c(x) = 0$.

1. Le syndrome **S** possède quatre composantes.

$$S_1 = r(\alpha) = \alpha^8 + \alpha^3 = \alpha^{13}$$
$$S_3 = r(\alpha^3) = \alpha^{24} + \alpha^9 = 0$$
$$S_2 = S_1^2 = \alpha^{26} = \alpha^{11}$$
$$S_4 = S_2^2 = \alpha^{22} = \alpha^7$$

   Le polynôme $S(x)$ est égal à :

$$S(x) = \alpha^{13}x + \alpha^{11}x^2 + \alpha^7 x^4$$

2. Calcul du polynôme $\Lambda(x)$ à partir de l'algorithme de Berlekamp-Massey

p	Δ_{2p+1}	δ_{2p+1}	L_{2p+1}	$\Lambda^{2p+1}(x)$	$\Theta^{2p+1}(x)$
-1			0	1	x^{-1}
0	α^{13}	1	1	$1 + \alpha^{13}x$	α^2
1	α^9	1	2	$1 + \alpha^{13}x + \alpha^{11}x^2$	$\alpha^6 + \alpha^4 x$

   On notera que le polynôme $\Lambda(x)$ obtenu est identique à celui déterminé à partir de la méthode directe. Les racines de $\Lambda(x)$ sont $1/\alpha^3$ et $1/\alpha^8$, les erreurs portent donc sur les termes x^3 et x^8. Le mot de code estimé est $\hat{c}(x) = 0$.

Algorithme d'Euclide pour les codes binaires

Exemple 4.20
Reprenons le décodage du code BCH(15,7). Le mot reçu est $r(x) = x^8 + x^3$.

$$\begin{array}{rcc@{\qquad}rcc} j & = & 0 & j & = & 1 \\ R_{-1}(x) & = & x^5 & R_0(x) & = & S(x) \\ R_0(x) & = & S(x) & R_1(x) & = & \alpha^4 x^3 + \alpha^6 x^2 \\ Q_0(x) & = & \alpha^8 x & Q_1(x) & = & \alpha^3 x + \alpha^5 \\ R_1(x) & = & \alpha^4 x^3 + \alpha^6 x^2 & R_2(x) & = & \alpha^{13} x \\ U_1(x) & = & \alpha^8 x & U_2(x) & = & \alpha^{11} x^2 + \alpha^{13} x + 1 \end{array}$$

On peut vérifier que $\deg(U_2(x)) = 2$ est inférieur ou égal à t ($t = 2$) et que le degré de R_2 est inférieur ou égal à t. L'algorithme est donc terminé et le polynôme $\Lambda(x)$ a pour expression :

$$\Lambda(x) = U_2(x) = 1 + \alpha^{13}x + \alpha^{11}x^2$$
$$\Gamma(x) = R_2(x) = \alpha^{13}x$$

Pour un code BCH binaire il n'est pas nécessaire d'utiliser le polynôme évaluateur d'erreur pour déterminer la valeur des coefficients e_3 et e_8. Toutefois, on peut vérifier que :

$$e_3 = \alpha^3 \frac{\Gamma(\alpha^{-3})}{\Lambda'(\alpha^{-3})} = 1$$
$$e_8 = \alpha^8 \frac{\Gamma(\alpha^{-8})}{\Lambda'(\alpha^{-8})} = 1$$

Le mot décodé est donc $\hat{c}(x) = r(x) + e(x) = 0$ et les deux erreurs sont bien corrigées.

4.4.4 Performances du décodage à entrée ferme des codes de Reed-Solomon

Rappelons que pour un code de Reed-Solomon, les blocs d'information à coder et les mots de code sont constitués respectivement de k et $n = q - 1$ ($q = 2^m$) symboles q-aires. La probabilité $P_{e,\text{mot}}$ d'avoir un mot de code faux après un décodage ferme peut être bornée supérieurement par :

$$P_{e,\text{mot}} \leqslant \sum_{j=t+1}^{n} \left(\begin{array}{c} n \\ j \end{array} \right) p_s^j (1-p_s)^{n-j} \tag{4.51}$$

où p_s est la probabilité d'erreur par symbole q-aire sur le canal de transmission, t est le pouvoir de correction du code en nombre de symboles q-aires.
Lorsqu'un mot de code est mal décodé, la probabilité d'erreur par symbole $P_{e,\text{symbole}}$ correspondante après décodage est égale à :

$$P_{e,\text{symbole}} \leqslant \frac{1}{n} \sum_{j=t+1}^{n} (j+t) \left(\begin{array}{c} n \\ j \end{array} \right) p_s^j (1-p_s)^{n-j} \tag{4.52}$$

La probabilité d'erreur binaire après décodage s'obtient à partir de la probabilité d'erreur par symbole en tenant compte que un symbole est représenté par m éléments binaires.

$$P_{e,\text{bit}} = 1 - (1 - P_{e,\text{symbole}})^{\frac{1}{m}}$$

À fort rapport signal à bruit, on peut approcher la probabilité d'erreur par élément binaire après décodage.

$$P_{\text{e,bit}} \cong \frac{1}{m} P_{\text{e,symbole}} \quad \frac{E_b}{N_0} >> 1$$

4.5 Bibliographie

[4.1] J. G. Proakis, *Digital Communications*, 4$^{\text{th}}$ edition. McGraw-Hill, New-York, 2000.

[4.2] R.H. Morelos-Zaragoza, *The Art of Error Correcting Coding*, John Wiley & sons, 2005.

Annexe

Notions sur les corps de Galois et sur les polynômes minimaux

Définition

Un corps de Galois à $q = 2^m$ éléments noté $\mathbf{F}_q$, où m est un entier positif est défini comme une extension polynomiale du corps à deux éléments $(0, 1)$ noté $\mathbf{F}_2$. Le polynôme $\varphi(x)$ utilisé pour construire le corps $\mathbf{F}_q$ doit être

- irréductible c'est-à-dire être non factorisable dans $\mathbf{F}_2$ (autrement dit 0 et 1 ne sont pas racines de $\varphi(x)$,
- de degré m,
- et à coefficients dans $\mathbf{F}_2$.

Les éléments du corps de Galois $\mathbf{F}_q$ sont définis modulo $\varphi(x)$ et ainsi, chaque élément de ce corps peut être représenté par un polynôme de degré au plus égal à $(m-1)$ et à coefficients dans $\mathbf{F}_2$.

Exemple 1

Considérons un polynôme irréductible $\varphi(x)$ dans le corps $\mathbf{F}_2$ de degré $m = 2$.

$$\varphi(x) = x^2 + x + 1$$

Ce polynôme permet de construire un corps de Galois à 4 éléments. Les éléments de ce corps $\mathbf{F}_4$ sont de la forme :

$$a\alpha + b \quad \text{où} \quad a, b \in \mathbf{F_2}$$

soit :

$$F_4 = \{0,\, 1,\, \alpha,\, \alpha + 1\}$$

On peut remarquer que si on élève l'élément α aux puissances successives 0, 1 et 2 on obtient tous les éléments du corps $\mathbf{F}_4$ à l'exception de l'élément 0. En effet, α^2 est encore égal à $(\alpha + 1)$ modulo $\varphi(\alpha)$. L'élément α est appelé *élément primitif* du corps $\mathbf{F}_4$.

Les éléments du corps $\mathbf{F}_4$ peuvent aussi être représentés sous forme binaire :

$$\mathbf{F}_4 : \{00,\ 01,\ 10,\ 11\}$$

Les couples binaires correspondent aux quatre valeurs prises par les coefficients a et b.

Élément primitif d'un corps de Galois

On appelle élément primitif d'un corps de Galois $\mathbf{F}_q$, un élément de ce corps qui, lorsqu'il est élevé aux puissances successives $0, 1, 2, \cdots, (q-2)$; $q = 2^m$, permet de retrouver tous les éléments du corps sauf l'élément 0. Tout corps de Galois possède au moins un élément primitif. Si α est un élément primitif du corps $\mathbf{F}_q$ alors, les éléments de ce corps sont :

$$\mathbf{F}_q = \left\{0, \alpha^0, \alpha^1, \cdots, \alpha^{q-2}\right\} \text{ avec } \alpha^{q-1} = 1$$

Notons que dans un tel corps de Galois le signe « - » est équivalent au signe « + » soit :

$$-\alpha^j = \alpha^j \quad \forall j \in \{0, 1, \cdots, (q-2)\}$$

En observant que $2\alpha^j = 0$ modulo 2, on peut toujours ajouter à $-\alpha^j$ la quantité nulle $2\alpha^j$ et on obtient ainsi l'égalité ci-dessus.

Donnons pour le corps $\mathbf{F}_4$, à titre d'exemple, les règles régissant les opérations d'addition et de multiplication. Toutes les opérations sont faites modulo 2 et modulo $\alpha^2 + \alpha + 1$.

$+$	$\mathbf{0}$	$\mathbf{1}$	α	α^2
$\mathbf{0}$	0	1	α	α^2
$\mathbf{1}$	1	0	$1+\alpha=\alpha^2$	$1+\alpha^2=\alpha$
α	α	$1+\alpha=\alpha^2$	0	$\alpha+\alpha^2=1$
α^2	α^2	$1+\alpha^2=\alpha$	$\alpha+\alpha^2=1$	0

Table 4.7 – Addition dans le corps $\mathbf{F}_4$.

$\times$	$\mathbf{0}$	$\mathbf{1}$	α	α^2
$\mathbf{0}$	0	0	0	0
$\mathbf{1}$	0	1	α	α^2
α	0	α	α^2	$\alpha^3=1$
α^2	0	α^2	$\alpha^3=1$	$\alpha^4=\alpha$

Table 4.8 – Multiplication dans le corps $\mathbf{F}_4$.

Polynôme minimal à coefficients dans $\mathbf{F_2}$ associé à un élément d'un corps de Galois $\mathbf{F}_q$

Le polynôme minimal $m_\beta(x)$ à coefficients dans $\mathbf{F_2}$ associé à un élément quelconque β d'un corps de Galois $\mathbf{F}_q$, est un polynôme de degré au plus égal à $m = \log_2(q)$, ayant β comme racine. Ce polynôme est unique et irréductible dans $\mathbf{F_2}$. Si β est un élément primitif du corps de Galois $\mathbf{F}_q$ alors le polynôme $m_\beta(x)$ est exactement de degré m. Remarquons qu'un polynôme à coefficients dans $\mathbf{F_2}$ vérifie la propriété suivante :

$$[f(x)]^2 = f(x^2) \Rightarrow [f(x)]^{2^p} = f(x^{2^p})$$

Ainsi si β est racine du polynôme $f(x)$ alors $\beta^2, \beta^4, \cdots$ sont aussi des racines de ce polynôme. Le polynôme minimal à coefficients dans $\mathbf{F_2}$ ayant β comme racine peut donc s'écrire sous la forme :

$$m_\beta(x) = (x+\beta)(x+\beta^2)(x+\beta^4)\cdots$$

Si β est un élément primitif de $\mathbf{F}_q$, le polynôme minimal à coefficients dans $\mathbf{F_2}$ étant de degré m, il peut aussi s'écrire :

$$m_\beta(x) = (x+\beta)(x+\beta^2)(x+\beta^4)\cdots(x+\beta^{2^{m-1}})$$

Exemple 2

Calculons le polynôme minimal associé à l'élément primitif α du corps de Galois $\mathbf{F}_4$.

$$\mathbf{F}_4 : \{0,\, 1,\, \alpha,\, \alpha^2\}$$

Le polynôme minimal associé à l'élément α a donc pour racines α et α^2 ($m = 2$) et a pour expression :

$$m_\alpha(x) = (x+\alpha)(x+\alpha^2) = x^2 + x(\alpha+\alpha^2) + \alpha^3$$

En tenant compte du fait que $\alpha^3 = 1$ et que $\alpha + \alpha^2 = 1$ dans le corps $\mathbf{F}_4$, le polynôme $m_\alpha(x)$ est encore égal à :

$$m_\alpha(x) = x^2 + x + 1$$

Polynôme minimal à coefficients dans $\mathbf{F}_q$ associé à un élément d'un corps de Galois $\mathbf{F}_q$

Le polynôme minimal $m_\beta(x)$, à coefficients dans le corps de Galois $\mathbf{F}_q$ associé à un élément $\beta = \alpha^j$ (α élément primitif du corps $\mathbf{F}_q$) de ce corps, est le polynôme de plus bas degré ayant β comme racine.

En rappelant que pour un polynôme à coefficients dans $\mathbf{F}_q$, nous pouvons écrire :

$$[f(x)]^q = f(x^q) \Rightarrow [f(x)]^{q^p} = f(x^{q^p})$$

Alors si β est racine du polynôme $f(x)$, $\beta^q, \beta^{q^2}, \cdots$ sont aussi des racines de ce polynôme.
Puisque dans le corps $\mathbf{F}_q \quad \alpha^{q-1} = 1$, alors $\beta^{q^p} = (\alpha^j)^{q^p} = \alpha^j = \beta$ et ainsi, le polynôme minimal $m_\beta(x)$ est simplement égal à :

$$m_\beta(x) = x + \beta$$

Ces résultats sur les polynômes minimaux sont utilisés pour déterminer les polynômes générateurs de codes cycliques particuliers (BCH et Reed-Solomon).

Polynôme primitif

Un polynôme à coefficients dans $\mathbf{F}_2$ est primitif si il est le polynôme minimal associé à un élément primitif d'un corps de Galois. Un polynôme primitif est donc irréductible dans $\mathbf{F}_2$ et peut par conséquent être utilisé pour engendrer un corps de Galois. Lorsqu'un polynôme primitif est utilisé pour engendrer un corps de Galois, tous les éléments du corps sont obtenus en élevant l'élément primitif, racine du polynôme primitif, à des puissances successivement croissantes. Les principaux polynômes primitifs étant répertoriés dans la littérature, la construction d'un corps de Galois à $q = 2^m$ éléments peut alors se faire simplement en utilisant un polynôme primitif de degré m. La table 4.9 donne quelques polynômes primitifs.

Degré du polynôme	**Polynôme primitif**
2	$\alpha^2 + \alpha + 1$
3	$\alpha^3 + \alpha + 1$
4	$\alpha^4 + \alpha + 1$
5	$\alpha^5 + \alpha^2 + 1$
6	$\alpha^6 + \alpha + 1$
7	$\alpha^7 + \alpha^3 + 1$
8	$\alpha^8 + \alpha^4 + \alpha^3 + \alpha^2 + 1$
9	$\alpha^9 + \alpha^4 + 1$
10	$\alpha^{10} + \alpha^3 + 1$

Table 4.9 – Exemples de polynômes primitifs

Pour terminer cette introduction aux corps de Galois et aux polynômes minimaux, donnons un exemple de corps de Galois à $q = 16$ ($m = 4$) éléments construit à partir du polynôme primitif $x^4 + x + 1$. Ce corps est utilisé pour construire des polynômes générateurs de codes BCH et de Reed-Solomon et pour les décoder. Les éléments de ce corps sont :

$$\mathbf{F}_{16} = \left\{0,\, 1,\, \alpha,\, \alpha^2,\, \alpha^3 \cdots \alpha^{14}\right\}$$

où α est un élément primitif de $\mathbf{F}_{16}$. À ces 16 éléments, on peut également associer une représentation polynomiale ainsi qu'une représentation binaire. La représentation polynomiale d'un élément de ce corps est de la forme :

$$a\alpha^3 + b\alpha^2 + c\alpha + d$$

où a, b, c et d sont des coefficients binaires appartenant à $\mathbf{F}_2$.

Le corps de Galois $\mathbf{F}_{16}$ étant constitué de 16 éléments, la représentation binaire d'un élément de ce corps est faite à l'aide de 4 symboles binaires appartenant à $\mathbf{F}_2$. Ces 4 symboles sont respectivement égaux aux valeurs prises par les coefficients a, b, c et d.

Éléments du corps	**Représentation polynomiale**	**Représentation binaire**
0	0	0 0 0 0
1	1	0 0 0 1
α	α	0 0 1 0
α^2	α^2	0 1 0 0
α^3	α^3	1 0 0 0
α^4	$\alpha + 1$	0 0 1 1
α^5	$\alpha^2 + \alpha$	0 1 1 0
α^6	$\alpha^3 + \alpha^2$	1 1 0 0
α^7	$\alpha^3 + \alpha + 1$	1 0 1 1
α^8	$\alpha^2 + 1$	0 1 0 1
α^9	$\alpha^3 + \alpha$	1 0 1 0
α^{10}	$\alpha^2 + \alpha + 1$	0 1 1 1
α^{11}	$\alpha^3 + \alpha^2 + \alpha$	1 1 1 0
α^{12}	$\alpha^3 + \alpha^2 + \alpha + 1$	1 1 1 1
α^{13}	$\alpha^3 + \alpha^2 + 1$	1 1 0 1
α^{14}	$\alpha^3 + 1$	1 0 0 1

Table 4.10 – Différentes représentations des éléments du corps de Galois $\mathbf{F}_{16}$

Exemple 3

Quelques calculs dans le corps $\mathbf{F}_{16}$:

$+$	α^2	α^4
α^8	$0100 + 0101 = 0001 = 1$	$0011 + 0101 = 0110 = \alpha^5$
α^{10}	$0100 + 0111 = 0011 = \alpha^4$	$0011 + 0111 = 0100 = \alpha^2$

Table 4.11 – Addition dans $\mathbf{F}_{16}$

$\times$	α^2	α^6
α^8	α^{10}	α^{14}
α^{14}	$\alpha^{16} = \alpha$ car $\alpha^{15} = 1$	$\alpha^{20} = \alpha^5$ car $\alpha^{15} = 1$

Table 4.12 – Multiplication dans $\mathbf{F}_{16}$

$\div$	α^2	α^{12}
α^8	$\alpha^{-6} = \alpha^9$ car $\alpha^{15} = 1$	α^4
α^{14}	$\alpha^{-12} = \alpha^3$ car $\alpha^{15} = 1$	$\alpha^{-2} = \alpha^{13}$ car $\alpha^{15} = 1$

Table 4.13 – Division dans $\mathbf{F}_{16}$

Chapitre 5

Les codes convolutifs et leur décodage

5.1 Historique

C'est en 1955 que Peter Elias introduit la notion de code convolutif [5.1]. L'exemple de codeur décrit dans la publication est reproduit dans la figure 5.1. Il s'agit d'un codeur systématique, c'est-à-dire que le message codé contient le message à transmettre, auquel est ajouté de l'information redondante. Le message est de longueur infinie, ce qui à première vue limite le champ d'application de ce type de codes. Il est cependant aisé de les adapter pour des transmissions de paquets grâce à des techniques de *fermeture de treillis*.

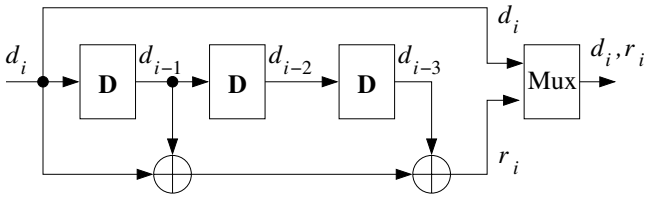


Figure 5.1 – Exemple de codeur convolutif.

Le codeur présenté en figure 5.1 est conçu autour d'un registre à décalage de trois éléments mémoire. Le bit de redondance à l'instant i, noté r_i est construit à l'aide d'une somme modulo 2 de la donnée à l'instant i, d_i et des données présentes aux instants $i-1$ et $i-3$ (d_{i-1} et d_{i-3}). Un multiplexeur joue le rôle d'un convertisseur parallèle-série et fournit le résultat du codage à un débit double de celui de l'entrée. Le rendement de ce codeur est de 1/2 car, à chaque instant i, il reçoit une donnée d_i et délivre deux éléments en sortie : d_i (partie systématique) et r_i (partie redondante).

Il faut attendre 1957 pour voir apparaître le premier algorithme capable de décoder de tels codes. Inventé par Wozencraft [5.2], cet algorithme, appelé décodage séquentiel, est ensuite amélioré par Fano [5.3] en 1963. Quatre ans plus tard, Viterbi introduit un nouvel algorithme particulièrement intéressant lorsque la longueur du registre à décalage du codeur n'est pas trop grande [5.4]. En effet, la complexité de l'algorithme de Viterbi croît de manière exponentielle avec la taille de ce registre alors que la complexité de l'algorithme de Fano en est quasi-indépendante.

En 1974, Bahl, Cocke, Jelinek et Raviv présentent un nouvel algorithme [5.5] capable d'associer une probabilité à la décision binaire. Cette propriété est très largement utilisée dans le décodage des codes concaténés et plus particulièrement des turbocodes, qui ont remis cet algorithme au goût du jour. Il est maintenant référencé dans la littérature sous une des trois dénominations : *BCJR* (initiales des inventeurs), *MAP* (*Maximum A Posteriori*) ou (*A Posteriori Probability*) *APP*. Plutôt complexe à mettre en oeuvre dans sa version initiale, l'algorithme *MAP* existe sous des versions simplifiées, dont les plus courantes sont présentées dans le chapitre 7.

En parallèle avec ces avancées sur les algorithmes de décodage, nombre de travaux ont porté sur la construction des codeurs convolutifs. La diminution de la complexité du codeur n'était pas l'objet de ces travaux, car son implantation est triviale. L'enjeu est de trouver des codes dont le pouvoir de correction est le plus élevé possible. En 1970, Forney écrit un papier de référence sur l'algèbre des codes convolutifs [5.6]. Il y démontre qu'un *bon* code convolutif n'est pas nécessairement systématique et suggère une construction différente de celle de la figure 5.1 Ce papier éloigne, pour un temps, les codes convolutifs systématiques du champ de la recherche en codage de canal.

La figure 5.2 donne un exemple de codeur convolutif non-systématique. Contrairement au codeur de la figure 5.1, les données ne sont pas présentes à la sortie du codeur et sont remplacées par une somme modulo 2 de la donnée à l'instant i, d_i, et des données présentes aux instants $i-2$ et $i-3$ (d_{i-2} et d_{i-3}). Le rendement du codeur reste inchangé à $1/2$ car le codeur fournit toujours deux éléments en sortie : $r_i^{(1)}$ et $r_i^{(2)}$, à l'instant i.

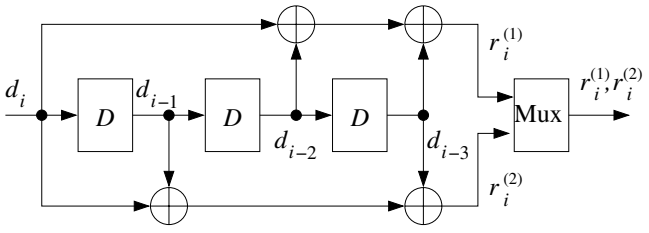


Figure 5.2 – Exemple de codeur de code convolutif non-systématique.

Lorsque Berrou *et al* présentent leurs travaux sur les turbocodes [5.7], ils réhabilitent les codes convolutifs systématiques en les utilisant sous une forme

récursive. L'intérêt des codes récursifs est présenté dans les sections 5.2 et 5.3. La figure 5.3 donne un exemple de codeur de code convolutif systématique récursif. Le message d'origine étant transmis (d_i), le code est donc bien systématique. Une boucle de rétroaction apparaît, la structure du codeur étant maintenant analogue à celle des générateurs de séquences pseudo-aléatoires.

Ce rapide historique a permis de présenter les trois familles de codes convolutifs les plus courantes : systématique, non-systématique, systématique récursif. Les deux sections suivantes abordent la représentation et la performance des codes convolutifs. Elles donnent l'occasion de comparer les propriétés de ces trois familles. Les algorithmes de décodage les plus couramment utilisés dans les systèmes actuels sont présentés dans la section 5.4. Enfin, la section 5.5 aborde les principales techniques de *fermeture de treillis* et de*poinçonnage*.

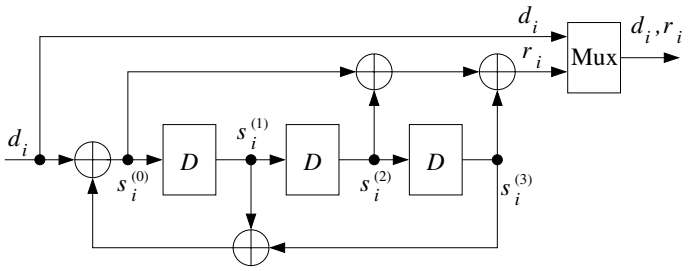


Figure 5.3 – Exemple de codeur de code convolutif systématique récursif.

5.2 Représentations des codes convolutifs

Ce chapitre n'a pas la prétention de traiter de manière exhaustive le thème des codes convolutifs. Les codes non-binaires ou non-linéaire ainsi que les codeurs à plusieurs registres ne sont pas traités. Seuls les codes les plus couramment utilisés, en particulier pour la construction de turbocodes, sont introduits. Le lecteur souhaitant approfondir le sujet peut, par exemple, se référer à [5.8].

5.2.1 Représentation générique d'un codeur convolutif

La figure 5.4 donne un modèle suffisamment général pour que l'on puisse représenter l'ensemble des codes convolutifs étudiés dans ce chapitre. A chaque instant i, il reçoit, en entrée, un vecteur $\mathbf{d}_i$ de m bits. Le code ainsi généré est un code binaire. Cependant pour la simplification de l'écriture nous l'appellerons code *m-binaire* et *double-binaire* si $m = 2$ (comme dans l'exemple présenté en figure 5.5). Lorsque $c = 1$, le code généré est systématique, puisque $\mathbf{d}_i$ est transmis en sortie du codeur. Le code est donc composé de la partie systématique $\mathbf{d}_i$ sur m bits et de la redondance $\mathbf{r}_i$ sur n bits. Le rendement du

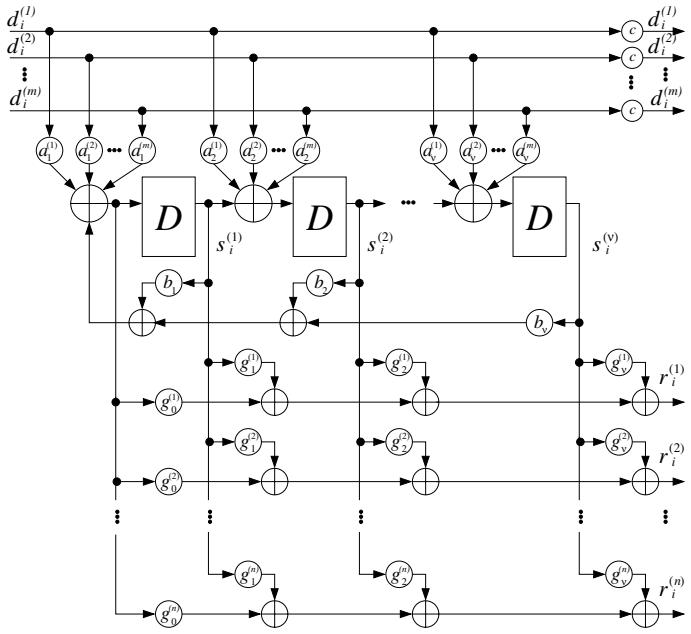


Figure 5.4 – Représentation générique d'un codeur de code convolutif.

code est alors $R = m/(m+n)$. Si $c = 0$, le code est non-systématique et le rendement devient $R = m/n$.

La partie non-systématique est construite à l'aide d'un registre à décalage composé de ν bascules et d'additionneurs binaires, autrement dit de portes « ou exclusif » (XOR en anglais). On définit alors une caractéristique importante du code convolutif : la *longueur de contrainte* égale ici à $\nu + 1$ (certains auteurs la notent ν, ce qui implique que le registre est alors composé de $\nu - 1$ bascules). Le registre à l'instant i est caractérisé par les ν bits $s_i^{(1)}, s_i^{(2)}, \ldots, s_i^{\nu}$ mémorisés : ils définissent son *état*, que l'on peut donc coder sur ν bits et représenter sous forme d'un vecteur $\mathbf{s}_i = (s_i^{(1)}, s_i^{(2)}, \cdots, s_i^{\nu})$. Un tel codeur convolutif possède donc 2^{ν} valeurs possibles d'état, que l'on note souvent sous forme binaire naturelle ou décimale. Ainsi l'état d'un codeur constitué de trois bascules pourra prendre $2^3 = 8$ valeurs. Si $s_1 = 1$, $s_2 = 1$ et $s_3 = 0$, le codeur est dans l'état 110 en binaire naturel soit 6 en décimal.

Par l'intermédiaire des coefficients $a_j^{(l)}$, chacune des m composantes du vecteur $\mathbf{d}_i$ est sélectionnée ou non comme terme d'une addition avec le contenu d'une bascule précédente (sauf dans le cas de la première bascule) pour fournir la valeur à stocker dans la bascule suivante. Le nouveau contenu d'une bascule dépend donc de l'entrée courante et du contenu de la bascule précédente. Le cas de la première bascule est à considérer différemment. Si tous les coefficients b_j sont nuls, l'entrée est le résultat de la somme des seules composantes sélec-

tionnées de $\mathbf{d}_i$. Dans le cas contraire, à la somme des composantes sélectionnées de $\mathbf{d}_i$ s'ajoutent les contenus des bascules sélectionnées par les coefficients b_j non-nuls. Le code ainsi engendré est récursif. Ainsi, la succession des états du registre dépend donc de l'état de départ et de la succession des données en entrée. Les composantes de la redondance $\mathbf{r}_i$ sont finalement produites en sommant le contenu des bascules sélectionnées par les coefficients g.

Considérons quelques exemples.

— Le codeur représenté en figure 5.1 est binaire systématique, donc $m = 1$ et $c = 1$. De plus, tous les coefficients $a_j^{(l)}$ sont nuls sauf $a_1^{(1)} = 1$. Ce codeur n'est pas récursif car tous les coefficients b_j sont nuls. Le bit de redondance (ou parité) est défini par $g_0^{(1)} = 1$, $g_1^{(1)} = 1$, $g_2^{(1)} = 0$ et $g_3^{(1)} = 1$.

— Dans le cas du codeur binaire non-systématique non-récursif (ici dénommé « classique ») de la figure 5.2, $m = 1$, $c = 0$; parmi les $a_j^{(l)}$, seul $a_1^{(1)} = 1$ est non-nul et $b_j = 0 \quad \forall j$. Deux bits de parité sont issus du codeur et définis par $g_0^{(1)} = 1$, $g_1^{(1)} = 1$, $g_2^{(1)} = 0$, $g_3^{(1)} = 1$ et $g_0^{(2)} = 1$, $g_1^{(2)} = 0$, $g_2^{(2)} = 1$, $g_3^{(2)} = 1$.

— La figure 5.3 présente un codeur binaire systématique ($m = 1$, $c = 1$ et $a_1^{(1)} = 1$) récursif. Les coefficients de la boucle de récursivité sont alors $b_1 = 1$, $b_2 = 0$, $b_3 = 1$ et ceux de la redondance sont $g_0^{(1)} = 1$, $g_1^{(1)} = 0$, $g_2^{(1)} = 1$, $g_3^{(1)} = 1$.

— La figure 5.5 représente un codeur double-binaire systématique récursif. Les seuls coefficients qui diffèrent du précédent cas sont les $a_j^{(l)}$: les coefficients $a_1^{(1)}$, $a_1^{(2)}$, $a_2^{(2)}$ et $a_3^{(2)}$ sont égaux à 1, les autres $a_j^{(l)}$ sont nuls.

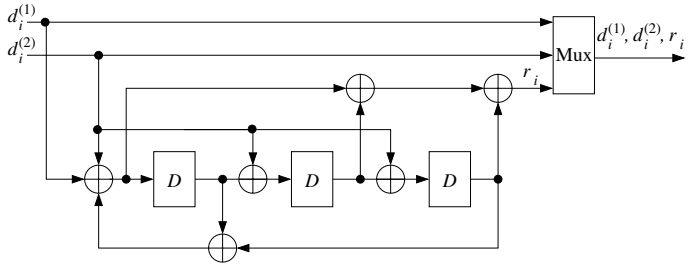


Figure 5.5 – Exemple de codeur convolutif double-binaire systématique récursif.

Pour définir un codeur, il n'est cependant pas nécessaire d'en faire une représentation graphique puisque la connaissance des paramètres présentés en figure 5.4 est suffisante. Une représentation condensée de ces paramètres est connue sous le nom de polynômes générateurs. Cette notation est présentée dans le paragraphe suivant.

5.2.2 Représentation polynomiale

Considérons tout d'abord un code binaire classique (non-systématique et non-récursif) : $c = 0$, tous les coefficients b_j sont nuls et $m = 1$. La connaissance de $\left[g_j^{(k)}\right]_{j=0..\nu}^{k=1..n}$ suffit alors à décrire le code. Les coefficients $\left[g_j^{(k)}\right]_{j=0..\nu}$ définissent donc n polynômes générateurs $G^{(k)}$ en algèbre de D (*Delay*) :

$$G^{(k)}(D) = \sum_{j=1...n} g_j^{(k)} D^j \tag{5.1}$$

Prenons le cas du codeur défini en figure 5.2. Les sorties $r_i^{(1)}$ et $r_i^{(2)}$ s'expriment en fonction des données d successives de la manière suivante :

$$r_i^{(1)} = d_i + d_{i-2} + d_{i-3} \tag{5.2}$$

ce qui peut aussi s'écrire, via la transformée en D :

$$r^{(1)}(D) = G^{(1)}(D)\ d(D) \tag{5.3}$$

avec $G^{(1)}(D) = 1 + D^2 + D^3$, premier polynôme générateur du code et $d(D)$ transformée en D du message à coder. De même, le second polynôme générateur est $G^2(D) = 1 + D + D^3$.

Ces polynômes générateurs peuvent aussi se résumer par la suite de leurs coefficients, respectivement (1011) et (1101), généralement notée en représentation octale, respectivement $(13)_{octal}$ et $(15)_{octal}$. Dans le cas d'un code systématique non-récursif, comme l'exemple en figure 5.1, les polynômes générateurs s'expriment selon le même principe. Dans cet exemple, le codeur a pour polynômes générateurs $G^{(1)}(D) = 1$ et $G^{(2)}(D) = 1 + D + D^3$.

Définir les polynômes générateurs d'un code systématique récursif est moins évident. Considérons l'exemple de la figure 5.3. Le premier polynôme générateur est trivial puisque le code est systématique. Pour identifier le second, il faut noter que

$$s_i^{(0)} = d_i + s_i^{(1)} + s_i^{(3)} = d_i + s_{i-1}^{(0)} + s_{i-3}^{(0)} \tag{5.4}$$

et que :

$$r_i = s_i^{(0)} + s_i^{(2)} + s_i^{(3)} = s_i^{(0)} + s_{i-2}^{(0)} + s_{i-3}^{(0)} \tag{5.5}$$

ce qui est équivalent à :

$$\begin{array}{l} d_i = s_i^{(0)} + s_{i-1}^{(0)} + s_{i-3}^{(0)} \\ r_i = s_i^{(0)} + s_{i-2}^{(0)} + s_{i-3}^{(0)} \end{array} \tag{5.6}$$

Ce résultat peut être reformulé en introduisant la transformée en D :

$$\begin{array}{l} d(D) = G^{(2)}(D)s(D) \\ r(D) = G^{(1)}(D)s(D) \end{array} \tag{5.7}$$

où $G^{(1)}(D)$ et $G^{(2)}(D)$ sont les polynômes générateurs du code représenté en figure 5.2, ce qui conduit à :

$$\begin{aligned} s(D) &= \frac{d(D)}{G^{(2)}(D)} \\ r(D) &= \frac{G^{(1)}(D)}{G^{(2)}(D)} d(D) \end{aligned} \tag{5.8}$$

Ainsi, un code systématique récursif se dérive aisément d'un code non-systématique non-récursif. Les codes générés par de tels codeurs peuvent être représentés graphiquement selon trois modèles : l'arbre, le treillis et la machine à états.

5.2.3 Arbre d'un code

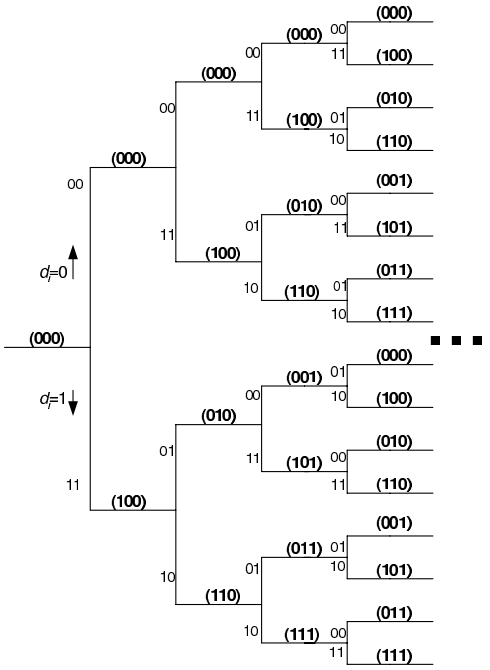


Figure 5.6 – Diagramme en arbre du code de polynômes $[1, 1 + D + D^3]$. Les couples binaires indiquent les sorties du codeur et les valeurs entre parenthèses sont les états futurs.

La première représentation graphique d'un code et certainement la moins pertinente pour la suite du chapitre est la représentation en arbre. Elle permet de présenter toutes les séquences d'états possibles. La racine est associée à l'état de départ du codeur. On en dérive tous les états successifs possibles en fonction de l'entrée d_i du codeur. La branche reliant un état père à un état fils

est étiquetée par la valeur des sorties du codeur lors de la transition associée. Ce principe est itéré pour chacun des états fils et ainsi de suite. Le diagramme en arbre associé au codeur systématique de la figure 5.1 est illustré en figure 5.6. Ce type de diagramme ne sera pas utilisé dans la suite, la seule utilisation qui en est faite concernant un algorithme de décodage séquentiel(l'algorithme de Fano) non traité dans cet ouvrage.

5.2.4 Treillis d'un code

La représentation la plus courante d'un code convolutif est le diagramme en treillis. Il est d'une importance majeure aussi bien pour la définition des propriétés d'un code que pour son décodage, comme nous le verrons dans la suite du chapitre 5.

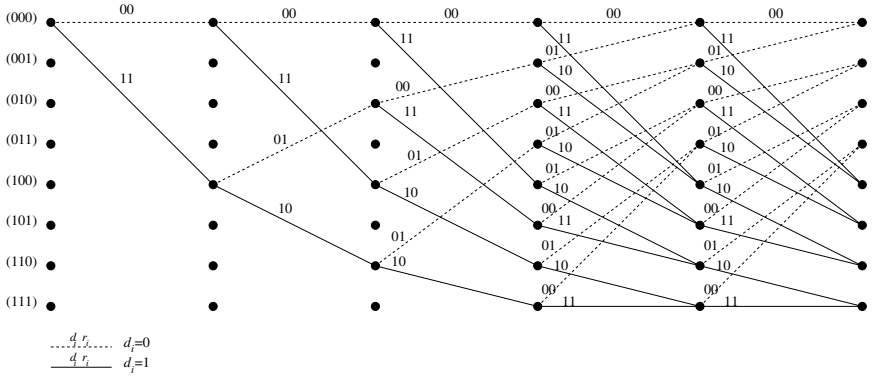


Figure 5.7 – Diagramme en treillis d'un code de polynômes générateurs $[1, 1+D+D^3]$.

À un instant i, l'état d'un codeur convolutif peut prendre 2^ν valeurs. Chacune de ces valeurs possibles est représentée par un nœud. À chaque instant i est associée une colonne d'états-nœuds et en fonction de l'entrée d_i, le codeur transite d'un état $\mathbf{s}_i$ à $\mathbf{s}_{i+1}$ en délivrant les bits codés. Cette transition entre deux états est représentée par un arc entre les deux nœuds associés et étiqueté avec les sorties du codeur. Dans le cas d'un code binaire, la transition sur une entrée à 0 (resp. 1) est représentée par un trait pointillé (resp. plein). La succession des états $\mathbf{s}_i$ jusqu'à l'instant t est représentée par les différents chemins entre l'état de départ et les différents états possibles à l'instant t.
Illustrons ceci par l'exemple du codeur systématique de la figure 5.1. En faisant l'hypothèse que l'état de départ $\mathbf{s}_0$ est l'état (000) :

- si $d_1 = 0$ alors l'état suivant, $\mathbf{s}_1$, est aussi (000). La transition sur une entrée à 0 est représentée par un trait pointillé et étiquetée dans ce premier cas par 00, la valeur des sorties du codeur ;
- si $d_1 = 1$, alors l'état suivant, $\mathbf{s}_1$, est (100). La transition sur une entrée à 1 est représentée par un trait plein et étiquetée ici par 11.

– Il faut ensuite envisager les quatre transitions possibles : de $\mathbf{s}_1 = (000)$ si $d_2 = 0$ ou $d_2 = 1$ et de $\mathbf{s}_1 = (100)$ si $d_2 = 0$ ou $d_2 = 1$.

En itérant cette construction, on aboutit à la représentation, en figure 5.7, de toutes les successions possibles d'états à partir de l'état de départ jusqu'à l'instant 5, sans la croissance illimitée du diagramme en arbre.

Une section complète du treillis suffit à caractériser le code. La section de treillis du code précédent est ainsi représentée en figure 5.8(a). De même, les codeurs présentés en figures 5.2 et 5.3 sont associés à des sections de treillis, illustrées respectivement en figures 5.8(b) et 5.8(c).

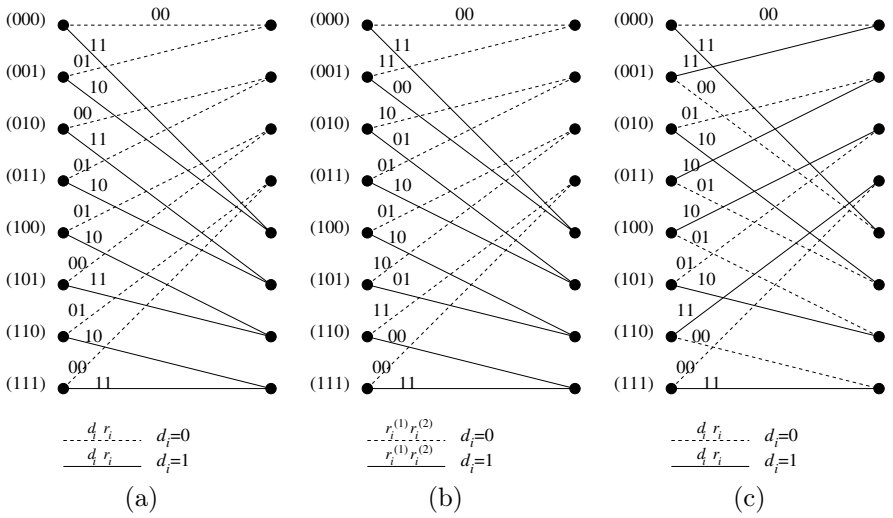


Figure 5.8 – Sections en treillis des codes des polynômes générateurs (a) $[1, 1+D+D^3]$, (b) $[1+D^2+D^3, 1+D+D^3]$, (c) $[1, (1+D^2+D^3)/(1+D+D^3)]$.

Une telle représentation met en évidence le motif élémentaire de ces treillis : le papillon, dont l'appellation est directement associée à son dessin. Chacune des sections des figures 5.8 est ainsi composée de 4 papillons (les transitions des états 0 et 1 vers les états 0 et 4 en forment un). La structure en papillons des trois treillis illustrés est identique mais les séquences codées diffèrent. Il faut noter en particulier que toutes les transitions aboutissant à un même nœud d'un treillis d'un code non-récursif sont dues à une même valeur en entrée du codeur. Ainsi, parmi les deux exemples non-récursifs traités (figures 5.8(a) et 5.8(b)), une transition associée à un 0 en entrée aboutit nécessairement dans un des états compris entre 0 à 3 et une transition à 1 aboutit dans un des états compris entre 4 et 7. Il en va différemment dans le cas d'un code récursif (comme celui présenté en figure 5.8(c)) : chaque état admet une transition incidente associée à une entrée à 0 et une autre associée à 1. Nous en verrons les conséquences en section 5.3.

5.2.5 Machine à états d'un code

Pour représenter les différentes transitions entre les états d'un codeur, il existe une dernière représentation qui est celle d'une machine à états. La convention pour définir les arcs de transition est identique à celle utilisée dans la section précédente. Seuls 2^ν nœuds sont représentés, indépendamment de l'instant i, ce qui revient à la représentation précédente sous forme de section de treillis. Les codeurs des figures 5.1, 5.2 et 5.3 admettent ainsi une représentation sous forme de machine à états illustrée respectivement en figures 5.9, 5.10 et 5.11.

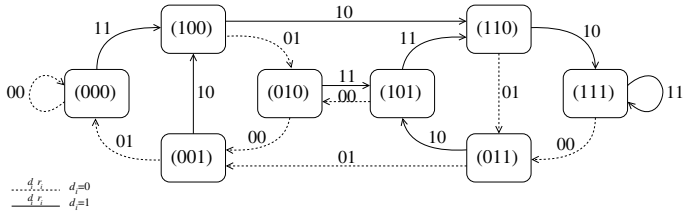


Figure 5.9 – Machine à état pour un code de polynômes générateurs $[1, 1 + D + D^3]$.

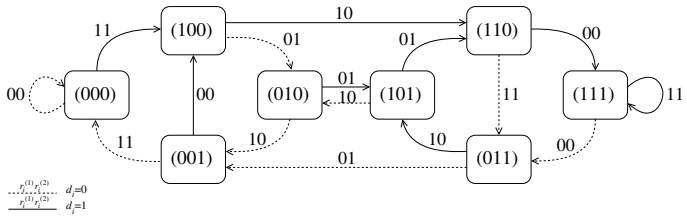


Figure 5.10 – Machine à état pour un code de polynômes générateurs $[1+D^2+D^3, 1+D+D^3]$.

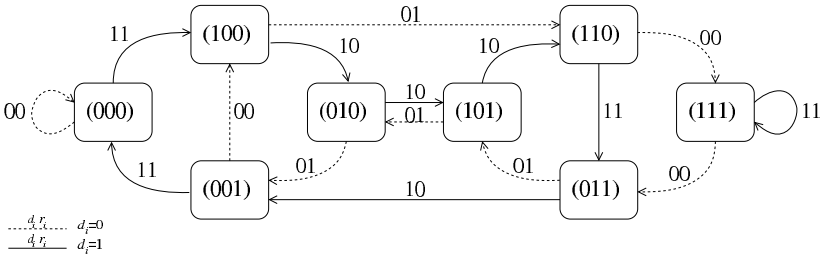


Figure 5.11 – Machine à état pour un code de polynômes générateurs $[1, (1 + D^2 + D^3)/(1 + D + D^3)]$.

Cette représentation est particulièrement utile pour déterminer la fonction de transfert et le spectre de distances d'un code convolutif (section 5.3). Il est

d'ores et déjà possible de remarquer une différence notable entre une machine à état d'un code récursif et celle d'un code non-récursif : l'existence et le nombre de cycles[1] sur une séquence nulle en entrée.
Dans le cas des deux machines à état non-récursives, il existe un seul cycle sur séquence nulle en entrée : la boucle sur l'état 0.
La machine à état récursive admet par contre un autre cycle sur séquence nulle en entrée : état 4 → état 6 → état 7 → état 3 → état 5 → état 2 → état 1 → état 4.
De plus, ce cycle est relié à la boucle sur l'état 0 par deux transitions associées à des entrées à 1 (transitions 0 → 4 et 1 → 0). Il existe alors une infinité de séquences d'entrée de poids de Hamming[2] égal à 2 produisant un cycle sur l'état 0. Ce poids 2 est le poids minimal de toute séquence non nulle qui fait quitter l'état 0 du codeur récursif et l'y conduit à nouveau. Du fait de la linéarité du code (cf. chapitre 1), cette valeur 2 est aussi la plus petite distance qui peut séparer deux séquences d'entrée différentes qui font quitter le codeur d'un même état pour l'y conduire également dans un même état.

Dans le cas des codes non-récursifs, le poids de Hamming des séquences d'entrée permettant un cycle sur l'état 0 peut être seulement de 1 (état 0 → état 4 → état 2 → état 1 → état 0). Cette distinction est essentielle pour comprendre l'intérêt des codes récursifs utilisés seuls (section 5.3) ou dans une structure de turbocode (chapitre 7).

5.3 Distances et performances des codes

5.3.1 Du choix d'un bon code

Un code étant exploité pour ses capacités de correction, il faut être en mesure de pouvoir les estimer pour choisir un code parmi d'autres de manière judicieuse, en fonction de l'application visée. Parmi les mauvais choix possibles, les *codes catastrophiques* sont tels qu'un nombre fini d'erreurs à l'entrée du décodeur peut produire un nombre infini d'erreurs en sortie du décodeur, ce qui justifie bien leur appellation. Une propriété majeure de ces codes est qu'il existe au moins une séquence d'entrée de poids infini qui engendre une séquence codée de poids fini : les codes systématiques ne peuvent donc pas être catastrophiques. Ces codes peuvent être identifiés très simplement s'ils ont un rendement de la forme $R = 1/N$. On peut alors montrer que le code est catastrophique si le plus grand commun diviseur (P.G.C.D.) de ses polynômes générateurs est différent de l'unité. Ainsi, le code de polynômes générateurs $G^{(1)}(D) = 1 + D + D^2 + D^3$ et $G^{(2)}(D) = 1 + D^3$ est catastrophique car le P.G.C.D. est $1 + D$.

Cependant le choix d'un code convolutif ne peut se résumer à la question « est-il catastrophique ? ». En exploitant les représentations graphiques introduites précédemment, les propriétés et les performances des codes peuvent être

[1] Un cycle est une succession d'états telles que l'état initial est aussi l'état final.
[2] Le poids de Hamming d'une séquence binaire est égal au nombre de bits différents de 0.

comparées.

5.3.2 Séquences *RTZ*

Puisque les codes convolutifs sont linéaires, déterminer les distances entre les différentes séquences codées revient à déterminer les distances entre les séquences codées non-nulles et la séquence « tout 0 ». Il suffit donc de calculer le poids de Hamming de toutes les séquences codées qui partent de l'état 0 et qui y reviennent. Ces séquences sont appelées séquences *RTZ* (pour *Return To Zero*). Le poids de Hamming le plus faible ainsi obtenu est appelé *distance libre* du code. La distance minimale de Hamming d'un code convolutif est égale à sa distance libre à partir d'une certaine longueur de séquence codée. Par ailleurs, le nombre de séquences *RTZ* qui ont le même poids est appelé multiplicité de ce poids.

Considérons les codes qui ont servi d'exemples jusqu'à présent. Chaque séquence *RTZ* de poids minimal est représentée en gras sur les figures 5.12, 5.13 et 5.14.

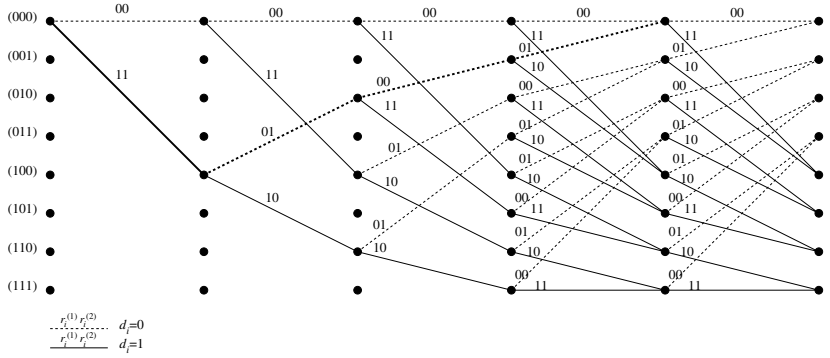


Figure 5.12 – Séquence *RTZ* (en gras) définissant le distance libre du code de polynômes générateurs $[1, 1 + D + D^3]$.

Le code systématique non-récursif possède une séquence *RTZ* de poids minimal de Hamming égal à 4. La distance libre de ce code est donc égale à 4. Par contre, le code classique et le code systématique récursif possèdent chacun deux séquences *RTZ* de poids minimal 6, leur distance libre est donc de 6. La capacité de correction des codes non-systématique non-récursif et systématique récursif est donc meilleure que celle du code systématique non-récursif.

Il est intéressant, en outre, de comparer les poids des séquences en entrée associées aux séquences *RTZ* de poids minimal. Dans le cas du code systématique non-récursif, la seule séquence de ce type est de poids égal à 1, ce qui signifie que si la séquence *RTZ* est décidée à la place de la séquence émise « tout 0 », seul un bit est en erreur. Dans le cas du code classique, une séquence en entrée a un poids de 1 et une autre un poids de 3 : un ou trois bits sont donc en

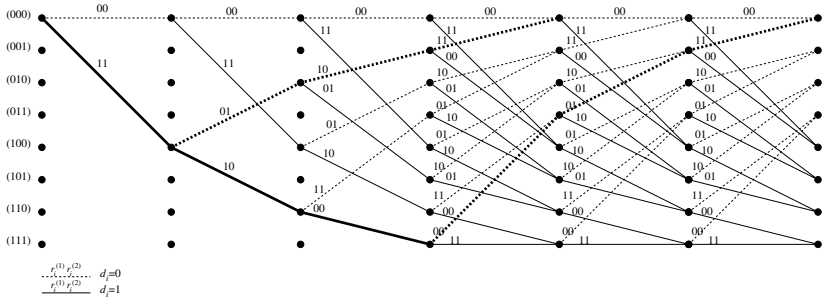


Figure 5.13 – Séquences *RTZ* (en gras) définissant la distance libre du code de polynômes générateurs $[1 + D^2 + D^3, 1 + D + D^3]$.

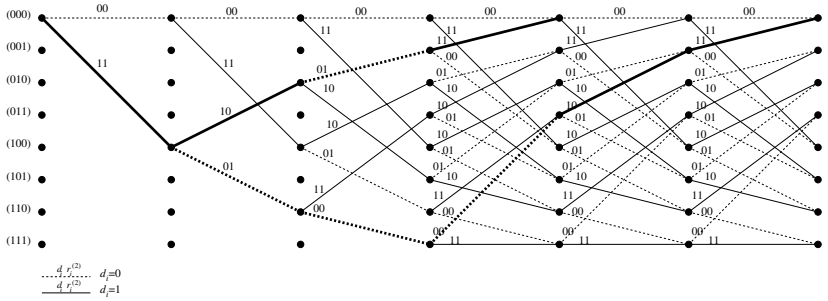


Figure 5.14 – Séquences *RTZ* (en gras) définissant la distance libre du code de polynômes générateurs $[1, (1 + D^2 + D^3)/(1 + D + D^3)]$.

erreur si une telle séquence *RTZ* est décodée. Dans le cas du code systématique récursif, les séquences *RTZ* de poids minimal ont un poids d'entrée de 3.

La connaissance de la distance minimale de Hamming et du poids d'entrée qui lui est associé n'est pas suffisante pour évaluer finement la probabilité d'erreur en sortie du décodeur d'un code convolutif simple. L'énumération des distances, au delà de la distance minimale de Hamming, et de leurs poids est nécessaire à cette évaluation. Cette énumération porte le nom de *spectre des distances*.

5.3.3 Fonction de transfert et spectre de distances

Le pouvoir de correction d'un code dépend de toutes les séquences *RTZ*, que l'on va considérer suivant l'ordre croissant de leurs poids. Plutôt que de les énumérer à partir d'une lecture des graphes, il est possible d'établir la fonction de transfert du code. Celle-ci s'obtient à partir du diagramme de transitions d'états dans lequel l'état de départ (000) est scindé en deux états a_e et a_s, qui ne sont autres que l'état de départ et l'état d'arrivée de toute séquence *RTZ*.

Illustrons le calcul de la fonction de transfert par le cas du code systématique de la figure 5.1, dont le diagramme de transitions d'états est de nouveau représenté en figure 5.15.

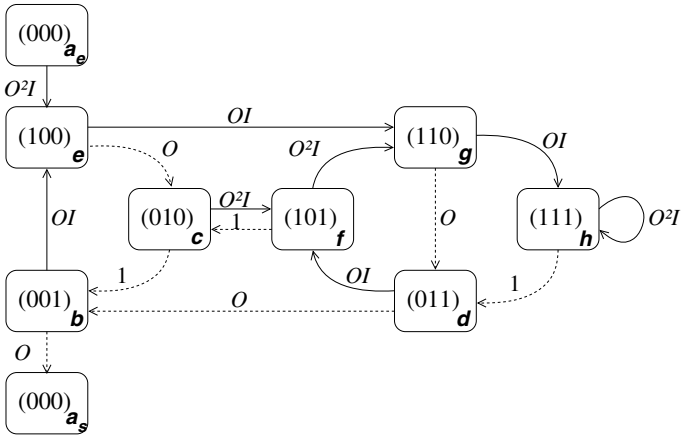


Figure 5.15 – Machine à états du code $[1, 1 + D + D^3]$, modifiée pour le calcul de la fonction de transfert associée.

Chaque transition possède une étiquette $O^i I^j$, où i est le poids de la séquence codée et j celui de la séquence en entrée du codeur. Dans notre exemple, j peut prendre la valeur 0 ou 1 selon le niveau du bit en entrée du codeur à chaque transition et i varie entre 0 et 2, puisque 4 symboles codés sont possibles (00, 01, 10, 11), de poids compris entre 0 et 2.

La fonction de transfert du code $T(O, I)$ est alors définie par :

$$T(O, I) = \frac{a_s}{a_e} \tag{5.9}$$

Pour établir cette fonction, il faut résoudre le système d'équations issu des relations entre les 9 états (ae, b, c... h et as) :

$$\begin{aligned} b &= c + Od \\ c &= Oe + f \\ d &= h + Dg \\ e &= O^2 I a_e + OIb \\ f &= O^2 Ic + OId \\ g &= OIe + O^2 If \\ h &= O^2 Ih + OIg \\ a_s &= Ob \end{aligned} \tag{5.10}$$

À l'aide d'un outil de calcul formel, il est aisé d'aboutir au résultat suivant :

$$T(O, I) = \frac{-I^4O^{12} + (3I^4 + I^3)O^{10} + (-3I^4 - 3I^3)O^8 + (I^4 + 2I^3)O^6 + IO^4}{I^4O^{10} + (-3I^4 - I^3)O^8 + (3I^4 + 4I^3)O^6 + (-I^4 - 3I^3)O^4 - 3IO^2 + 1} \tag{5.11}$$

$T(O,I)$ peut ensuite être développée en série :

$$\begin{aligned} T(O,I) = \quad & IO^4 \\ + \quad & (I^4 + 2I^3 + 3I^2)O^6 \\ + \quad & (4I^5 + 6I^4 + 6I^3)O^8 \\ + \quad & (I^8 + 5I^7 + 21I^6 + 24I^5 + 17I^4 + I^3)O^{10} \\ + \quad & (7I^9 + 30I^8 + 77I^7 + 73I^6 + 42I^5 + 3I^4)O^{12} \\ + \quad & \cdots \end{aligned} \tag{5.12}$$

Cette écriture permet d'observer qu'une séquence RTZ de poids 4 est produite par une séquence en entrée de poids 1, que des séquences RTZ de poids 6 sont produites par une séquence de poids 4, par deux séquences de poids 3 et par trois séquences de poids 2, etc.

Dans le cas du code classique précédemment cité, la fonction de transfert est :

$$\begin{aligned} T(O,I) = \quad & (I^3 + I)O^6 \\ + \quad & (2I^6 + 5I^4 + 3I^2)O^8 \\ + \quad & (4I^9 + 16I^7 + 21I^5 + 8I^3)O^{10} \\ + \quad & (8I^{12} + 44I^{10} + 90I^8 + 77I^6 + 22I^4)O^{12} \\ + \quad & (16I^{15} + 112I^{13} + 312I^{11} + 420I^9 + 265I^7 + 60I^5)O^{14} \\ + \quad & \cdots \end{aligned} \tag{5.13}$$

De même, le code systématique récursif déjà étudié admet pour fonction de transfert :

$$\begin{aligned} T(O,I) = \quad & 2I^3O^6 \\ + \quad & (I^6 + 8I^4 + I^2)O^8 \\ + \quad & 8I^7 + 33I^5 + 8I^3)O^{10} \\ + \quad & I^{10} + 47I^8 + 145I^6 + 47I^4 + I^2)O^{12} \\ + \quad & 14I^{11} + 254I^9 + 649I^7 + 254I^5 + 14I^3)O^{14} \\ + \quad & \cdots \end{aligned} \tag{5.14}$$

La comparaison des fonctions de transfert du point de vue du monôme de plus faible degré permet d'apprécier le pouvoir de correction à très fort rapport signal à bruit (comportement asymptotique). Ainsi, le code systématique non-récursif est plus faible que ses rivaux puisque de distance minimale inférieure. Un code classique et son équivalent systématique récursif ont la même distance libre, mais leurs monômes de degré minimal diffèrent. Le premier est en $(I^3 + I)O^6$ et le second en $2I^3O^6$. Cela signifie que par le code classique une séquence en entrée de poids 3 et une autre de poids 1 produisent une séquence RTZ de poids 6 tandis que par le code systématique récursif deux séquences de poids 3 produisent une séquence RTZ de poids 6. De la sorte, si une séquence RTZ de poids minimal est introduite par le bruit, le code classique introduira une ou trois erreurs, tandis son comparse systématique récursif introduira trois ou trois autres erreurs. En conclusion, la probabilité d'erreur binaire sur une

telle séquence est plus faible avec un code classique qu'avec un code systématique récursif, ce qui explique que le premier sera légèrement meilleur à fort rapport signal à bruit. Il en va généralement autrement lorsque les codes sont poinçonnés (cf. section 5.5) pour disposer de rendements plus élevés [5.9].

Pour comparer les performances des codes à faible rapport signal à bruit, il faut considérer l'ensemble des monômes. Prenons l'exemple du monôme en O^{12} respectivement pour le code systématique non-récursif, le classique et le systématique récursif :

$$(7I^9 + 30I^8 + 77I^7 + 73I^6 + 42I^5 + 3I^4)O^{12}$$
$$(8I^{12} + 44I^{10} + 90I^8 + 77I^6 + 22I^4)O^{12}$$
$$(I^{10} + 47I^8 + 145I^6 + 47I^4 + I^2)O^{12}$$

Si 12 erreurs sont introduites par le bruit sur le canal, 232 séquences *RTZ* sont « disponibles » en erreur pour le premier code, 241 pour le deuxième et encore 241 pour le troisième. Il est donc (un peu) moins probable qu'une séquence *RTZ* apparaisse si le code employé est le code systématique non-récursif. De plus, l'espérance d'erreur par séquence *RTZ* des trois codes est respectivement $6, 47$, $7, 49$ et 6 : le code systématique récursif introduit donc en moyenne moins d'erreurs de décodage que le code classique sur des séquences de *RTZ* de 12 erreurs sur la trame codée. Ceci se vérifie aussi pour les monômes de plus haut degré. Les codes systématiques récursif et non-récursif sont donc plus performants à faible rapport signal à bruit que le code classique. En outre, nous retrouvons dans la fonction de transfert du code récursif les monômes I^2O^{8+4c}, où c est un entier. Cette infinité de monômes de ce type est due à l'existence du cycle sur une séquence en entrée nulle différent de la boucle sur l'état 0. Un tel code ne fournit d'ailleurs pas de monômes de la forme IO^c, contrairement aux codes non-récursifs. Ces conclusions rejoignent celles issues de l'étude des machines à états en section 5.2.

Cette notion de fonction de transfert est donc efficace pour étudier les performances d'un code convolutif. Une version dérivée est par ailleurs essentielle à la classification des codes vis-à-vis de leurs performances. Il s'agit du spectre de distance $\omega(d)$ dont la définition est la suivante :

$$\left(\frac{\partial T(O,I)}{\partial I}\right)_{I=1} = \sum_{d=d_f}^{\infty} \omega(d) O^d \tag{5.15}$$

Par exemple, les premiers termes du spectre du code systématique récursif, obtenus à partir de (5.14), sont présentés dans le tableau 5.1.

Ce spectre est essentiel pour l'estimation des performances des codes en termes de calcul de probabilité d'erreur, comme l'illustre l'abondante littérature à ce sujet [5.10].

Les codes utilisés dans les exemples précédents ont un rendement de $1/2$. En augmentant le nombre de bits de redondance n le rendement devient plus faible. Dans ce cas, les puissances de O associées aux branches des machines à

d	6	8	10	12	14	...
$\omega(d)$	6	40	245	1446	8295	...

Table 5.1 – Premiers termes du spectre du code systématique récursif de polynômes générateurs [1, (1+D2+D3)/(1+D+D3)].

états seront supérieures ou égales à celles des figures précédentes. Ceci conduit à des fonctions de transfert avec des puissances de O plus élevées c'est-à-dire à des séquences RTZ de poids de Hamming plus importants. Les codes de rendements faibles ont donc un pouvoir de correction supérieur.

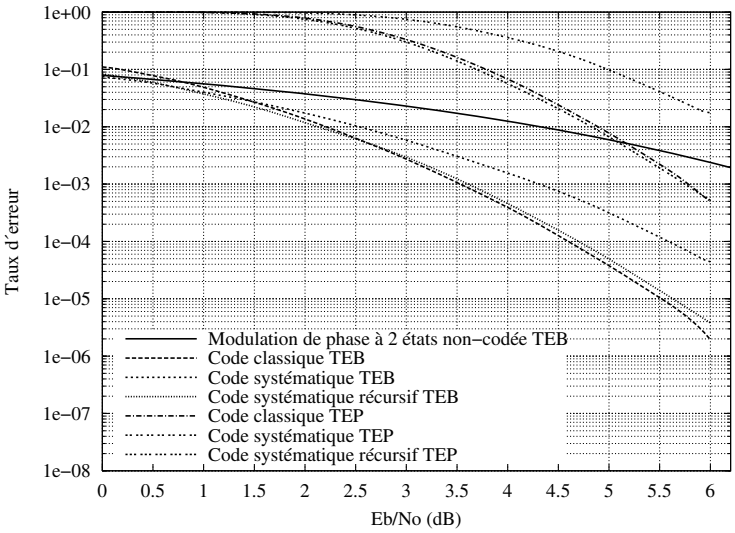


Figure 5.16 – Comparaison des performances (Taux d'Erreurs Binaire et Taux d'Erreurs Paquet) simulées des trois catégories de codes convolutifs après transmission de paquets de 53 octets sur un canal gaussien (décodage selon l'algorithme MAP).

5.3.4 Performances

La performance d'un code est définie par la probabilité d'erreur au décodage après transmission sur un canal bruité. La section précédente nous a permis de comparer de manière intuitive les codes non-systématique non-récursif, systématique non-récursif et systématique récursif de même longueur de contrainte. Toutefois, pour estimer les performances absolues d'un code, il faut pouvoir estimer la probabilité d'erreur de décodage en fonction du bruit, ou du moins la borner. La littérature, par exemple [5.10], définit ainsi de nombreuses bornes qui ne seront pas détaillées ici et nous nous limiterons à comparer les trois catégories de codes convolutifs. Pour ce faire, une transmission sur canal gaus-

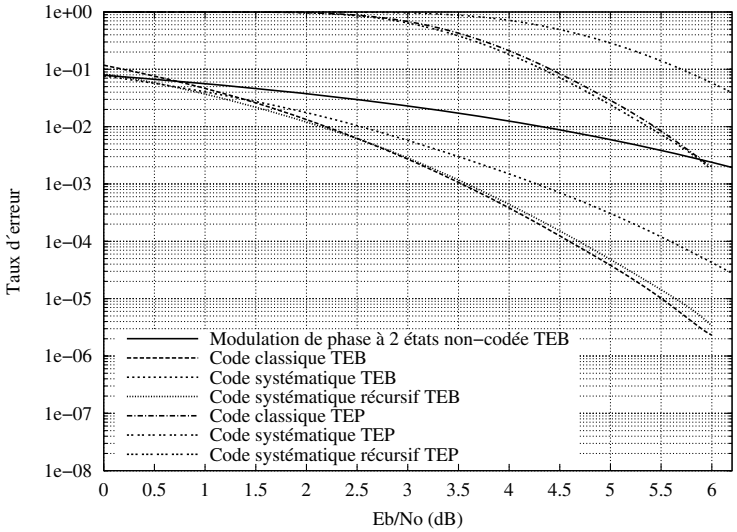


Figure 5.17 – Comparaison des performances simulées des trois catégories de codes convolutifs après transmission de blocs de 200 octets sur un canal gaussien.

sien de blocs de 53 puis 200 octets codés selon différents schémas a été simulée (figures 5.16 et 5.17) : classique (non-systématique non-récursif), systématique non-récursif et systématique récursif.

Les blocs ont été construits suivant la technique de fermeture de treillis classique pour les codes non-récursifs alors que le code récursif est de terminaison circulaire (voir section 5.5). L'algorithme de décodage utilisé est le *MAP*.

Les courbes de TEB sont en parfaite concordance avec les conclusions tirées lors de l'analyse de la distance libre des codes et de leur fonction de transfert : le code systématique est nettement moins bon que les autres à fort rapport signal à bruit et le code classique est alors légèrement meilleur que le récursif. À faible rapport signal à bruit, la hiérarchie est différente : le code récursif et le code systématique sont équivalents et meilleurs que le code classique.

La comparaison des performances en fonction de la taille de la trame (53 et 200 octets) montre que la hiérarchie de performance des codes n'est pas modifiée. De plus les taux d'erreurs bit sont quasiment identiques. Ceci était prévisible car les tailles des trames sont suffisamment importantes pour que les fonctions de transferts des codes ne soient pas affectées par des effets de bord. Par contre, le taux d'erreurs paquet est affecté par la longueur des blocs puisque si la probabilité d'erreur bit est constante, la probabilité d'erreur paquet croit avec sa taille.

Les comparaisons précédentes ne concernent que des codes à 8 états. Il est cependant facile de remarquer que les performances d'un code convolutif sont liées à sa capacité à fournir une information sur la succession des données

transmises : plus il peut intégrer de données successives dans ses symboles de sortie, plus il améliore la qualité de protection des-dites données. Autrement dit, un code convolutif est d'autant plus performant (au sein de sa catégorie) que son nombre d'états (donc la taille du registre du codeur) est grand. Comparons trois codes systématiques récursifs :

- à 4 états $[1, (1 + D^{2)}/(1 + D + D^2)]$,
- à 8 états $[1, (1 + D^2 + D^{3)}/(1 + D + D^3)]$
- et à 16 états $[1, (1 + D + D^2 + D^{4)}/(1 + D^3 + D^4)]$.

Leurs performance en termes de TEB et TEP ont été simulées sur un canal gaussien et sont présentées en figure 5.18.

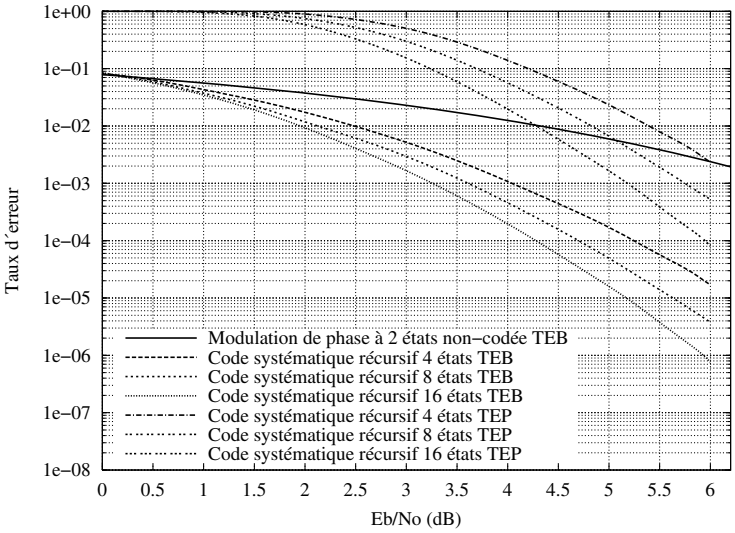


Figure 5.18 – Comparaison des performances simulées de codes convolutifs systématiques récursifs à 4, 8 et 16 états après transmission de paquets de 53 octets sur un canal gaussien (décodage selon l'algorithme *MAP*

Les taux d'erreurs résiduelles sont d'autant plus faibles que le nombre d'états du code est important. Pour un TEB de 10^{-4}, 0,6dB sont ainsi gagnés lors du passage du 4 états au 8 états et 0.5dB lors du passage du 8 états au 16 états. Cette remarque est cohérente avec la justification qualitative de l'intérêt d'un grand nombre d'états. Il semblerait alors logique de choisir un code convolutif avec un grand nombre d'états pour assurer la protection souhaitée. D'autant plus que de tels codes offrent la possibilité de produire une redondance sur bien plus que deux composantes, et donc de fournir une protection encore bien supérieure. Ainsi, le projet du *Big Viterbi Decoder* du *Jet Propulsion Laboratory* de la *NASA* utilisé pour des transmissions avec des sondes spatiales fut conçu pour traiter des trames encodées avec des codes convolutifs de 2 à 16384 états et des rendements bien inférieurs à 1/2 (16384 états et $R = 1/6$ pour les

sondes *Cassini* vers Saturne et *Mars Pathfinder*). Pourquoi ne pas utiliser de tels codes pour des transmissions radio-mobiles terrestres grand-public ? Parce que la complexité du décodage deviendrait rédhibitoire pour des transmissions terrestres actuelles avec un terminal fonctionnant en temps réel et tenant dans une poche de taille raisonnable...

5.4 Le décodage des codes convolutifs

Il existe plusieurs algorithmes de décodage des codes convolutifs. Le plus célèbre est probablement l'algorithme de Viterbi qui repose sur la représentation en treillis des codes ([5.4], [5.11]). Il permet de trouver, à partir de la séquence des symboles reçus, la séquence d'états dans le treillis la plus probable.

L'algorithme de Viterbi originel effectue un décodage à *sortie ferme*, c'est-à-dire qu'il fournit une estimation binaire de chacun des symboles transmis. Il n'est donc pas directement adapté aux systèmes itératifs qui requièrent une information de *confiance* sur les décisions. Des adaptations de l'algorithme de Viterbi telles que celles proposées dans [5.12], [5.13] ou [5.14] ont conduit aux versions à sortie pondérée dites *SOVA* (*Soft-Output Viterbi Algorithm*). Les algorithmes *SOVA* ne sont pas décrits dans cet ouvrage, car on leur préfère pour le turbo-décodage une autre famille d'algorithmes reposant sur la minimisation de la probabilité d'erreur de chaque symbole transmis. Ainsi, l'algorithme *Maximum A Posteriori* (*MAP*) permet le calcul de la valeur exacte de la probabilité *a posteriori* associée à chaque symbole transmis en utilisant la séquence reçue [5.5]. L'algorithme *MAP* et ses variantes sont détaillés dans le chapitre 7.

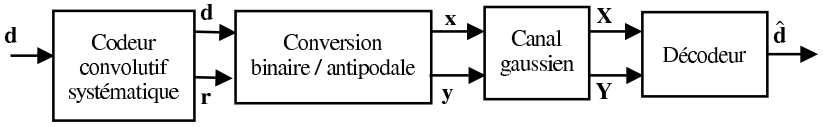


Figure 5.19 – Modèle de la chaîne de transmission étudiée.

5.4.1 Modèle de la chaîne de transmission et notations

Les algorithmes de Viterbi et *MAP* sont utilisés dans la chaîne de transmission décrite en figure 5.19. Le code convolutif considéré est systématique et calcule pour chaque séquence de données d'information $\mathbf{d} = \mathbf{d}_1^N = \{\mathbf{d}_1, \cdots, \mathbf{d}_N\}$ une séquence redondante $\mathbf{r} = \mathbf{r}_1^N = \{\mathbf{r}_1, \cdots, \mathbf{r}_N\}$. Le code est m-binaire de rendement $R = m/(m+n)$: chaque vecteur de données $\mathbf{d}_i$ en entrée du codeur est ainsi constitué de $m$bits, $\mathbf{d}_i = (d_i^{(1)}, d_i^{(2)}, \cdots, d_i^{(m)})$, et le vecteur de redondance correspondant en sortie s'écrit $\mathbf{r}_i = (r_i^{(1)}, r_i^{(2)}, \cdots, r_i^{(n)})$. La valeur de $\mathbf{d}_i$ pourra également être représentée par la grandeur scalaire entière $j = \sum_{l=1}^{m} 2^{l-1} d_i^{(l)}$, comprise entre 0 et $2^m - 1$, et l'on écrira alors $\mathbf{d}_i \equiv j$.

Les séquences de données systématiques $\mathbf{d}$ et redondantes $\mathbf{r}$ sont transmises après une conversion binaire/antipodale faisant correspondre à chaque valeur binaire (0 ou 1) issue du codeur une valeur antipodale (-1 ou +1) émise vers le canal. $\mathbf{X}$ et $\mathbf{Y}$ représentent les séquences de symboles systématiques et redondants bruités reçus à l'entrée du décodeur et $\hat{\mathbf{d}}$ la séquence décodée qui peut désigner soit une séquence binaire dans le cas de l'algorithme de Viterbi, soit une séquence de décisions pondérées associées aux $\mathbf{d}_i$ en sortie de l'algorithme *MAP*.

5.4.2 L'algorithme de Viterbi

L'algorithme de Viterbi est la méthode la plus couramment utilisée pour le décodage à maximum de vraisemblance (MV) des codes convolutifs de faible longueur de contrainte (typiquement $\nu \leqslant 8$). Au delà de cette limite, sa complexité de mise en œuvre impose d'avoir plutôt recours à un algorithme de décodage séquentiel tel que celui de Fano [5.3].

Le décodage à MV est basé sur la recherche du mot de code $\mathbf{c}$ qui est à la plus petite distance du mot reçu. Dans le cas d'un canal à décision binaire (canal binaire symétrique), le décodage à MV s'appuie sur la distance de Hamming, alors que dans celui d'un canal gaussien, il s'appuie sur la distance euclidienne (voir chapitre 2). Une recherche exhaustive des mots de codes associés aux différents chemins dans le treillis conduit à la prise en compte de $2^{\nu+k}$ chemins. En pratique, une recherche du chemin à distance minimale sur une fenêtre de travail de largeur l inférieure à k, limite la recherche à $2^{\nu+l}$chemins.

L'algorithme de Viterbi permet d'apporter une réduction notable de la complexité de calcul. Il est basé sur l'idée que, parmi l'ensemble des chemins du treillis qui convergent en un nœud à un instant donné, seul le chemin le plus probable peut être retenu pour les étapes suivantes de recherche. Notons $\mathbf{s}_i = (s_i^{(1)}, s_i^{(2)}, \ldots, s_i^{(\nu)})$ l'état du codeur à l'instant i et $T(i, \mathbf{s}_{i-1}, \mathbf{s}_i)$ la branche du treillis correspondant à l'émission de la donnée $\mathbf{d}_i$ et associée à la transition entre les nœuds $\mathbf{s}_{i-1}$ et $\mathbf{s}_i$. L'application de l'algorithme de Viterbi consiste à effectuer, la suite d'opérations décrite ci-après.

- *À chaque instant i, pour i allant de 1 à k :*
  - Calcul pour chaque branche d'une *métrique de branche*, $d\left(T(i, \mathbf{s}_{i-1}, \mathbf{s}_i)\right)$. Pour un canal à sortie binaire, cette métrique est définie comme la distance de Hamming entre le symbole porté par la branche du treillis et le symbole reçu, $d\left(T(i, \mathbf{s}_{i-1}, \mathbf{s}_i)\right) = d_H\left(T(i, \mathbf{s}_{i-1}, \mathbf{s}_i)\right)$.
    Pour un canal gaussien, la métrique est égale au carré de la distance euclidienne entre la branche considérée et l'observation à l'entrée du décodeur (voir aussi section 1.3) :

$$\begin{aligned} d\left(T(i, \mathbf{s}_{i-1}, \mathbf{s}_i)\right) &= \|\mathbf{X}_i - \mathbf{x}_i\|^2 + \|\mathbf{Y}_i - \mathbf{y}_i\|^2 \\ &= \sum_{j=1}^{m} \left(x_i^{(j)} - X_i^{(j)}\right)^2 + \sum_{j=1}^{n} \left(y_i^{(j)} - Y_i^{(j)}\right)^2 \end{aligned}$$

- Calcul de la *métrique cumulée* associée à chaque branche $T(i,\ \mathbf{s}_{i-1},\ \mathbf{s}_i)$ définie par :

$$\lambda(T(i,\mathbf{s}_{i-1},\mathbf{s}_i)) = \mu(i\ -1,\mathbf{s}_{i-1}) + d(T(i,\mathbf{s}_{i-1},\mathbf{s}_i))$$

  où $\mu(i-1,\mathbf{s}_{i-1})$ est la métrique cumulée associée au nœud $\mathbf{s}_{i-1}$.
- Pour chaque nœud $\mathbf{s}_i$, sélection de la branche du treillis correspondant à la métrique cumulée minimale et mémorisation de cette branche en mémoire (en pratique, c'est la valeur de d_i associée à la branche qui est stockée). Le chemin dans le treillis constitué des branches successivement mémorisées aux instants compris entre 0 et i est le *chemin survivant* arrivant en $\mathbf{s}_i$. Si les deux chemins qui convergent en $\mathbf{s}_i$ possèdent des métriques cumulées identiques, le survivant est alors choisi de façon arbitraire entre ces deux chemins.
- Calcul de la métrique cumulée associée à chaque nœud $\mathbf{s}_i$, $\mu(i,\mathbf{s}_i)$. Elle est égale à la métrique cumulée associée au chemin survivant arrivant en $\mathbf{s}_i$:

$$\mu(i,\mathbf{s}_i) = \min_{\mathbf{s}_{i-1}} \left(\lambda\left(T(i,\mathbf{s}_{i-1},\mathbf{s}_i)\right)\right).$$

• *Initialisation (instant i =0) :*

Les valeurs d'initialisation des métriques μ au démarrage de l'algorithme dépendent de l'état initial $\mathbf{s}$ du codeur : $\mu(0,\mathbf{s}) = +\infty$ si $\mathbf{s} \neq \mathbf{s}_0$ et $\mu(0,\mathbf{s}_0) = 0$. Si cet état n'est pas connu, toutes les métriques sont initialisées à la même valeur, typiquement 0. Dans ce cas, le décodage du début de la trame est moins efficace puisque les métriques cumulées associées à chaque branche à l'instant 1 ne dépendent que de la branche elle-même. Le passé ne peut pas être pris en compte puisqu'il n'est pas connu : $\lambda(T(1,\mathbf{s}_0,\mathbf{s}_1)) = d(T(1,\mathbf{s}_0,\mathbf{s}_1))$.

• *Calcul des décisions (instant i=k) :*

À l'instant k, si l'état final du codeur est connu, le chemin à vraisemblance maximale est le chemin survivant issu du nœud correspondant à l'état final du codeur. La séquence décodée est donnée par la suite des valeurs de d_i, i allant de 1 à k, stockées dans la mémoire associée au chemin à vraisemblance maximale. Cette opération est appelée *remontée du treillis*.

Si l'état final n'est pas connu, le chemin à vraisemblance maximale est le chemin survivant issu du nœud à métrique cumulée minimale. Dans ce cas, la problématique est analogue à ce qui a été évoqué pour l'initialisation : le décodage de la fin de trame est moins efficace.

Lorsque la séquence transmise est longue, voire de longueur infinie, il n'est pas possible d'attendre que toute la séquence binaire émise soit reçue pour commencer l'opération de décodage. Pour limiter la latence de décodage ainsi que la taille de mémoire nécessaire à la mémorisation des chemins survivants, il est nécessaire de tronquer le treillis. En observant le déroulement de l'algorithme, on peut remarquer qu'en remontant suffisamment dans le temps à partir de l'instant i, les chemins survivants issus des différents nœuds du treillis

convergent presque toujours vers un même chemin. En pratique, la mémorisation des survivants peut donc être limitée à un intervalle temporel de durée l. Il est alors suffisant de faire une remontée du treillis à chaque instant i sur une longueur l pour prendre la décision sur la donnée d_{i-l} . Pour diminuer la complexité, les survivants sont parfois mémorisés sur un intervalle supérieur à l (par exemple l+3). Le nombre d'opérations de remontée de treillis est alors diminué (divisé par 3 dans notre exemple) mais chacune des remontées fournit plusieurs décisions (d_{i-l}, d_{i-l-1} et d_{i-l-2}, pour notre exemple).

La valeur de l doit être d'autant plus grande que la mémoire du codeet le rendement de codage sont élevés. On observe que, pour un code systématique, des valeurs de l correspondant à la production par le codeur d'un nombre de symboles de redondance égal à 5 fois la longueur de contrainte du code sont suffisantes. A titre d'exemple, pour un code de rendement $R = 1/2$, on prend typiquement l égal à $5(\nu + 1)$.

Du point de vue de la complexité, l'algorithme de Viterbi nécessite le calcul de $2^{\nu+1}$ métriques cumulées à chaque instant i et sa complexité varie linéairement avec la longueur de la séquence k ou de la fenêtre de décodage l.

Exemple d'application de l'algorithme de Viterbi

Illustrons les différentes étapes de l'algorithme de Viterbi décrit précédemment par une application au décodage du code convolutif systématique récursif binaire (7,5) à 4 états, de rendement de codage $R = 1/2$ ($m = n = 1$). La structure du codeur et le treillis sont représentés dans la figure 5.20.

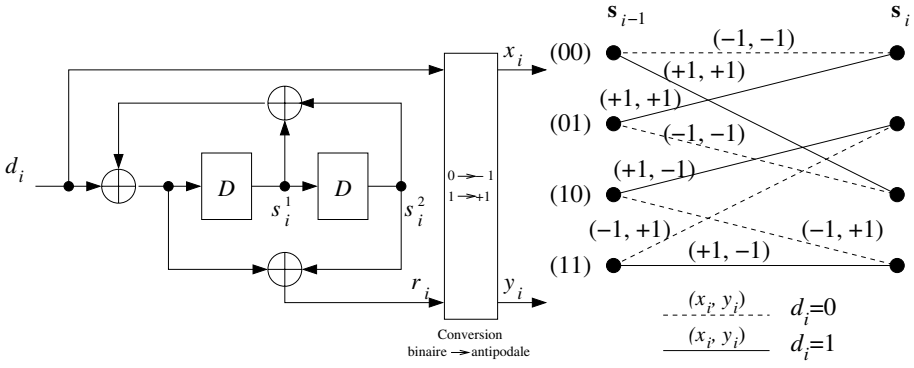


Figure 5.20 – Structure du code convolutif systématique récursif (7,5) et treillis associé.

Calcul des métriques de branches à l'instant i :

$$
\begin{aligned}
d(T(i,\mathbf{0},\mathbf{0})) = d(T(i,\mathbf{1},\mathbf{2})) &= (X_i+1)^2 + (Y_i+1)^2 \\
d(T(i,\mathbf{0},\mathbf{2})) = d(T(i,\mathbf{1},\mathbf{0})) &= (X_i-1)^2 + (Y_i-1)^2 \\
d(T(i,\mathbf{2},\mathbf{1})) = d(T(i,\mathbf{3},\mathbf{3})) &= (X_i-1)^2 + (Y_i+1)^2 \\
d(T(i,\mathbf{2},\mathbf{3})) = d(T(i,\mathbf{3},\mathbf{1})) &= (X_i+1)^2 + (Y_i-1)^2
\end{aligned}
$$

Calcul des métriques cumulées de branches à l'instant i :

$$\begin{aligned}
\lambda(T(i,\mathbf{0},\mathbf{0})) &= \mu(i-1,\mathbf{0}) + d(T(i,\mathbf{0},\mathbf{0})) = \mu(i-1,\mathbf{0}) + (X_i+1)^2 + (X_i+1)^2 \\
\lambda(T(i,\mathbf{1},\mathbf{2})) &= \mu(i-1,\mathbf{1}) + d(T(i,\mathbf{1},\mathbf{2})) = \mu(i-1,\mathbf{1}) + (X_i+1)^2 + (X_i+1)^2 \\
\lambda(T(i,\mathbf{0},\mathbf{2})) &= \mu(i-1,\mathbf{0}) + d(T(i,\mathbf{0},\mathbf{2})) = \mu(i-1,\mathbf{0}) + (X_i-1)^2 + (X_i-1)^2 \\
\lambda(T(i,\mathbf{1},\mathbf{0})) &= \mu(i-1,\mathbf{1}) + d(T(i,\mathbf{1},\mathbf{0})) = \mu(i-1,\mathbf{1}) + (X_i-1)^2 + (X_i-1)^2 \\
\lambda(T(i,\mathbf{2},\mathbf{1})) &= \mu(i-1,\mathbf{2}) + d(T(i,\mathbf{2},\mathbf{1})) = \mu(i-1,\mathbf{2}) + (X_i-1)^2 + (X_i+1)^2 \\
\lambda(T(i,\mathbf{3},\mathbf{3})) &= \mu(i-1,\mathbf{3}) + d(T(i,\mathbf{3},\mathbf{3})) = \mu(i-1,\mathbf{3}) + (X_i-1)^2 + (X_i+1)^2 \\
\lambda(T(i,\mathbf{2},\mathbf{3})) &= \mu(i-1,\mathbf{2}) + d(T(i,\mathbf{2},\mathbf{3})) = \mu(i-1,\mathbf{2}) + (X_i+1)^2 + (X_i-1)^2 \\
\lambda(T(i,\mathbf{3},\mathbf{1})) &= \mu(i-1,\mathbf{3}) + d(T(i,\mathbf{3},\mathbf{1})) = \mu(i-1,\mathbf{3}) + (X_i+1)^2 + (X_i-1)^2
\end{aligned}$$

Calcul des métriques cumulées de nœud à l'instant i :

$$\begin{aligned}
\mu(i,\mathbf{0}) &= \min\left(\lambda(T(i,\mathbf{0},\mathbf{0}), \lambda(T(i,\mathbf{1},\mathbf{0})\right) \\
\mu(i,\mathbf{1}) &= \min\left(\lambda(T(i,\mathbf{3},\mathbf{1}), \lambda(T(i,\mathbf{2},\mathbf{1})\right) \\
\mu(i,\mathbf{2}) &= \min\left(\lambda(T(i,\mathbf{0},\mathbf{2}), \lambda(T(i,\mathbf{1},\mathbf{2})\right) \\
\mu(i,\mathbf{3}) &= \min\left(\lambda(T(i,\mathbf{3},\mathbf{3}), \lambda(T(i,\mathbf{2},\mathbf{3})\right)
\end{aligned}$$

Pour chacun des quatre nœuds du treillis la valeur de d_i correspondant à la transition de métrique cumulée λ minimale est stockée en mémoire.

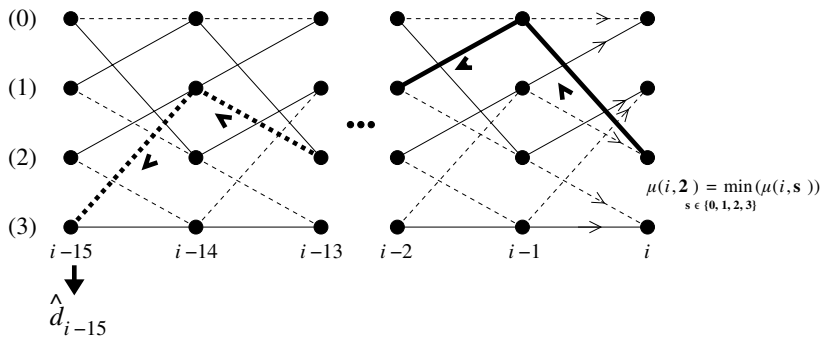


Figure 5.21 – Opération de remontée du chemin survivant (en gras) dans le treillis à partir de l'instant i et détermination de la décision binaire à l'instant $i-15$.

Après sélection du nœud de métrique cumulée minimale, noté $\mathbf{s}$ (dans l'exemple de la figure 5.21, $\mathbf{s} = \mathbf{3}$), on remonte le treillis le long du chemin survivant sur une profondeur $l = 15$. A l'instant $i-15$, la décision binaire $\hat{d}_{i-15}$ est égale à la valeur de d_{i-15} stockée dans la mémoire associée au chemin survivant.

L'application de l'algorithme de Viterbi à entrées pondérées vise à rechercher le mot de code $\mathbf{c}$ qui est à la plus petite distance euclidienne entre deux mots de codes. De manière équivalente (voir chapitre 1), il s'agit de rechercher le mot de code qui maximise :

$$\langle \mathbf{x}, \mathbf{X} \rangle + \langle \mathbf{y}, \mathbf{Y} \rangle = \sum_{i=1}^{k} \left(\sum_{l=1}^{m} x_i^{(l)} X_i^{(l)} + \sum_{l=1}^{n} y_i^{(l)} Y_i^{(l)} \right)$$

Dans ce cas, l'application de l'algorithme de Viterbi utilise des métriques de branches de la forme :

$$d\left(T(i, \mathbf{s}_{i-1}, \mathbf{s}_i)\right) = \sum_{l=1}^{m} x_i^{(l)} X_i^{(l)} \; + \sum_{l=1}^{n} y_i^{(l)} Y_i^{(l)}$$

et le chemin survivant correspond alors au chemin à métrique cumulée maximale.

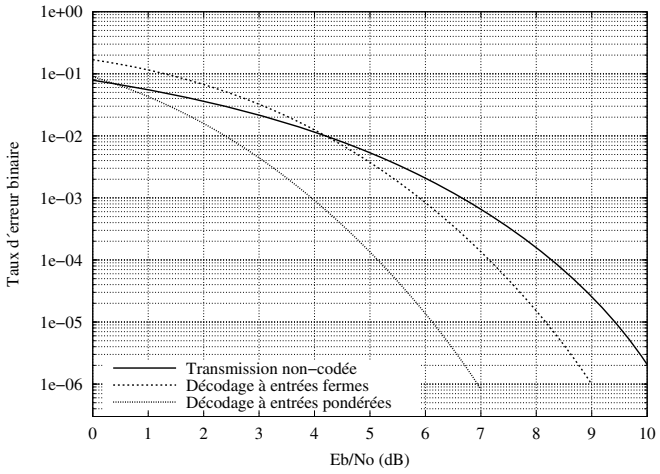


Figure 5.22 – Exemple de performances de correction sur canal gaussien de l'algorithme de Viterbi à entrées fermes et à entrées pondérées. Code convolutif systématique récursif (CSR) de polynômes générateurs 7 (récursivité) et 5 (redondance). Rendement de codage $R = 1/2$.

La figure 5.22 fournit les performances des deux variantes, à entrées fermes et à entrées pondérées, d'un décodeur utilisant l'algorithme de Viterbi pour le code CSR(7,5) pour une transmission sur un canal à bruit additif blanc gaussien. On observe en pratique un gain d'environ 2 dB lorsque l'on substitue un décodage à entrées pondérées à un décodage à entrées fermes.

5.4.3 L'algorithme *Maximum A Posteriori* ou *MAP*

L'algorithme de Viterbi détermine le mot de code le plus proche du mot reçu. Cependant, il ne minimise pas nécessairement la probabilité d'erreur des bits d'information ou symboles transmis. L'algorithme *MAP* permet de calculer la probabilité *a posteriori* de chaque bit d'information ou de chaque symbole transmis et le décodeur correspondant sélectionne à chaque instant le bit ou le symbole le plus probable. Cet algorithme a été publié en 1974 par Bahl, Cocke, Jelinek et Raviv [5.5]. La portée de cette méthode de décodage est restée confidentielle jusqu'à la découverte des turbocodes car elle n'apporte pas

d'amélioration de performance notable par rapport à l'algorithme de Viterbi pour le décodage des codes convolutifs et s'avère plus complexe à mettre en œuvre. En revanche la situation a changé en 1993 car le décodage des turbocodes fait appel à des décodeurs élémentaires à sorties pondérées ou souples et l'algorithme *MAP*, contrairement à l'algorithme de Viterbi, permet d'associer naturellement une pondération à chaque décision. L'algorithme *MAP* est présenté dans le chapitre 7.

5.5 Codes convolutifs en bloc

Les codes convolutifs sont naturellement adaptés aux applications de diffusion où le message transmis est de longueur infinie. Cependant, la plupart des systèmes de télécommunications utilisent des transmissions par trames indépendantes. Le paragraphe 5.4.2 a mis en évidence l'importance de la connaissance des états initial et final du codeur lors du décodage d'une trame. Les techniques utilisées pour connaître ces états sont généralement appelées *fermeture de treillis*. Elles consistent généralement à forcer les états initial et final à des valeurs connues du décodeur (en général zéro).

5.5.1 Fermeture de treillis

Fermeture classique

Le codeur étant construit autour d'un registre, il est aisé de l'initialiser à l'aide d'entrées de remise à zéro avant de commencer le codage d'une trame. Cette opération n'a aucune conséquence sur le rendement du code. La fermeture en fin de trame est moins simple.

Lorsque les k bits de la trame ont été codés, le registre du codeur est dans l'un quelconque des 2^{ν} états possibles. L'objectif de la fermeture est de conduire le codeur vers l'état zéro en suivant un des chemins du treillis pour que l'algorithme de décodage puisse utiliser cette connaissance de l'état final. Dans le cas de codes non-récursifs, l'état final est forcé à zéro en injectant ν bits à zéro en fin de trame. Tout se passe comme si la trame codée était de longueur $k+\nu$ avec $d_{k+1} = d_{k+2} = \ldots = d_{k+\nu} = 0$. Le rendement de codage est légèrement diminué par la transmission des *bits de fermeture*. Cependant, compte tenu de la taille des trames généralement transmises cette dégradation du rendement est bien souvent négligeable. Dans le cas des codes récursifs, il est aussi possible d'injecter un zéro à l'entrée du registre. La figure 5.23 illustre une méthode simple pour résoudre cette question.

Après initialisation du registre à zéro, l'interrupteur I est maintenu en position 1 et les données d_1 à d_k sont codées. A la fin de cette opération de codage, des instants k à $k+\nu$, l'interrupteur I est placé en position 2 et d_i prend la valeur issue de la rétroaction du registre, c'est à dire une valeur qui force une entrée de registre à zéro. En effet, $S_i^{(0)}$ est le résultat d'une somme modulo 2

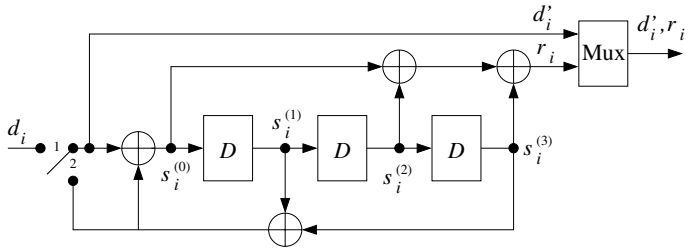


Figure 5.23 – Exemple de codeur de codes convolutifs systématiques récursifs permettant une terminaison à l'état 0 en ν périodes.

de deux membres identiques. Le codeur, quant à lui, continue de produire les redondances associées r_i.

Cette fermeture classique possède un inconvénient majeur : la protection des données n'est pas indépendante de leur position dans la trame. Cela peut notamment entraîner des effets de bord dans la construction d'un turbocode (voir chapitre 7).

Fermeture circulaire

Une technique fut introduite au tournant des années 70 et 80 [5.15] pour terminer les treillis des codes convolutifs sans les effets de bord : le *tail-biting*. Elle consiste à rendre le treillis de décodage circulaire, c'est-à-dire à faire en sorte que l'état de départ et l'état final du codeur soient identiques. Cet état est alors appelé état de circulation. Cette technique de fermeture appliquée aux codes récursifs génère un code dit convolutif systématique récursif circulaire (CSRC). Le treillis d'un tel code est illustré par la figure 5.24.

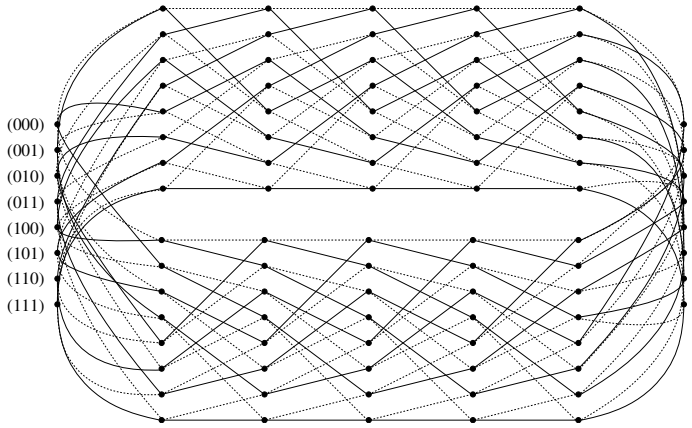


Figure 5.24 – Treillis d'un code CSRC à 8 états.

Pour expliquer la construction d'un tel code, il est nécessaire de revenir aux équations fondamentales qui régissent le fonctionnement d'un codeur. Il est possible d'établir une relation entre l'état $\mathbf{s}_{i+1}$ du codeur à un instant $i+1$, son état $\mathbf{s}_i$ et la donnée entrante $\mathbf{d}_i$ à un instant i :

$$\mathbf{s}_{i+1} = \mathbf{A}\mathbf{s}_i + \mathbf{B}\mathbf{d}_i \tag{5.16}$$

où $\mathbf{A}$ est la matrice d'état et $\mathbf{B}$ celle d'entrée. Dans le cas du code systématique récursif précédemment cité de polynômes générateurs $[1, (1+D^2+D^3)/(1+D+D^3)]$, ces matrices sont :

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \text{ et } \mathbf{B} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

Si le codeur est initialisé à l'état 0 ($\mathbf{s}_0 = 0$), l'état final $\mathbf{s}_k^0$ obtenu à la fin d'une trame de longueur k est :

$$\mathbf{s}_k^0 = \sum_{j=1}^{k} \mathbf{A}^{j-1}\mathbf{B}\mathbf{d}_{k-j} \tag{5.17}$$

Lorsqu'il est initialisé dans un état quelconque $\mathbf{s}_c$, l'état final $\mathbf{s}'_k$ s'exprime de la manière suivante :

$$\mathbf{s}'_k = \mathbf{A}^k\mathbf{s}_c + \sum_{j=1}^{k} \mathbf{A}^{j-1}\mathbf{B}\mathbf{d}_{k-j} \tag{5.18}$$

Pour que cet état $\mathbf{s}'_k$ soit égal à l'état de départ $\mathbf{s}_c$ et que celui-ci devienne donc l'état de circulation, il faut et il suffit que :

$$\left(\mathbf{I} - \mathbf{A}^k\right)\mathbf{s}_c = \sum_{j=1}^{k} \mathbf{A}^{j-1}\mathbf{B}\mathbf{d}_{k-j} \tag{5.19}$$

où $\mathbf{I}$ est la matrice identité de dimension $\nu \times \nu$.
Ainsi, en introduisant l'état $\mathbf{s}_k^0$ du codeur après initialisation à 0

$$\mathbf{s}_c = \left(\mathbf{I} - \mathbf{A}^k\right)^{-1}\mathbf{s}_k^0 \tag{5.20}$$

Ceci n'est possible qu'à condition que la matrice $(\mathbf{I} - \mathbf{A}^k)$ soit *inversible* : c'est la condition d'existence de l'état de circulation.

En conclusion, s'il existe, l'état de circulation s'obtient en deux étapes. La première consiste à coder la trame de k bits en entrée après avoir initialisé le codeur à l'état zéro et à conserver l'état de terminaison. La seconde se résume à déduire l'état de circulation du précédent état de terminaison et d'une table (obtenue par l'inversion de $\mathbf{I} - \mathbf{A}^k$).

Prenons pour exemple le code systématique récursif précédemment utilisé. Puisqu'il s'agit d'un code binaire, addition et soustraction sont équivalentes : l'état de circulation existe si $\mathbf{I} + \mathbf{A}^k$ est inversible. La matrice $\mathbf{A}$ est telle que $\mathbf{A}^7$ est égale à $\mathbf{I}$. Ainsi, si k est un multiple de $7(= 2^\nu - 1)$, $\mathbf{I} + \mathbf{A}^k$ est nulle et donc non-inversible : ce cas est à éviter. Une autre conséquence est que $\mathbf{A}^k = \mathbf{A}^{k \mod 7}$: il suffit de calculer une fois pour toutes les 6 tables de transformation d'état associées aux 6 valeurs possibles de $(\mathbf{I} - \mathbf{A}^k)^{-1}$, de les stocker et de lire la bonne table, après calcul de $k \mod 7$. La table du code CSRC à 8 états de polynômes générateurs $[1, (1 + D^2 + D^3)/(1 + D + D^3)]$ est donnée dans le tableau (5.2).

$\mathbf{s}_k^0$ \ $k \mod 7$	1	2	3	4	5	6
0	0	0	0	0	0	0
1	6	3	5	4	2	7
2	4	7	3	1	5	6
3	2	4	6	5	7	1
4	7	5	2	6	1	3
5	1	6	7	2	3	4
6	3	2	1	7	4	5
7	5	1	4	3	6	2

Table 5.2 – Table du code CSRC de polynômes générateurs $[1, (1 + D^2 + D^3)/(1 + D + D^3)]$ fournissant l'état de circulation en fonction de $k \mod 7$ (k étant la longueur de la trame en entrée) et de l'état terminal $\mathbf{s}_k^0$ obtenu après un encodage initialisé à l'état 0.

Une méthode simple pour le codage selon un treillis circulaire se résume en cinq étapes, après vérification de l'existence de l'état de circulation :

1. initialiser le codeur à l'état 0 ;
2. coder la trame pour obtenir l'état final $\mathbf{s}_k^0$;
3. calculer l'état $\mathbf{s}_c$ de circulation à partir des tables déjà calculées et stockées ;
4. initialiser le codeur à l'état $\mathbf{s}_c$;
5. coder la trame et transmettre les redondances calculées.

5.5.2 Poinçonnage

Certaines applications ne peuvent allouer que peu de place à la partie redondante des mots de code. Hors, par construction, le rendement naturel d'un code convolutif systématique est $m/(m + n)$, où m est le nombre de bits de l'entrée $\mathbf{d}_i$ du codeur et n est le nombre de bits de sortie. Il est donc maximum lorsque $n = 1$ et devient $R = m/(m + 1)$. Des rendements élevés ne peuvent donc être obtenus qu'avec des valeurs importantes de m. Malheureusement, le nombre de

transitions partant d'un nœud quelconque du treillis est 2^m. Autrement dit, la complexité du treillis, donc du décodage, croît de manière exponentielle avec le nombre de bits à l'entrée du codeur. Cette solution n'est donc, en général, pas satisfaisante. Elle est souvent écartée au profit d'une technique au pouvoir de correction légèrement plus faible mais plus facile à mettre en œuvre : le *poinçonnage*.

La technique du poinçonnage est couramment utilisée pour obtenir des rendements élevés. Elle consiste à utiliser un codeur de faible valeur de m (1 ou 2 par exemple), pour conserver une complexité raisonnable de décodage, mais à ne transmettre qu'une partie des bits codés. Un exemple est proposé en figure 5.25. Dans cet exemple, un codeur de rendement 1/2 produit des sorties d_i et r_i à chaque instant i. Seuls 3 bits sur 4 sont transmis, ce qui conduit à un rendement global de 2/3. Le motif suivant lequel les bits sont poinçonnés est appelé masque de poinçonnage.

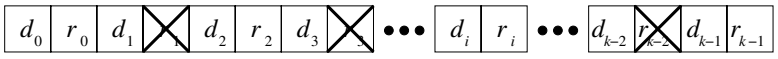


Figure 5.25 – Poinçonnage d'un code systématique pour obtenir un rendement 2/3.

Dans le cas des codes systématiques, c'est généralement la redondance qui est poinçonnée. La figure 5.26 décrit le treillis du code $[1, (1 + D^2 + D^3)/(1 + D + D^3)]$ résultant d'une opération de poinçonnage suivant le masque de la figure 5.25. Les « X » repèrent les bits qui ne sont pas transmis et qui ne pourront donc être utilisés pour le décodage.

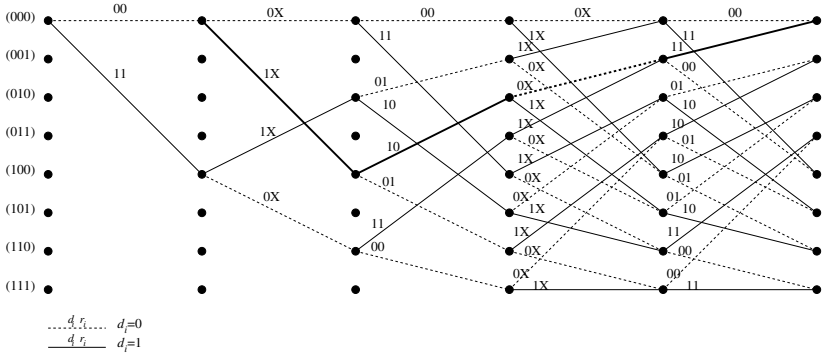


Figure 5.26 – Diagramme en treillis du code récursif poinçonné pour un rendement 2/3.

La technique de décodage la plus couramment utilisée consiste à prendre le décodeur du code originel en insérant des valeurs *neutres* à la place des éléments poinçonnés. Les valeurs neutres sont des valeurs représentatives d'informations

a *priori* non connues. Dans le cas usuel d'une transmission utilisant une signalisation antipodale (+1 pour le '1' logique, -1 pour le '0' logique), la valeur nulle (0 analogique) s'impose comme valeur neutre.

L'introduction d'un poinçonnage augmente le rendement du code mais diminue bien évidemment son pouvoir de correction. Ainsi, dans l'exemple de la figure 5.26, la distance libre du code est réduite de 6 à 4 (une séquence RTZ associée est mise en évidence sur la figure). De même la figure 5.27, où sont représentées les courbes de taux d'erreurs du code $[1, (1+D^2+D^3)/(1+D+D^3)]$ pour des rendements de 1/2, 2/3, 3/4 et 6/7, montre à l'évidence une diminution du pouvoir de correction avec l'augmentation du rendement du code.

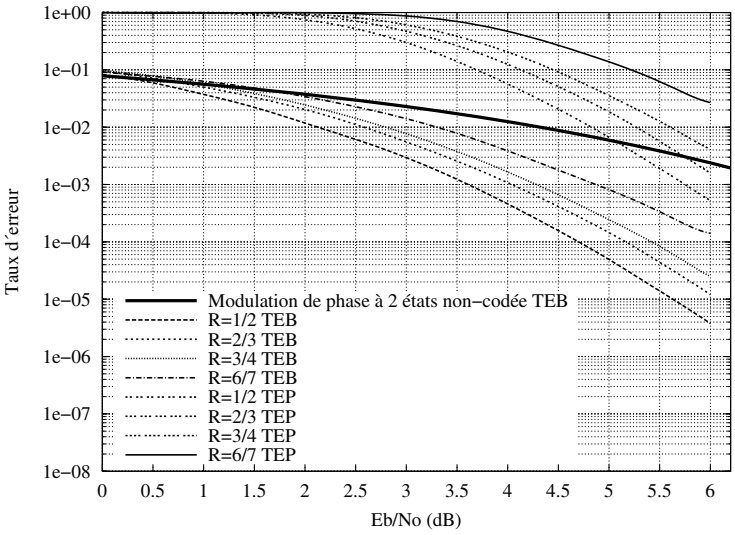


Figure 5.27 – Performance simulée d'un code CSRC à 8 états poinçonné, en fonction de son rendement.

Le choix du masque de poinçonnage influe bien évidemment sur les performances du code. Il est ainsi possible de privilégier une partie de la trame, transportant des données sensibles, en la poinçonnant faiblement au détriment d'une autre partie qui sera plus fortement poinçonnée. Un masque régulier est cependant souvent choisi car il est simple à mettre en œuvre.

Le poinçonnage est donc une technique souple et facile à mettre en œuvre. Le codeur et le décodeur restant identiques quel que soit le poinçonnage appliqué, il est possible de modifier le rendement de codage à tout moment. Certaines applications utilisent cette souplesse pour adapter, au fil de l'eau, le rendement au canal et/ou à l'importance des données transmises.

5.6 Bibliographie

[5.1] P. Elias, « Coding for Noisy Channels », *IRE conv. Rec.*, vol. 3, pt. 4, pp. 37-46, 1955.

[5.2] J. M. Wozencraft, « Sequential Decoding for Reliable Communication », *IRE Nat. Conv. Rec.*, vol. 5, pt. 2, pp. 11-25, 1957.

[5.3] R. M. Fano, « a Heuristic Discussion of Probabilistic Decoding », *IEEE Transaction on Information Theory*, vol. IT-9, pp. 64-74, April 1963.

[5.4] A. J. Viterbi, « Error bounds for convolutional codes and an asymptotically optimum decoding algorithm », *IEEE Transaction on Information Theory*, vol. IT-13, pp. 260-269, Apr. 1967.

[5.5] L. R. Bahl, J. Cocke, F. Jelinek and J. Raviv, « Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate », *IEEE Transactions on Information Theory*, vol. IT-20, pp. 284-287, March 1974.

[5.6] G. D. Forney., « Convolutional codes I : Algebraic structure », *IEEE Transactions on Information Theory*, vol. IT-16, pp. 720-738, November 1970.

[5.7] C. Berrou, A. Glavieux and P. Thitimajshima, « Near Shannon limit error-correcting coding and decoding : turbo-codes », *Proceedings of IEEE ICC'93*, Geneva, pp. 1064-1070, 1993.

[5.8] R. Johanneson and K. Sh. Zigangirov, « Fundamentals of Convolutional Coding », *IEEE Press*.

[5.9] P. Thitimajshima, *Les codes convolutifs récursifs systématiques et leur application à la concaténation parallèle*, thèse de doctorat en électronique, n ° d'ordre 284, Université de Bretagne Occidentale, Décembre 1993.

[5.10] A. Glavieux, *Codage de canal, des bases théoriques aux turbocodes*, Hermès-Science, 2005.

[5.11] G. D. Forney, « The Viterbi algorithm », *Proceedings of the IEEE*, vol. 61, n ° 3, pp. 268-278, March 1973.

[5.12] G. Battail, « Pondération des symboles décodés par l'algorithme de Viterbi », *Annales des Télécommunications*, vol. 42, n ° 1-2, pp. 31-38, Jan.-Fév. 1987.

[5.13] J. Hagenauer and P. Hoeher, « A Viterbi algorithm with soft-decision outputs and its applications », *IEEE Global Communications Conference*, Globecom'89, Dallas, Texas, Nov. 1989, pp. 1680-1686.

[5.14] C. Berrou, P. Adde, E. Angui and S. Faudeuil, « A low complexity soft-output Viterbi decoder architecture », *Proceedings of IEEE ICC'93*, Geneva, pp. 737-740, 1993.

[5.15] H. H. Ma and J. K. Wolf, « On tail biting convolutional codes », *IEEE Transactions on Communications*, vol. COM-34, pp. 104-111, Feb. 1986.

Chapitre 6

Concaténation de codes

Dans les chapitres précédents, des lois élémentaires de codage telles que BCH, Reed-Solomon ou CSRC ont été présentées. La plupart de ces codes élémentaires sont asymptotiquement bons, en ce sens que leurs distances minimales de Hamming (DMH) peuvent être rendues aussi grandes que l'on veut, en augmentant suffisamment le degré des polynômes générateurs. La complexité des décodeurs est malheureusement rédhibitoire pour les degrés de polynômes qui garantiraient les DMH requises par les applications pratiques.

Un moyen simple de disposer de codes à grande DMH et néanmoins aisément décodables est de combiner plusieurs codes élémentaires de taille raisonnable, de telle sorte que le code global résultant possède un pouvoir de correction élevé. Le décodage s'effectue par étapes, chacune d'entre elles correspondant à une des étapes de codage élémentaire. Le premier schéma de codage composite fut proposé par Forney durant son travail de thèse en 1965, et appelé concaténation de codes [6.1]. Dans ce schéma, un premier codeur, dit codeur extérieur, fournit un mot de code qui est ensuite recodé par un deuxième codeur, dit codeur intérieur. Si les deux codes sont systématiques, le code concaténé est lui-même systématique. Dans la suite de ce chapitre, seuls des codes systématiques seront considérés.

La figure 6.1(a) représente un code concaténé, tel qu'imaginé par Forney, et le décodeur par étapes correspondant. Le choix le plus judicieux des codes constituants se porte sur un code algébrique, typiquement Reed-Solomon, pour le code extérieur, et un code convolutif pour le code intérieur. Le décodeur intérieur est alors le décodeur de Viterbi, qui tire aisément profit des valeurs souples fournies par le demodulator, et le décodeur extérieur, qui travaille sur des symboles de plusieurs bits (par exemple 8 bits), peut s'accommoder d'erreurs en rafales à la sortie du premier décodeur. Une fonction de permutation ou d'entrelacement insérée entre les deux codeurs, et sa fonction inverse placée entre les deux décodeurs, peuvent augmenter très nettement la robustesse du code concaténé (figure 6.1(b)). Un tel schéma de codage a rencontré beaucoup de succès dans des applications aussi variées que les transmissions en espace

lointain et la diffusion de télévision numérique, satellitaire et terrestre. C'est en particulier le schéma de codage adopté dans de nombreux pays pour la télévision numérique terrestre [6.2].

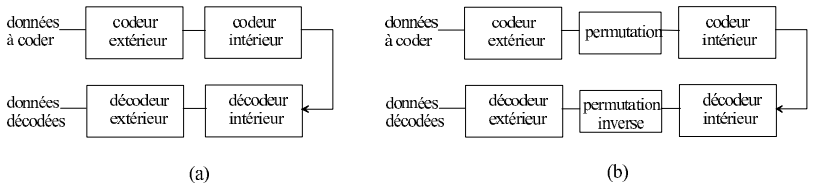


Figure 6.1 – Code concaténé en série, (a) sans et (b) avec permutation intercalaire. Dans les deux cas, la sortie du codeur extérieur est entièrement recodée par le codeur intérieur.

La concaténation de codes, dans cette première version, est dite aujourd'hui *concaténation série* (CS). Son décodage, tel que représenté dans les figures 6.1 n'est pas optimal. En effet, même si, localement, les deux décodeurs élémentaires sont optimaux, l'enchaînement simple de ces deux décodages n'est pas globalement optimal, le décodeur intérieur ne tirant pas profit de la redondance produite par le code extérieur. C'est cette observation, plutôt tardive dans l'histoire de la théorie de l'information, qui a conduit au développement de nouveaux principes de décodage, à commencer par le turbo-décodage. On sait maintenant décoder, de manière quasi-optimale, toutes sortes de schémas concaténés, à la seule condition que les décodeurs des codes élémentaires soient de type *SISO* (*soft-in/soft-out*). Dans ce cens, on peut remarquer que le concept de concaténation a beaucoup évolué ces dernières années, pour aller vers une notion plus large de codage multi-dimensionnel. Ici, la *dimension d'un code*, à ne pas confondre avec la longueur (k) du message d'information que l'on appelle également dimension, est le nombre de codes élémentaires utilisés dans la production du mot de code final.

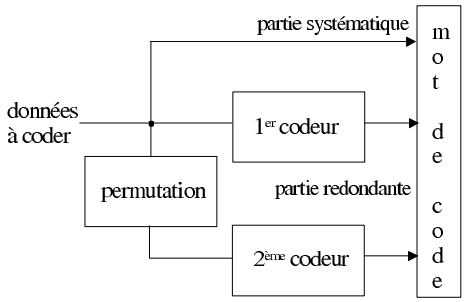


Figure 6.2 – Concaténation parallèle de codeurs systématiques.

Une nouvelle forme de concaténation, dite *concaténation parallèle* (CP), a été introduite au début des années 1990 pour élaborer les turbocodes [6.3]. La

figure 6.2 représente une CP de dimension 2, qui est la dimension classique des turbocodes. Dans ce schéma, le message est codé deux fois, dans son ordre naturel et dans un ordre permuté. La partie redondante du mot de code est formée par la réunion des sorties redondantes des deux codeurs. La CP diffère de la CS par plusieurs aspects, détaillés dans la section suivante.

6.1 Concaténation parallèle et concaténation série

En se limitant à la dimension 2, la CP , qui associe deux codes de rendements élémentaires R_1 (code C_1) et R_2 (code C_2), a un rendement global de codage :

$$R_p = \frac{R_1 R_2}{R_1 + R_2 - R_1 R_2} = \frac{R_1 R_2}{1 - (1 - R_1)(1 - R_2)} \tag{6.1}$$

Ce rendement est plus élevé que le rendement global R_s d'un code concaténé en série ($R_s = R_1 R_2$), pour des valeurs identiques de R_1 et R_2, et la différence est d'autant plus grande que les rendements de codage sont faibles. On en déduit qu'à même pouvoir de correction des codes élémentaires, la concaténation parallèle offre un meilleur rendement de codage, mais cet avantage s'amenuise lorsque les rendements considérés tendent vers 1. Lorsque la dimension du code composite augmente, l'écart entre R_p et R_s augmente également. Par exemple, trois codes élémentaires de rendement $1/2$ forment un code concaténé de rendement global $1/4$ en parallèle et $1/8$ en série.Voilà pourquoi il ne semble pas avantageux de porter la dimension d'un code concaténé en série au-delà de 2, sauf pour des rendements très proches de l'unité.

Toutefois, avec la CS, la partie redondante d'un mot traité par le décodeur extérieur a bénéficié de la correction du ou des décodeur(s) qui le précède(nt). À première vue donc, le pouvoir de correction d'un code concaténé en série semble être supérieur à celui d'un code concaténé en parallèle, dans lequel les valeurs représentatives de la partie redondante ne sont jamais corrigées. En d'autres termes, la DMH d'un code concaténé en série doit normalement être supérieure à celle d'un code concaténé en parallèle. On se trouve donc devant le dilemme énoncé dans le chapitre 1 : la CP affiche une meilleure performance dans la zone de convergence (près de la limite théorique) car le rendement de codage est plus favorable, la CS se comporte mieux à faible taux d'erreurs grâce à une plus grande DMH. Des solutions de codage à base de CS de codes convolutifs ont été étudiées [6.3], qui peuvent être une alternative intéressante aux turbocodes classiques, lorsque sont recherchés de faibles taux d'erreurs. Les codes convolutifs concaténés en série ne seront toutefois pas détaillés dans la suite de cet ouvrage.

Lorsque les parties redondantes des mots de codes intérieur et extérieur font toutes deux l'objet d'un codage supplémentaire, la concaténation est dite doublement série. L'exemple le plus connu d'une telle structure de codage est le code produit, qui met en oeuvre des codes BCH (voir chapitres 4 et 8). Des

structures mixtes, combinant concaténations parallèle et série ont également été proposées [6.4]. De plus, des codes élémentaires concaténés peuvent être de nature différente, par exemple un code convolutif et un code BCH [6.5]. On parle alors de code concaténé hybride. Dès lors que les décodeurs élémentaires acceptent et produisent des valeurs pondérées, toutes sortes de schémas mixtes et/ou hybrides peuvent être imaginés.

Alors que la CS peut utiliser des codes systématiques ou non systématiques indifféremment, la concaténation parallèle emploie des codes systématiques. S'il s'agit de codes convolutifs, au moins l'un de ces codes doit être récursif, pour une raison fondamentale liée au poids minimal d'entrée $w_{\min}$, qui n'est que 1 pour les codes non récursifs mais vaut 2 pour les codes récursifs (cf. chapitre 5). Pour nous en convaincre, observons la figure 6.3 qui représente deux codes systématiques non récursifs, concaténés en parallèle. La séquence d'entrée est « tout 0 » (séquence de référence) excepté en une position. Cet unique « 1 » perturbe la sortie du codeur C_1 pendant un laps de temps court, égal à la longueur de contrainte 4 du codeur. L'information redondante Y_1 est pauvre, relativement à cette séquence particulière, car elle ne contient que 3 valeurs différentes de 0. Après permutation, quelqu'elle soit, la séquence est toujours « tout 0 », excepté en une seule position. De nouveau, ce « 1 » perturbe la sortie du codeur C_2 pendant un laps de temps égal à la longueur de contrainte, et la redondance Y_2 délivrée par le second code est aussi peu informative que la première. En fait, la distance minimale de ce code à deux dimensions n'est pas plus élevée que celle d'un code unique, de même rendement que celui du code concaténé. Si l'on remplace au moins l'un des deux codeurs non récursifs par un codeur récursif, la séquence « tout 0 » sauf en une position n'est plus une séquence *RTZ* pour ce codeur récursif, et la redondance qu'il produit alors est de poids bien plus élevé.

Ce qui vient d'être expliqué sur la CP de codes convolutifs non récursifs laisse imaginer que le choix de codes élémentaires pour la CP en général est restreint. À titre de nouvel exemple, construisons un code concaténé en parallèle à partir du code de Hamming étendu défini par la figure 1.1 et la table de codage 1.1. Le message d'information contient 16 bits, qui sont rangés dans un tableau carré 4x4 (figure 6.4(a)). Chaque ligne, ainsi que chaque colonne, est codée par le code élémentaire de Hamming. Les bits de parité horizontaux et verticaux sont notés $r_{i,j}$ et $r'_{i,j}$, respectivement Le rendement de codage global est 1/3. Le décodage d'un tel code peut être mené suivant les principes du turbo-décodage (décodage local optimal selon le maximum de vraisemblance et échanges répétés d'informations extrinsèques).

La DMH du code est donnée par le motif d'erreurs de poids d'entrée 1 (figure 6.4(b)). Quelle que soit la place du 1 dans le message d'information, le poids est 7. Le facteur de mérite $Rd_{\min}$ est donc égal à 7x(1/3), à comparer avec le facteur de mérite du code élémentaire 4x(1/2). Le gain asymptotique n'a donc pas été extraordinairement augmenté par le truchement de la concaténation (précisément 0,67 dB), et cela s'est fait par une réduction importante du rendement de codage. Si l'on souhaite conserver le même rendement global

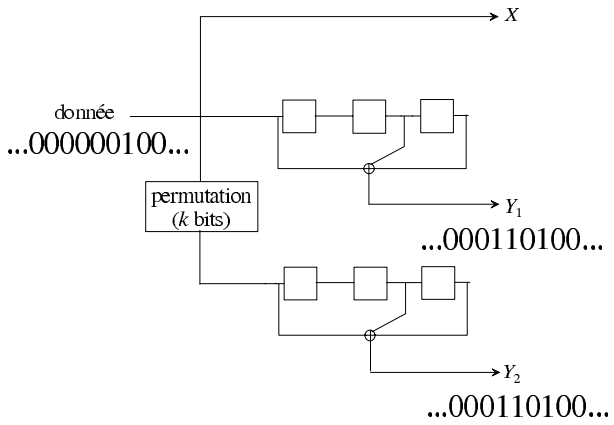


Figure 6.3 – La concaténation parallèle de codes systématiques non récursifs constitue un code pauvre vis-à-vis des séquences d'information de poids 1. Dans cet exemple, les symboles de redondance Y_1 et Y_2 ne contiennent chacun que 3 valeurs différentes de 0.

$d_{0,0}$	$d_{0,1}$	$d_{0,2}$	$d_{0,3}$	$r_{0,0}$	$r_{0,1}$	$r_{0,2}$	$r_{0,3}$
$d_{1,0}$	$d_{1,1}$	$d_{1,2}$	$d_{1,3}$	$r_{1,0}$	$r_{1,1}$	$r_{1,2}$	$r_{1,3}$
$d_{2,0}$	$d_{2,1}$	$d_{2,2}$	$d_{2,3}$	$r_{2,0}$	$r_{2,1}$	$r_{2,2}$	$r_{2,3}$
$d_{3,0}$	$d_{3,1}$	$d_{3,2}$	$d_{3,3}$	$r_{3,0}$	$r_{3,1}$	$r_{3,2}$	$r_{3,3}$
$r'_{0,0}$	$r'_{0,1}$	$r'_{0,2}$	$r'_{0,3}$				
$r'_{1,0}$	$r'_{1,1}$	$r'_{1,2}$	$r'_{1,3}$				
$r'_{2,0}$	$r'_{2,1}$	$r'_{2,2}$	$r'_{2,3}$				
$r'_{3,0}$	$r'_{3,1}$	$r'_{3,2}$	$r'_{3,3}$				

(a)

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

(b)

Figure 6.4 – Concaténation parallèle de codes de Hamming étendus (rendement global : 1/3). À droite : un motif d'erreurs de poids d'entrée 1 et de poids total 7.

de 1/2, une partie de la redondance doit être poinçonnée. On peut choisir, par exemple, de ne pas transmettre les 16 symboles présents dans les deux dernières colonnes et les deux dernières lignes du tableau de la figure 6.4(a). La DMH chute alors à la valeur 3, c'est-à-dire moins que la DMH du code élémentaire. La CP n'a donc pas d'intérêt dans ce cas.

Toujours à partir du code de Hamming étendu, une concaténation doublement série peut être élaborée sous la forme d'un code produit (figure 6.5(a)). Dans ce schéma, les parties redondantes des mots de code horizontaux et verticaux sont elles-mêmes recodées par des codes élémentaires, qui produisent des symboles de redondance notés $w_{i,j}$. Une propriété algébrique avantageuse de

ce code produit est l'identité des symboles de redondance issus du deuxième niveau de codage, dans les directions horizontale et verticale. La DMH du code, qui a un rendement global 1/4, est à nouveau donnée par les motifs d'erreurs de poids d'entrée 1 et vaut 16, soit le carré de la DMH du code élémentaire (figure 6.5(b)). Le facteur de mérite $Rd_{\min} = 4$ a donc été nettement augmenté par rapport à la concaténation parallèle. Tenter d'augmenter le rendement de ce code, par un poinçonnage des symboles de redondance, tout en conservant une bonne DMH est voué à l'échec.

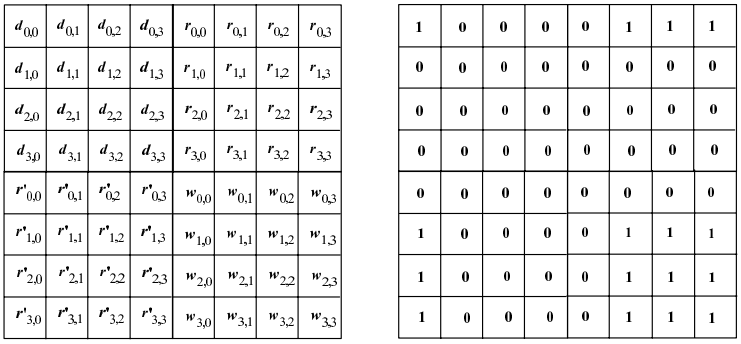


Figure 6.5 – Concaténation doublement série (code produit) de codes de Hamming étendus (rendement global : 1/4). À droite : un motif d'erreurs de poids d'entrée 1 et de poids total 16.

En conclusion, la concaténation parallèle ne s'accommode pas de n'importe quels codes élémentaires. Seuls, aujourd'hui, les codes convolutifs récursifs systématiques sont utilisés dans ce type de concaténation, de dimension 2 le plus souvent. La concaténation série peut offrir de grandes DMH. Le choix des codes est plus large : codes convolutifs, récursifs ou non, codes BCH ou de Reed-Solomon. Cependant, à mêmes rendements de codage des codes élémentaires, la concaténation série a un rendement global plus faible que la concaténation parallèle.

6.2 Concaténation parallèle et codage *LDPC*

Les codes *LDPC*, qui sont détaillés dans le chapitre 9, sont des codes dont les lignes et les colonnes de la matrice de contrôle contiennent peu de 1. Les codes *LDPC* peuvent être vus comme une concaténation multiple de $n - k$ relations de parité contenant peu de variables. Ce n'est pas ici une concaténation au sens où nous l'avons défini plus haut, puisque les relations de parité contiennent plusieurs variables de redondance et que ces variables apparaissent dans plusieurs relations. On ne peut donc assimiler les codes *LDPC* à des schémas usuels de concaténation série ou parallèle. En revanche, on peut, à l'instar de MacKay

[6.6], observer qu'un turbocode est un code *LDPC*. Un code CSR de polynômes générateurs $G_X(D)$ (récursivité) et $G_Y(D)$ (redondance), dont l'entrée est X et la sortie redondante Y, est caractérisée par la relation de parité glissante :

$$G_Y(D)X(D) = G_X(D)Y(D) \tag{6.2}$$

En faisant appel à la technique de fermeture circulaire, la matrice de contrôle de parité prend une forme très régulière, telle que celle représentée en figure 6.6 pour un rendement de codage 1/2, et en choisissant $G_X(D) = 1 + D + D^3$ et $G_Y(D) = 1 + D^2 + D^3$. Un code CSRC est donc un code *LDPC* car la matrice de contrôle est creuse. Ce n'est certes pas un bon code *LDPC*, car la matrice de contrôle ne respecte pas certaines propriétés sur la place des 1. En particulier, les 1 d'une même ligne sont très voisins les uns des autres, ce qui n'est pas favorable à la méthode de décodage par propagation de croyance.

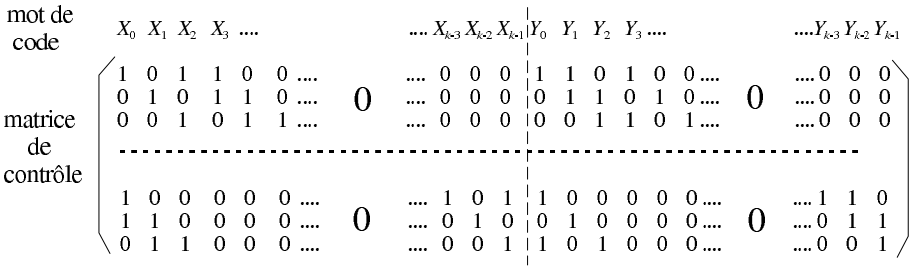


Figure 6.6 – Matrice de contrôle d'un code convolutif à terminaison circulaire. Un code convolutif, de type CSRC en particulier, peut être vu comme un code *LDPC* car la matrice de contrôle est creuse.

Une concaténation parallèle de codes CSRC, c'est-à-dire un turbocode, est aussi un code *LDPC*, puisque elle associe des codes élémentaires qui sont du type *LDPC*. Il y a, bien sûr, plus de degrés de liberté dans la construction d'un code *LDPC*, car chaque 1 de la matrice de contrôle peut être placé indépendamment des autres. En revanche, le décodage d'un code convolutif, par le biais d'un algorithme basé sur le treillis ne connaît pas le problème de corrélation entre symboles successifs qu'un décodage de type propagation de croyance soulèverait, s'il était appliqué à un code convolutif simple. Un turbocode ne peut donc être décodé comme un code *LDPC*.

6.3 Les permutations

Les fonctions de permutation ou d'entrelacement, utilisées entre codeurs élémentaires dans un schéma concaténé, ont un double rôle. D'une part, elles assurent, à la sortie de chaque décodeur élémentaire, une dispersion temporelle des erreurs qui peuvent être en produites en rafales. Ces paquets d'erreurs

deviennent alors erreurs isolées pour le décodeur qui suit, avec des effets de corrélation bien amoindris. Cette technique de dispersion des erreurs est utilisée dans un cadre plus large que celui du codage de canal. On l'utilise avec profit par exemple pour réduire les effets des atténuations plus ou moins longues dans les transmissions affectées d'évanouissements, et plus généralement dans des situations où des perturbations peuvent altérer des symboles consécutifs. D'autre part, en liaison étroite avec les caractéristiques des codes composants, la permutation est conçue pour que la DMH du code concaténé soit la plus grande possible. C'est une problématique de pure mathématique qui associe géométrie, algèbre et combinatoire et qui n'a pas encore, dans la plupart des cas, trouvé de réponse définitive. Les sections 7.3.2 et **??** développent le thème de la permutation pour les turbocodes et du graphe pour les codes *LDPC*, respectivement.

6.4 Turbo mots croisés

Pour terminer ce chapitre, voici un exemple de concaténation parallèle bien connu de tous : les mots croisés. Une grille dont le contenu a été fortement altéré lors de sa retranscription est représentée en figure 6.7. Nous disposons heureusement d'une définition correcte pour chaque ligne et pour chaque colonne et d'un dictionnaire de synonymes.

	1	2	3	4	5
I	**T**	**H**	**E**	**T**	**A**
II	**A**	**R**	**I**	**O**	**N**
III	**L**	**I**	**E**	**S**	**T**
IV	**S**	**E**	**N**	**T**	**N**
V	**E**	**R**	**S**	**U**	**L**

Horizontalement	**Verticalement**
I. Grecque.	1. Ballet.
II. Aéronef.	2. Lavabo.
III. Réfutent.	3. Attaches.
IV. Chemin.	4. Taille.
V. Anneaux.	5. Piliers

Figure 6.7 – Une grille de mots croisés dont le contenu est erroné et dont les définitions sont correctes.

Pour corriger (ou décoder) cette grille, un travail itératif en ligne et en colonne doit être effectué. La règle de décodage élémentaire est la suivante : « S'il existe un mot du dictionnaire, synonyme ou équivalent à la définition donnée, ne différant du mot contenu dans la grille que par une lettre au plus, alors ce synonyme est adopté ».

Les définitions horizontales permettent de commencer une correction en ligne de la grille (figure 6.8(a)) :

I. THETA est bien une lettre grecque.

II. En remplaçant le R par un V, nous obtenons AVION, synonyme d'aéronef.

III. Il y a sûrement plus d'une lettre erronée.

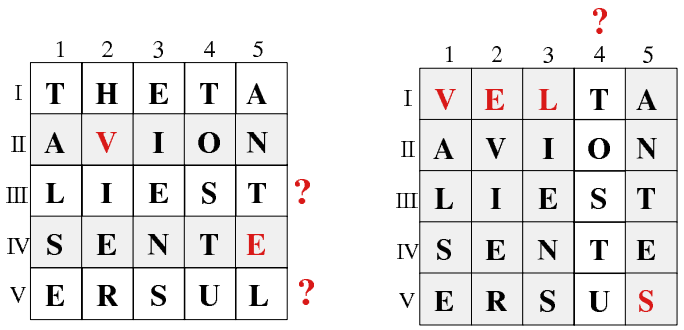


Figure 6.8 – Première itération du processus de décodage ligne - colonne.

IV. SENTE est un petit chemin.

V. Il n'y a pas de synonyme évident.

À l'issue de ce décodage ligne, trois mots sont corrects ou ont été corrigés, et deux sont encore inconnus. À l'aide des définitions verticales, un décodage colonne peut être maintenant réalisé (figure 6.8(b)) :

1. VALSE répond à la définition.
2. En remplaçant le H par un E, nous obtenons EVIER.
3. Des LIENS sont des attaches.
4. Il y a au moins 2 lettres erronées. Aucune correction possible.
5. Les ANTES sont des piliers dans les temples grecs ou romains.

À l'issue de ce décodage, il reste encore des mots inconnus et il faut procéder à une deuxième itération du processus de décodage ligne - colonne (figure 6.9).

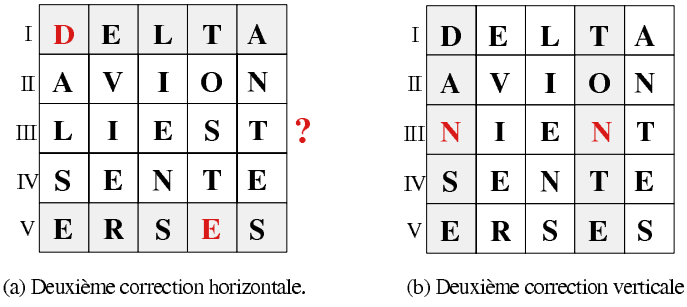


Figure 6.9 – Deuxième itération du processus de décodage ligne - colonne.

Le décodage ligne conduit au résultat suivant (figure 6.9(a)) :

I. DELTA est aussi une lettre grecque.

II. AVION est correct.

III. Il y a toujours au moins 2 lettres erronées.

IV. SENTE est correct.

V. Les ERSES sont des anneaux.

Après cette étape, il reste encore une ligne incorrecte. Il est possible de la corriger avec un nouveau décodage colonne (figure 6.9(b)).

1. Le ballet est une DANSE.
2. EVIER est correct.
3. LIENS est correct.
4. TONTE est synonyme de taille.
5. ANTES est correct.

À l'issue de cet ultime décodage, le mot inconnu de la ligne III est identifié : il s'agit de NIENT qui est bien synonyme de « réfutent ».

Un certain nombre de constatations peuvent être faites après cette expérience de décodage de code concaténé en parallèle.

- Pour atteindre le résultat correct, deux itérations du décodage ligne et du décodage colonne ont été nécessaires. Il aurait été dommage de s'arrêter après une itération. C'est pourtant bien ce que l'on fait dans le cas d'un code concaténé classique tel que celui de la figure 6.1.
- Un mot correct au départ (THETA est bien une lettre grecque) s'est avéré faux. De même, une correction effectuée lors de la première étape (VALSE) s'est révélée erronée. Il faut donc considérer les résultats intermédiaires avec prudence et éviter les décisions péremptoires. Dans les décodeurs itératifs modernes, cette prudence est mesurée par une probabilité qui n'est jamais exactement 0 ou 1.

6.5 Bibliographie

[6.1] G. D. Forney Jr., « Performance of concatenated codes », *Key papers in the development of coding theory* edited by E. R. Berlekamp, pp. 90-94, *IEEE Press*, 1974.

[6.2] DVB-Terrestrial, ETSI EN 302 296 V1.1.1 (2004-04).

[6.3] S. Benedetto, D. Divsalar, G. Montorsi and F. Pollara, « Serial concatenation of interleaved codes : performance analysis, design, and iterative decoding », *IEEE Trans. Info. Theory*, vol. 44, no. 3, pp. 909-926, May 1998.

[6.4] K. R. Narayanan and G. L. Stüber, « Selective serial concatenation of turbo codes », *IEEE Comm. Letters*, vol. 1, no. 5, pp. 136-139, Sept. 1997.

[6.5] P. Adde, R. Pyndiah and C. Berrou, « Performance of hybrid turbo codes », *Elect. Letters*, vol. 32, no. 24, pp. 2209-2210, Nov. 1996.

[6.6] D. J. C. MacKay, « Good error-correcting codes based on very sparse matrices », *IEEE Trans. Info. Theory*, vol. 45, no. 2, pp. 399-431, March 1999.

Chapitre 7

Turbocodes convolutifs

Le pouvoir de correction d'un code convolutif s'accroît lorsque la longueur du registre de codage augmente. Cela est mis en évidence dans la figure 7.1, qui fournit la performance de quatre codes CSR de mémoires respectives $\nu = 2, 4, 6$ et 8, pour des rendements 1/2, 2/3, 3/4 et 4/5, avec un décodage selon l'algorithme MAP. Pour chacun des rendements, le pouvoir de correction s'améliore avec l'augmentation de ν, au dessus d'un certain rapport signal à bruit que l'on peut confondre presque parfaitement avec la limite théorique calculée dans le chapitre 3 et identifiée ici par une flèche. Pour satisfaire les applications les plus courantes du codage de canal, une mémoire de l'ordre de 30 ou 40 serait nécessaire (à partir d'une certaine longueur de registre et pour un rendement de codage 1/2, la DMH (distance minimale de Hamming) d'un code convolutif de mémoire ν est de l'ordre de grandeur de ν). Si l'on savait décoder aisément un code convolutif à plus d'un milliard d'états, on ne parlerait plus beaucoup de codage de canal et ce livre n'existerait pas.

Un turbocode est un artifice de code visant à imiter un code convolutif de grande mémoire ν. Il est construit suivant le principe de l'adage *diviser pour régner*, c'est-à-dire par association de plusieurs petits codes CSR dont les décodages particuliers sont de complexité raisonnable. Un échange judicieux d'information entre les décodeurs élémentaires permet au décodeur composite d'approcher la performance du décodage à maximum de vraisemblance.

7.1 L'histoire des turbocodes

L'invention des turbocodes n'est pas l'aboutissement d'un développement mathématique. Elle est le résultat d'une démarche intuitive et expérimentale dont l'origine est à trouver dans les travaux de quelques chercheurs européens : Gérard Battail, Joachim Hagenauer et Peter Hoeher qui, à la fin des années 80 [7.1-7.4], soulignaient l'intérêt du traitement probabiliste dans les récepteurs. D'autres auparavant, principalement aux États-Unis : Peter Elias [7.5], Michael

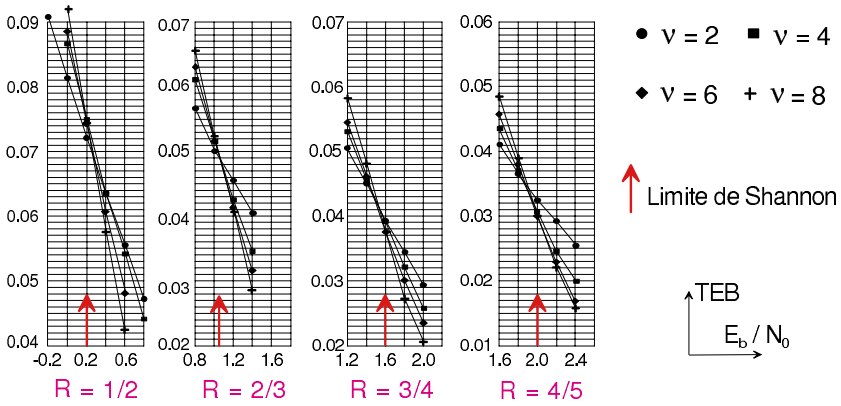


Figure 7.1 – Performance de codes convolutifs systématiques récursifs (CSR) pour différents rendements et quatre valeurs de la mémoire de code ν. Comparaison avec les limites de Shannon.

Tanner [7.6], Robert Gallager [7.7], etc. avaient imaginé plus tôt des procédés de codage et de décodage précurseurs des turbocodes.

Dans un laboratoire de l'École nationale supérieure des télécommunications (ENST) de Bretagne, Claude Berrou et Patrick Adde cherchaient à transcrire l'algorithme de Viterbi à sortie pondérée (*SOVA* : *Soft-Output Viterbi Algorithm*), proposé dans [7.2], en transistors MOS, de la manière la plus simple possible. Une solution convenable [7.8] fut trouvée après deux années qui permirent à ces chercheurs de se forger une opinion sur le décodage probabiliste. Claude Berrou, puis Alain Glavieux poursuivirent l'étude et observèrent, à la suite de Gérard Battail, qu'un décodeur à entrée et sortie pondérées pouvait être considéré comme un amplificateur de rapport signal à bruit et cela les encouragea à mettre en œuvre des concepts communément utilisés dans les amplificateurs, principalement la contre-réaction. La mise au point des turbocodes passa par de nombreuses étapes très pragmatiques et aussi par l'introduction de néologismes, comme « concaténation parallèle » ou « information extrinsèque », aujourd'hui communs dans le jargon de la théorie de l'information. Le dépôt du brevet en 1991 [7.9], et surtout la publication en 1993 [7.10] démontrant une performance à 0,5 dB de la limite de Shannon, secoua la communauté du codage. Un gain de près de 3 dB, par rapport aux solutions alors existantes, venait d'être apporté par une petite équipe inconnue, française de surcroît (France, pays de la rigueur mathématique *versus* turbocodes, invention pour le moins empirique). Il s'ensuivit une très sensible évolution des habitudes, comme le souligne Calderbank dans [7.11, p. 2573] : « *It is interesting to observe that the search for theoretical understanding of turbo codes has transformed coding theorists into experimental scientists* » [« Il est intéressant d'observer que la recherche de justification théorique des turbocodes a conduit les experts de la théorie des codes à se tourner vers l'expérimentation »].

On trouvera dans [7.12] une chronologie détaillée des idées successives qui apparurent dans la mise au point des turbocodes. Cette nouvelle technique de codage et de décodage fut d'abord baptisée turbo-code, avec le tiret pour signifier qu'il s'agissait d'un code décodé à la manière turbo (par analogie avec le moteur turbo qui utilise les gaz d'échappement pour en augmenter sa puissance). Le tiret étant peu usité dans la langue anglaise, cela devint *turbo code*, c'est-à-dire le code « turbo », ce qui n'a pas beaucoup de sens. En français aujourd'hui, turbocode s'écrit en un seul mot.

7.2 Concaténation multiple de codes CSR

Les codes aléatoires ont toujours constitué, depuis les travaux précurseurs de Shannon, une référence pour le codage correcteur d'erreurs (voir chapitre 3.1.5). Le codage aléatoire systématiqued'un bloc de k bits d'information, conduisant à un mot de code de longueur n, peut consister, en première étape et une fois pour toutes, à tirer au hasard et à mémoriser k marqueurs binaires de $n-k$ bits, dont l'adresse de mémorisation est notée i ($0 \leq i \leq k-1$). La redondance associée à un bloc quelconque d'information est alors formée par la sommation modulo 2 de tous les marqueurs dont l'adresse i est telle que le i-ième bit d'information vaut 1. En d'autres termes, les k marqueurs constituent les bases d'un espace vectoriel de dimension k. Le mot de code est finalement constitué par la concaténation des k bits d'information et des $n-k$ bits de redondance. Le rendement R du code est k/n. Cette construction du mot de code, très simple, s'appuie sur la propriété de linéarité de l'addition et conduit à des distances minimales élevées pour des valeurs suffisamment grandes de $n-k$. Parce que deux mots de code sont différents par au moins un bit d'information et que la redondance est tirée au hasard, la distance minimale moyenne est $1+\frac{n-k}{2}$. Cependant, la distance minimale de ce code étant une variable aléatoire, ses différentes réalisations peuvent être inférieures à cette grandeur. Une approximation simple et réaliste de la distance minimale effective est $\frac{n-k}{4}$.

Un moyen de construire un codeur quasiment aléatoire est représenté en figure 7.2. Il s'agit d'une concaténation parallèle multiple de codes Convolutifs Systématiques Récursifs Circulaires (CSRC, voir chapitre 5) [7.13]. La séquence de k données binaires est codée N fois par N codeurs CSRC, dans un ordre différent à chaque fois. Les permutations Π_j sont tirées au hasard, exceptée la première qui peut être la permutation identité. Chaque codeur élémentaire produit $\frac{k}{N}$ symboles de redondance (N étant un diviseur de k), le rendement global du code concaténé étant $1/2$.

La proportion des séquences d'entrée d'un codeur récursif construit à partir d'un générateur pseudo-aléatoire de mémoire ν, initialement positionné à l'état 0, qui remettent le registre dans ce même état à la fin du codage, est :

$$p_1 = 2^{-\nu} \tag{7.1}$$

car il y a 2^{ν} états de retour possibles, avec la même probabilité. Ces séquences,

appelées *RTZ* (*Return To Zero*, voir chapitre 5), sont des combinaisons linéaires de la séquence *RTZ* minimale, qui est donnée par le polynôme de récursivité du générateur $(1 + D + D^3$ dans le cas de la figure 7.2).

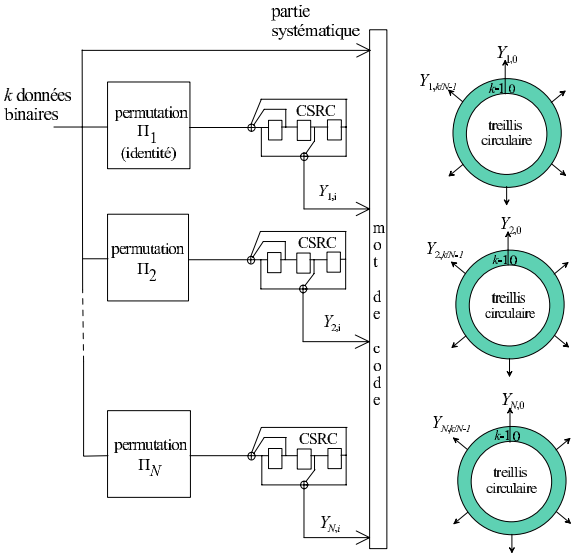


Figure 7.2 – Concaténation parallèle multiple de codes convolutifs systématiques récursifs circulaires (CSRC). Chaque codeur produit k/N symboles de redondance uniformément répartis sur le treillis circulaire. Rendement de codage global : 1/2.

La proportion des séquences *RTZ* pour le codeur multi-dimensionnel est abaissée à :

$$p_N = 2^{-N\nu} \tag{7.2}$$

car il faut que la séquence, après chaque permutation, reste *RTZ* pour les N codeurs.

Les autres séquences, de proportion $1 - p_N$, produisent des mots de code qui ont une distance d qui satisfait :

$$d > \frac{k}{2N} \tag{7.3}$$

Cette valeur du pire des cas suppose qu'une seule séquence permutée n'est pas *RTZ* et que la redondance Y prend la valeur 1 une fois sur deux en moyenne, sur le cercle correspondant. Si l'on prend par exemple $N = 8$ et $\nu = 3$, on obtient $p_8 \approx 10^{-7}$ et, pour des séquences à coder de longueur $k = 1024$, on a $d_{\min} = 64$, ce qui est une distance minimale bien suffisante si l'on se réfère aux courbes de la figure 3.6.

Le codage aléatoire peut donc être approché en utilisant de petits codes et des permutations aléatoires. Le décodage peut être effectué suivant le principe turbo, détaillé dans la section 7.4 pour $N = 2$. Le schéma de la figure 7.2 n'est

toutefois pas utilisé en pratique, pour des raisons liées à la performance et à la complexité du décodage. Tout d'abord, le seuil de convergence du turbo-décodeur, c'est-à-dire le rapport signal à bruit à partir duquel celui-ci peut commencer à corriger la plupart des erreurs, se dégrade lorsque la dimension de la concaténation croît. En effet, le principe même du turbo-décodage oblige à considérer l'un après l'autre, de manière itérative, les codes élémentaires. Leur taux de redondance diminuant lorsque la dimension du code composite croît, les premières étapes du décodage sont pénalisées par rapport à un code concaténé de simple dimension 2. Ensuite, la complexité et la latence du décodeur sont proportionnelles au nombre de codeurs élémentaires.

7.3 Les turbocodes

Fort heureusement, par rapport à ce qui précède, il n'est pas nécessaire de porter la dimension N à une grande valeur. En remplaçant la permutation aléatoire Π_2 par une permutation judicieusement élaborée, de bonnes performances peuvent être obtenues en se limitant à une dimension $N = 2$. C'est le principe du turbocode.

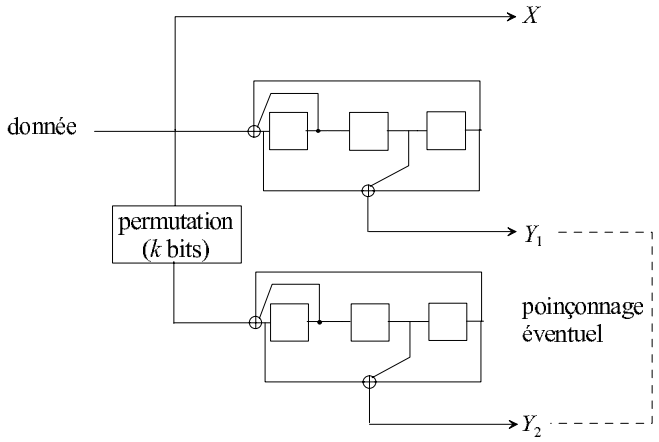


Figure 7.3 – Un turbocode binaire à mémoire $\nu = 3$ utilisant des codeurs CSR élémentaires identiques (polynômes 15, 13). Le rendement de codage naturel du turbocode, sans poinçonnage, est 1/3.

La figure 7.3 représente un turbocode, dans sa version la plus classique [7.10]. Le message binaire d'entrée, de longueur k, est codé, dans son ordre naturel et dans un ordre permuté, par deux codeurs CSR appelés C_1 et C_2, qui peuvent être terminés ou non. Dans cet exemple, les deux codeurs élémentaires sont identiques (polynômes générateurs 15 pour la récursivité et 13 pour la construction de la redondance) mais ce n'est pas une nécessité. Le rendement de codage naturel, sans poinçonnage, est 1/3. Pour obtenir des rendements

plus élevés, un poinçonnage des symboles de redondance Y_1 et Y_2 est effectué. Un autre moyen de disposer de rendements plus élevés est d'adopter des codes m-binaires (voir 7.5.2).

La fonction de permutation (Π) portant sur un message de taille finie k, le turbocode est par construction un code en bloc. Pour le distinguer toutefois des codes algébriques concaténés décodés à la manière « turbo », comme les codes produits et que l'on a appelés plus tard *turbocodes en bloc*, ce schéma de turbocodage est dit *convolutif* ou encore, plus techniquement, *PCCC* (*Parallel Concatenated Convolutional Code*).

Les arguments en faveur de ce schéma de codage (dont certains ont déjà été introduits dans le chapitre 6) sont les suivants :

1. Un décodeur de code convolutif est vulnérable aux erreurs survenant en paquets. Coder le message deux fois, suivant deux ordres différents (avant et après permutation), c'est rendre peu probable l'apparition simultanée de paquets d'erreurs à l'entrée des décodeurs de C_1 et de C_2. Si des erreurs groupées surviennent à l'entrée du décodeur de C_1, la permutation les disperse dans le temps et elles deviennent des erreurs isolées, aisément corrigibles, pour le décodeur de C_2. Le raisonnement tient également pour les paquets d'erreurs à l'entrée de ce second décodeur, qui correspondent, avant permutation, à des erreurs isolées. Ainsi le codage bi-dimensionnel réduit-il nettement, sur l'une au moins des deux dimensions, la vulnérabilité du codage convolutif vis-à-vis des perturbations groupées. Mais sur lequel des deux décodeurs s'appuyer pour prendre la décision finale ? Aucun critère ne permet d'accorder une plus grande confiance à l'un ou à l'autre. La réponse est donnée par l'algorithme « turbo » qui évite d'avoir à faire ce choix. Cet algorithme met en œuvre des échanges de probabilités entre les deux décodeurs et les contraint à converger, au fil de ces échanges, vers les mêmes décisions.
2. Comme on l'a vu dans la section 6.1, la concaténation parallèle conduit à un rendement de codage plus élevé que celui de la concaténation série. La concaténation parallèle est donc plus favorable lorsque des rapports signal à bruit proches des limites théoriques sont considérés, avec des taux d'erreurs visés moyens. Il peut en être autrement lorsque des taux d'erreurs très faibles sont recherchés car la DMH d'un code concaténé en série peut être plus grande.
3. La concaténation parallèle utilise des codes systématiques et au moins l'un de ces codes doit être récursif, pour des raisons également exposées dans la section 6.1
4. Les codes élémentaires sont de petits codes : codes à 16, 8, voire 4 états. Le décodage, même s'il met en œuvre des traitements probabilistes répétés, reste de complexité raisonnable.

La figure 7.4 représente les turbocodes utilisés en pratique et la table 7.2 recense les applications industrielles connues à ce jour. Les paramètres définissant un turbocode particulier sont les suivants :

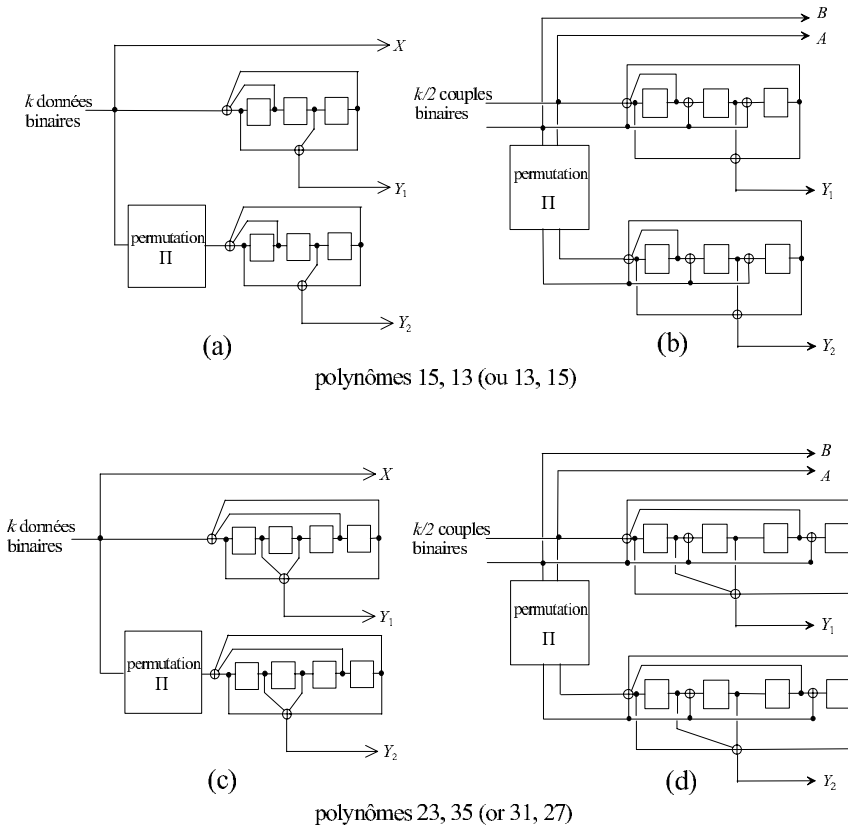


Figure 7.4 – Les turbocodes utilisés en pratique.

a– m est le nombre de bits dans les symboles appliqués au turbocodeur. Les applications connues à ce jour considèrent des symboles binaires ($m = 1$) ou double-binaires ($m = 2$).

b– Chacun des deux codeurs élémentaires C_1 et C_2 est caractérisé par
- sa mémoire de code ν
- ses polynômes générateurs de récursivité et de redondance
- son rendement

Les valeurs de ν sont en pratique inférieures ou égales à 4. Les polynômes générateurs sont généralement ceux que l'on utilise pour les codes convolutifs classiques et qui ont fait l'objet d'une littérature abondante dans les années 1980-90.

c– La manière dont on effectue la permutation est importante lorsque le taux d'erreurs binaires cible est inférieur à 10^{-5} environ. Au-dessus de cette valeur, la performance est peu sensible à la permutation, à condition bien sûr que celle-ci respecte au moins le principe de dispersion (cela peut être

Application	Turbocode	Terminaison	Polynômes	Rendements
CCSDS (espace lointain)	binaire, 16 états	*tail bits*	23, 33, 25, 37	1/6, 1/4, 1/3, 1/2
UMTS, CDMA200 (mobile 3G)	binaire, 8 états	*tail bits*	13, 15, 17	1/4, 1/3, 1/2
DVB-RCS (Return Channel on Satellite)	double binaire, 8 états	circulaire	15, 13	de 1/3 à 6/7
DVB-RCT (Return Channel on Terrestrial)	double binaire, 8 états	circulaire	15, 13	1/2, 3/4
Inmarsat (M4)	binaire, 16 états	aucune	23, 35	1/2
Eutelsat (skyplex)	double binaire, 8 états	circulaire	15, 13	4/5, 6/7
IEEE 802.16 (WiMAX)	double binaire, 8 états	circulaire	15, 13	de 1/2 à 7/8

Table 7.2 – Les applications standardisées des turbocodes convolutifs.

par exemple une permutation régulière). Pour un taux d'erreurs cible faible ou très faible, la performance est dictée par la distance minimale du code et celle-ci est très dépendante de la permutation Π.

d– Le motif de poinçonnage doit être le plus régulier possible, à l'image de ce qui se pratique pour les codes convolutifs classiques. Toutefois, il peut être avantageux d'avoir un motif de poinçonnage faiblement irrégulier quand on recherche de très faibles taux d'erreurs et lorsque la période de poinçonnage est un diviseur de la période du polynôme générateur de récursivité ou de parité.

Le poinçonnage s'effectue classiquement sur les symboles de redondance. Il peut être envisagé de plutôt poinçonner les symboles d'information, pour augmenter la distance minimale du code. Cela se fait au détriment du seuil de convergence du turbo décodeur. De ce point de vue, en effet, poinçonner des données partagées par les deux décodeurs est plus pénalisant que poinçonner des données qui ne sont utiles qu'à l'un des décodeurs.

Ce qui sera à considérer de près dans la construction d'un turbocode et dans son décodage, ce sont les séquences *RTZ*, dont les poids de sortie limitent la

distance minimale du code et en fixent les performances asymptotiques. Dans la suite, il sera admis que les motifs d'erreurs qui ne sont pas *RTZ* ne contribuent pas à la DMH du turbocode et n'auront donc pas à être recensés.

7.3.1 La terminaison des codes constituants

Pour un turbocode, la fermeture de deux treillis est à prendre en compte et les solutions présentées dans la section 5.5.1 peuvent être envisagées :

Ne rien faire de particulier concernant les états terminaux : les informations situées à la fin du bloc, aussi bien dans l'ordre naturel que dans l'ordre permuté, sont alors moins bien protégées. Ceci conduit à une diminution du gain asymptotique mais cette dégradation, qui est fonction de la taille du bloc, peut être compatible avec certaines applications. Il est à noter que la non-fermeture des treillis pénalise plus fortement le TEP (Taux d'Erreurs de Paquets) que le TEB.

Fermer le treillis d'un ou des deux codes élémentaires à l'aide de bits de bourrage : les standards CCSDS [7.14] et UMTS [7.15] utilisent cette technique. Les bits assurant la fermeture d'un des deux treillis ne sont pas utilisés dans l'autre codeur. Ces bits ne sont donc pas turbocodés ce qui conduit mais dans une moindre mesure, aux mêmes inconvénients que ceux présentés dans le cas précédent. De plus, la transmission des bits de fermeture entraîne une diminution du rendement de codage et donc, de l'efficacité spectrale.

Utiliser un entrelacement permettant une fermeture automatique du treillis : il est possible de fermer automatiquement, sans ajout de bits de fermeture, le treillis d'un turbocode en transformant légèrement le schéma de codage (auto-concaténation) et en utilisant un entrelacement respectant certaines règles de périodicité. Cette solution décrite dans [7.16] ne diminue pas l'efficacité spectrale mais impose des contraintes sur l'entrelacement qui rendent difficile la maîtrise des performances à faibles taux d'erreurs.

Adopter un codage circulaire : un codeur de code convolutif circulaire garantit que l'état initial et l'état final du registre sont identiques. Le treillis prend alors la forme d'un cercle ce qui, du point de vue du décodeur, peut être considéré comme un treillis de longueur infinie [7.17,7.18]. Ce procédé de fermeture, déjà connu sous le nom de *tail-biting* pour les codes non récursifs, offre deux avantages majeurs :

- Au contraire des autres techniques, la fermeture circulaire ne présente aucun effet de bord : tous les bits du message sont protégés de la même manière et tous sont doublement codés par le turbocode. Il n'y a donc pas lieu, lors de la conception de la permutation, d'accorder une importance particulière à tel ou tel bit, ce qui conduit à des modèles de permutation plus simples.
- Les séquences qui ne sont pas *RTZ* ont une influence sur l'ensemble du cercle : un symbole de parité sur deux, en moyenne, est modifié tout le long du bloc. Pour des valeurs typiques de k (quelques centaines ou plus), le poids de sortie correspondant est donc très élevé et ces motifs d'erreurs

ne contribuent pas à la DMH du code, comme on l'a déjà mentionné à la fin de la section précédente. Sans terminaison ou avec une terminaison par bits de bourrage, seule la partie du bloc après le début de la séquence non *RTZ* a une incidence sur les symboles de parité.

À ces deux avantages, s'ajoute bien sûr l'intérêt de n'avoir à transmettre aucune information supplémentaire sur la terminaison et donc de ne rien perdre en efficacité spectrale.

La technique de terminaison circulaire a été retenue dans les standards DVB-RCS et DVB-RCT [7.19,7.20] par exemple.

7.3.2 La fonction de permutation

Qu'on l'appelle entrelacement ou permutation, la technique consistant à disperser des données dans le temps a toujours rendu de grands services en communications numériques. On l'utilise avec profit par exemple pour réduire les effets des atténuations plus ou moins longues dans les transmissions affectées d'évanouissements et plus généralement dans des situations où des perturbations peuvent altérer des symboles consécutifs. Dans le cas des turbocodes aussi, la permutation permet de lutter efficacement contre l'apparition de paquets d'erreurs, sur l'une au moins des dimensions du code composite. Mais son rôle ne s'arrête pas là : elle détermine aussi, en relation étroite avec les propriétés des codes constituants, la distance minimale du code concaténé.

Considérons le turbocode représenté en figure 7.3. La plus mauvaise des permutations qu'on puisse utiliser est bien sûr la permutation identité, qui minimise la diversité du codage (on a alors $Y_1 = Y_2$). À l'opposé, la meilleure des permutations que l'on pourrait imaginer mais qui n'existe probablement pas [7.21], permettrait au code concaténé d'être équivalent à une machine séquentielle dont le nombre d'états irréductibles serait 2^{k+6}. Il y a en effet $k+6$ éléments binaires de mémorisation dans la structure : k pour la mémoire de permutation et 6 pour les deux codes convolutifs. Si on pouvait assimiler cette machine séquentielle à un codeur convolutif et pour les valeurs usuelles de k, le nombre d'états correspondant serait très grand, en tout cas assez grand pour garantir une large distance minimale. Par exemple, un codeur convolutif avec une mémoire de code de 60 (10^{18} états !) affiche une distance libre de l'ordre de la centaine (pour $R = 1/2$), ce qui est bien suffisant.

Ainsi, de la plus mauvaise à la meilleure des permutations, le choix est large et l'on n'a pas encore découvert de permutation parfaite. Cela dit, de bonnes permutations ont quand même pu être définies pour élaborer des schémas de turbocodage normalisés.

Il existe deux manières de spécifier une permutation, la première par des équations liant les adresses avant et après permutation, la seconde par un tableau (*look-up table*) fournissant la correspondance entre les adresses. La première est préférable du point de vue de la simplicité dans la spécification du turbocode (les comités de normalisation sont sensibles à cet aspect) mais la

seconde peut conduire à de meilleurs résultats car le degré de liberté est généralement plus grand dans le travail de conception de la permutation.

La permutation régulière

Le point de départ dans la conception d'un entrelacement est la permutation régulière, qui est décrite en figure 7.5 sous deux formes différentes. La première suppose que le bloc contenant k bits peut être organisé comme un tableau de M lignes et N colonnes. L'entrelacement consiste alors à écrire les données dans une mémoire *ad hoc*, ligne par ligne, et à les lire colonne par colonne (figure 7.5(a)). La seconde s'applique sans hypothèse sur la valeur de k. Après écriture des données dans une mémoire linéaire (adresse i, $0 \leq i \leq k-1$), le bloc est assimilé à un cercle, les deux extrémités ($i = 0$ et $i = k-1$) étant alors contiguës (figure 7.5(b)). Les données binaires sont alors extraites de telle sorte que la j-ième donnée lue ait été préalablement écrite à la place i, de valeur :

$$i = \Pi(j) = Pj + i_0 \bmod k \tag{7.4}$$

où P est un entier premier avec k et i_0 est l'indice de départ[1].

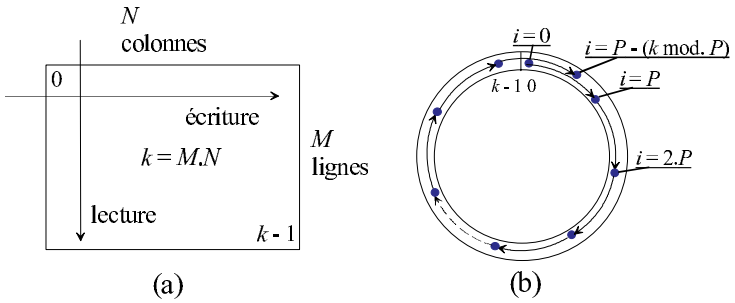


Figure 7.5 – Permutation régulière sous formes rectangulaire (a) et circulaire (b).

Pour une permutation circulaire, définissons la distance spatiale cumulée $S(j_1, j_2)$ comme la somme des deux distances spatiales séparant deux bits, avant et après permutation, dont les indices de lecture sont j_1 et j_2 :

$$S(j_1, j_2) = f(j_1, j_2) + f(\Pi(j_1), \Pi(j_2)) \tag{7.5}$$

où :

$$f(u, v) = \min\left\{|u - v|, k - |u - v|\right\} \tag{7.6}$$

[1] La permutation peut bien sûr être définie sous une forme réciproque, c'est-à-dire j fonction de i. C'est une convention à adopter une fois pour toutes et celle que nous avons retenue est compatible de la plupart des turbocodes normalisés.

La fonction f est introduite pour tenir compte du caractère circulaire des adresses. Finalement, nous appelons $S_{\min}$ la plus petite des valeurs de $S(j_1, j_2)$, pour toutes les paires j_1 et j_2 possibles :

$$S_{\min} = \min_{j_1, j_2} \{S(j_1, j_2)\} \tag{7.7}$$

Il est démontré dans [7.22] que la borne supérieure de $S_{\min}$ est :

$$\sup S_{\min} = \sqrt{2k} \tag{7.8}$$

Cette borne supérieure n'est atteinte que dans le cas d'une permutation régulière et avec les conditions :

$$P = P_0 = \sqrt{2k} \tag{7.9}$$

et :

$$k = \frac{P_0}{2} \bmod P_0 \tag{7.10}$$

Considérons maintenant une séquence de poids quelconque qui s'écrit, après permutation :

$$\tilde{d}(D) = \sum_{j=0}^{k-1} a_j D^j \tag{7.11}$$

où a_j peut prendre la valeur binaire 0 (pas d'erreur) ou 1 (une erreur) et, avant permutation :

$$d(D) = \sum_{i=0}^{k-1} a_i D^i = \sum_{j=0}^{k-1} a_{\Pi(j)} D^{\Pi(j)} \tag{7.12}$$

Notons $j_{\min}$ et $j_{\max}$ les indices j correspondant aux première et dernière valeurs a_j non nulles dans $\tilde{d}(D)$. Nous définissons de la même manière $i_{\min}$ et $i_{\max}$ pour la séquence $d(D)$. Alors, la permutation régulière satisfaisant (7.9) et (7.10) garantit la propriété :

$$(j_{\max} - j_{\min}) + (i_{\max} - i_{\min}) > \sqrt{2k} \tag{7.13}$$

Cela vient de ce que $d(D)$ et $\tilde{d}(D)$, toutes deux considérées entre les indices min et max, contiennent au moins 2 bits dont la distance spatiale cumulée, telle que définie par (7.5), est maximale et égale à $\sqrt{2k}$. Il nous faut maintenant considérer deux cas :

- les séquences $d(D)$ et $\tilde{d}(D)$ sont toutes deux du type *RTZ simple*, c'est-à-dire qu'elles débutent dans l'état 0 du codeur et y reviennent une seule fois, à la fin. Les bits de parité produits par ces séquences sont des 1, une fois sur deux statistiquement. Compte tenu de (7.13), pour des valeurs usuelles de k ($k > 100$), les poids de la redondance sont élevés et ces séquences *RTZ* ne contribuent pas à la DMH du turbocode.

- l'une au moins des séquences $d(D)$ et $\tilde{d}(D)$ est du type *RTZ multiple*, c'est-à-dire qu'elle correspond à plusieurs passages par l'état 0 du codeur. Si ces passages par l'état 0 sont longs, la parité associée à la séquence peut être de poids réduit et la distance associée faible. Généralement, dans ce type de situation, les séquences avant et après permutation sont toutes deux *RTZ* multiples.

La performance d'un turbocode, à faible taux d'erreurs, est intimement liée à la présence de motifs *RTZ* multiples et la permutation régulière n'est pas une bonne réponse pour éliminer ces motifs.

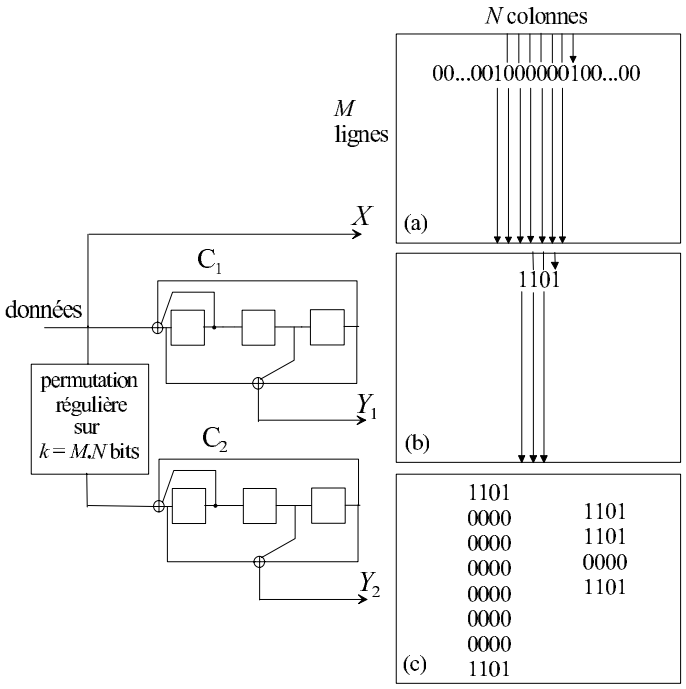


Figure 7.6 – Motifs d'erreurs possibles de poids 2, 3, 6 ou 9 avec un turbocode dont les codeurs élémentaires ont une période 7 et avec une permutation régulière.

Nécessité du désordre

En admettant toujours que les motifs d'erreurs qui ne sont pas *RTZ* ont des poids suffisamment élevés pour ne pas avoir d'incidence sur la performance, une permutation idéale pour un turbocode pourrait être définie par la règle suivante :

Si une séquence est du type *RTZ* avant permutation, alors elle ne l'est plus après permutation et vice-versa.

La règle précédente est impossible à satisfaire en pratique et un objectif plus réaliste est :

Si une séquence est du type *RTZ* avant permutation, alors elle ne l'est plus après permutation ou elle est devenue une séquence *RTZ* simple longue et vice-versa.

Le dilemme dans la conception de bonnes permutations pour turbocodes réside dans la nécessité de satisfaire cet objectif pour deux classes distinctes de séquences d'entrée qui demandent des traitements opposés : les séquences *RTZ* simples et les séquences *RTZ* multiples, telles que définies plus haut. Pour bien mettre ce problème en évidence, considérons un turbocode de rendement 1/3, avec une permutation rectangulaire régulière (écriture suivant M lignes, lecture suivant N colonnes) portant sur des blocs de $k = MN$ bits (figure 7.6). Les codeurs élémentaires sont des codeurs à 8 états dont la période est 7 (générateur de récursivité 15).

Le premier motif (a) de la figure 7.6 concerne une séquence d'erreurs possible de poids d'entrée $w = 2$: 10000001 pour le code C_1, qu'on appellera aussi code horizontal. Il s'agit de la séquence *RTZ* minimale de poids 2 pour le codeur considéré. La redondance produite par ce codeur est de poids 6 (exactement : 11001111). La redondance produite par le codeur vertical C_2, pour lequel la séquence considérée est aussi *RTZ* (sa longueur est multiple de 7), est autrement plus informative parce que *RTZ* simple et délivrée sur sept colonnes. En admettant que Y_2 est égal à 1 une fois sur deux en moyenne, le poids de cette redondance est environ $w(Y_2) \approx \frac{7M}{2}$, . Lorsqu'on fait tendre k vers l'infini à travers les valeurs de M et N ($M \approx N \approx \sqrt{k}$), la redondance produite par l'un des deux codes, pour ce type de motif, tend aussi vers l'infini. On dit alors que le code est *bon*.

Le second motif (b) est celui de la séquence *RTZ* minimale de poids d'entrée 3. Là aussi, la redondance est pauvre sur la première dimension et bien plus informative sur la seconde. Les conclusions sont les mêmes que précédemment.

Les deux autres dessins (c) représentent des exemples de séquences *RTZ* multiples, constituées de courtes séquences *RTZ* sur chacune des deux dimensions. Les poids d'entrée sont 6 et 9. Les distances associées à ces motifs (respectivement 30 et 27 pour ce code de rendement 1/3) ne sont généralement pas suffisantes pour assurer une bonne performance à faible taux d'erreurs. De plus, ces distances sont indépendantes de la taille du bloc et donc, vis-à-vis des motifs considérés, le code n'est pas *bon*.

La permutation régulière est donc une bonne permutation pour la classe des motifs d'erreurs *RTZ* simple. Pour les motifs *RTZ* multiples en revanche, la permutation régulière n'est pas appropriée. Une bonne permutation se doit de « casser » la régularité des motifs composites rectangulaires comme ceux de la figure 7.6(c), en introduisant un certain désordre. Mais cela ne doit pas se faire au détriment des motifs pour lesquels la permutation régulière est bonne. Le désordre doit donc être bien géré ! C'est là tout le problème dans la recherche d'une permutation qui doit conduire à une distance minimale suffisamment

élevée. Une bonne permutation ne peut être trouvée indépendamment des propriétés des codes élémentaires, de leurs motifs *RTZ*, de leurs périodicités, etc.

Le désordre intra-symbole

Lorsque les codes élémentaires sont des codes m-binaires, on peut introduire un certain désordre dans la permutation d'un turbocode sans pour autant lui enlever son caractère régulier ! Pour ce faire, on met en œuvre, outre une classique permutation inter-symbole, une permutation intra-symbole, c'est-à-dire une modification non-régulière du contenu des symboles de m bits, avant codage par le second code [7.23]. Nous développons brièvement cette idée sur l'exemple de turbocodes double-binaires ($m = 2$).

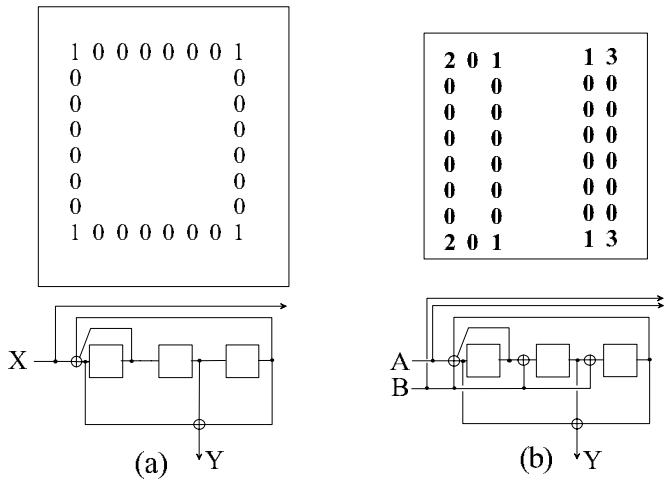


Figure 7.7 – Motifs d'erreurs possibles avec des turbocodes binaire (a) et double-binaire (b) et une permutation régulière.

La figure 7.7(a) représente le motif minimal d'erreurs de poids $w = 4$, toujours avec le code de la figure 7.6. C'est un motif carré dont le côté est égal à la période du générateur pseudo-aléatoire de polynôme 15, c'est-à-dire 7. Il a déjà été dit qu'un certain désordre devait être introduit dans la permutation pour « casser » ce genre de motif d'erreurs mais sans altérer les propriétés de la permutation régulière vis-à-vis des motifs de poids 2 et 3, ce qui n'est pas facile. Si l'on remplace, en tant que codeur élémentaire, le codeur binaire par un codeur double-binaire, les motifs d'erreurs à considérer sont formés, non plus par des bits mais par des couples de bits. La figure 7.7(b) donne un exemple de codeur double-binaire et de motifs d'erreurs possibles, lorsque la permutation est régulière. Les couples y sont numérotés de **0** à **3**, selon la correspondance suivante :

$$(0,0) : \mathbf{0}; \quad (0,1) : \mathbf{1}; \quad (1,0) : \mathbf{2}; \quad (1,1) : \mathbf{3}$$

Les périodicités du codeur double-binaire sont résumées par le diagramme de la figure 7.8. On y trouve toutes les combinaisons de paires de couples de type *RTZ*. Par exemple, si le codeur, initialisé dans l'état 0, est alimenté par les couples successifs **1** et **3**, celui-ci retrouve immédiatement l'état 0. Il en est de même pour les séquences **201** ou **2003** ou **3000001**, par exemple.

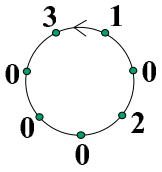


Figure 7.8 – Périodicités du codeur double-binaire de la figure 7.7(b). Les quatre couples d'entrée $(0,0)$, $(0,1)$, $(1,0)$ et $(1,1)$ sont notés 0, 1, 2 et 3, respectivement. Ce diagramme donne toutes les combinaisons de paires de couples de type RTZ.

La figure 7.7(b) donne deux exemples de motifs d'erreurs rectangulaires, de taille minimale. Observons tout d'abord que le périmètre de ces motifs est plus grand que la moitié du périmètre du carré de la figure 7.7(a). Or, à même rendement de codage, la redondance d'un code double-binaire est deux fois plus dense que celle d'un code binaire. On en déduit que les distances des motifs d'erreurs double-binaires seront naturellement plus grandes, toutes choses égales par ailleurs, que celles des motifs d'erreurs binaires. De plus, par un moyen simple, on peut éliminer ces motifs élémentaires.

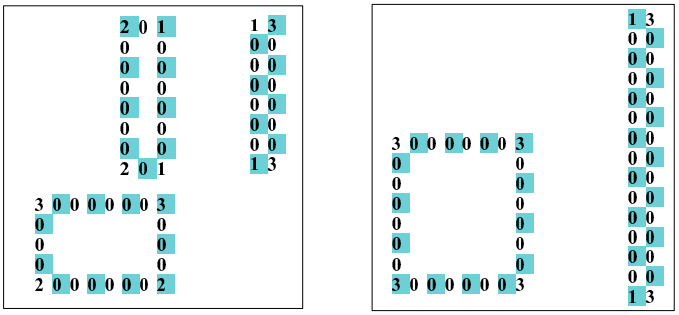


Figure 7.9 – Les couples des cases grises sont inversés avant le second codage (vertical). 1 devient 2, 2 devient 1 ; 0 et 3 restent inchangés. Les motifs de la figure 7.7(b), redessinés en (a), ne sont plus des motifs d'erreurs possibles. Ceux de (b) le sont toujours, avec des distances 24 et 26 pour un rendement de codage 1/2.

Supposons par exemple que les couples soient inversés (**1** devient **2** et réciproquement), une fois sur deux, avant d'être appliqués au codeur vertical. Alors les motifs d'erreurs représentés en figure 7.9(a) n'existent plus ; par exemple, si **30002** représente bien une séquence *RTZ* pour le codeur considéré, **30001**

ne l'est plus. Ainsi, beaucoup des motifs d'erreurs, en particulier les plus petits, disparaissent grâce au désordre introduit à l'intérieur des symboles. La figure 7.9(b) donne deux exemples de motifs que l'inversion périodique ne modifie pas. Les distances correspondantes sont suffisamment élevées (24 et 26 pour une rendement 1/2) pour ne pas poser problème pour des tailles de blocs petites ou moyennes. Pour des blocs longs (plusieurs milliers de bits), un désordre supplémentaire inter-symbole, de faible intensité, peut être ajouté à la non-uniformité intra-symbole, pour obtenir des distances minimales encore plus élevées.

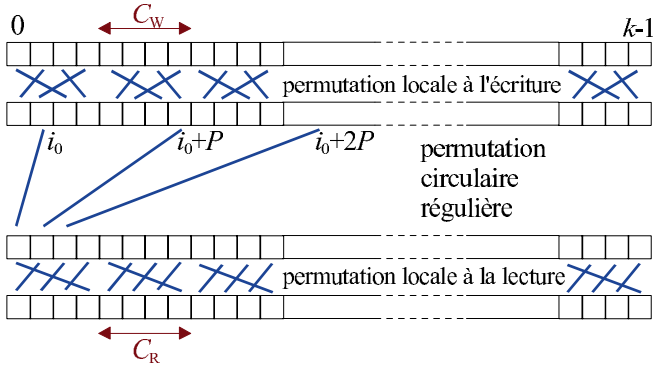


Figure 7.10 – Permutation de type *DRP*. C'est une permutation circulaire régulière à laquelle des permutations locales avant écriture et après lecture sont ajoutées.

Permutations irrégulières

Nous ne ferons pas, dans cette section, une description de toutes les permutations irrégulières qui ont pu être imaginées jusqu'à ce jour et qui ont fait l'objet de nombreuses publications ou de plusieurs chapitres d'ouvrages (voir [7.24, 7.25] par exemple). Nous avons plutôt retenu de présenter ce qui semble être, pour le moment, le type de permutation à la fois le plus simple et le plus performant. Il s'agit de permutations circulaires presque régulières, appelées*ARP* (*almost regular permutation* , [7.26]) ou *DRP* (*dithered relatively prime*, [7.27]) suivant les auteurs. Dans tous les cas, l'idée est de ne pas trop s'éloigner de la permutation régulière, bien adaptée aux motifs d'erreurs *RTZ* simples et d'instiller un petit désordre contrôlé pour contrer les motifs d'erreurs *RTZ* multiples.

La figure 7.10 donne un exemple, tiré de [7.27], de ce que peut être ce petit désordre. Avant que soit effectuée la permutation circulaire régulière, les bits subissent une permutation locale. Cette permutation s'effectue par groupes de C_W bits. C_W, qui est le cycle du désordre à l'écriture (*writing cycle*), est un diviseur de la longueur k du message. De même, une permutation locale de cycle C_R (*reading cycle*) est appliquée avant la lecture définitive.

En pratique C_W et C_R peuvent être de valeurs identiques $C_W = C_R = C$, typiquement, 4 ou 8. Cette manière d'introduire du désordre, par de petites fluctuations locales, ne diminue pas significativement la distance spatiale cumulée, dont la valeur maximale est $\sqrt{2k}$. Elle permet en retour de supprimer les motifs d'erreurs comparables à ceux des figures 7.5(c) et 7.6(b) à la condition que les hauteurs et largeurs de ces motifs ne soient pas toutes deux multiples de C.

Une autre manière de perturber la permutation régulière de façon contrôlée est décrite par la figure 7.11. La permutation est représentée ici sous sa forme rectangulaire, visuellement plus accessible, mais elle s'applique très bien aussi à la permutation circulaire. Une information (bit ou symbole) est positionnée à chaque croisement de ligne et de colonne. Avec la permutation régulière, ces données sont donc mémorisées ligne par ligne et lues colonne par colonne. Dans la figure 7.11, le désordre est introduit par l'intermédiaire de quatre vecteurs de déplacement $V_1, \cdots, V_4$ qui sont alternativement appliqués lors de la lecture. Ces vecteurs sont de petite amplitude par rapport aux dimensions de la matrice de permutation.

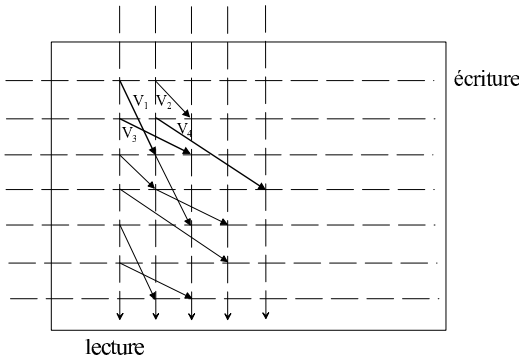


Figure 7.11 – Permutation de type *ARP*, d'après [7.26].

Le modèle mathématique associé à cette permutation presque régulière, sous sa forme circulaire, est une extension de (7.4) :

$$i = \Pi(j) \equiv Pj \ + Q(j) + i_0 \bmod k \tag{7.14}$$

Si l'on choisit :

$$Q(j) = A(j)P + B(j) \tag{7.15}$$

où les entiers positifs $A(j)$ et $B(j)$ sont périodiques, de cycle C (diviseur de k), alors ces grandeurs correspondent à des décalages positifs appliqués respectivement avant et après la permutation régulière. C'est la différence avec la permutation décrite par la figure 7.10, dans laquelle les perturbations en écriture et en lecture se réalisent à l'intérieur de petits groupes de données et non par décalage.

Pour que la permutation soit bien une bijection, les paramètres $A(j)$ et $B(j)$ ne sont pas quelconques. Une condition suffisante pour assurer l'existence de la permutation est que les paramètres soient tous multiples de C. Cette condition n'est pas très contraignante vis-à-vis de l'efficacité de la permutation. (7.15) peut alors être récrit sous la forme :

$$Q(j) = C(\alpha(j)P + \beta(j)) \tag{7.16}$$

où $\alpha(j)$ et $\beta(j)$ sont le plus souvent de petits entiers, de valeurs 0 à 8. Par ailleurs, puisque les propriétés d'une permutation circulaire ne sont pas modifiées par une simple rotation, l'une des valeurs $Q(j)$ peut être systématiquement 0.

Deux jeux typiques de valeurs Q, avec un cycle 4 et $\alpha = 0$ ou 1, sont donnés ci-dessous :

$$\begin{array}{ll} \text{si } j = 0 \mod 4, & \text{alors } Q = 0 \\ \text{si } j = 1 \mod 4, & \text{alors } Q = 4P + 4\beta_1 \\ \text{si } j = 2 \mod 4, & \text{alors } Q = 4\beta_2 \\ \text{si } j = 3 \mod 4, & \text{alors } Q = 4P + 4\beta_3 \end{array} \tag{7.17}$$

$$\begin{array}{ll} \text{si } j = 0 \mod 4, & \text{alors } Q = 0 \\ \text{si } j = 1 \mod 4, & \text{alors } Q = 4\beta_1 \\ \text{si } j = 2 \mod 4, & \text{alors } Q = 4P + 4\beta_2 \\ \text{si } j = 3 \mod 4, & \text{alors } Q = 4P + 4\beta_3 \end{array} \tag{7.18}$$

Ces modèles requièrent la connaissance de seulement quatre paramètres (P, β_1, β_2 et β_3), qui peuvent être déterminés selon la procédure détaillée dans [7.26]. L'utilisation de codes m-binaires (voir section 7.5), au lieu de codes binaires, demande simplement de remplacer k par k/m dans (7.14). En particulier, les permutations définies pour les turbocodes double-binaires ($m = 2$) des normes DVB-RCS, DVB-RCT et WiMax sont inspirées de (7.14) et (7.19).

7.4 Le décodage des turbocodes

7.4.1 Le turbodécodage

Le décodage d'un turbocode binaire s'appuie sur le schéma de principe de la figure 7.12. La boucle permet à chaque décodeur de tirer profit de l'ensemble des informations disponibles. Les grandeurs considérées à chaque noeud du montage sont des LRV (Logarithmes de Rapport de Vraisemblance), les opérations de décodage étant effectuées dans le domaine logarithmique.

Le LRV en sortie d'un décodeur de code systématique peut être vu comme la somme de deux termes : l'information intrinsèque, en provenance du canal de transmission, et l'information extrinsèque, que ce décodeur ajoute à la première pour effectuer son travail de correction. Comme l'information intrinsèque est utilisée par les deux décodeurs (à des instants différents), c'est l'information extrinsèque produite par chacun des décodeurs qui doit être transmise à

l'autre en tant qu'information nouvelle pour assurer la convergence conjointe. La section 7.4.2 détaille les opérations effectuées pour le calcul de l'information extrinsèque, par mise en œuvre de l'algorithme *MAP* ou de sa version simplifiée Max-Log-*MAP*.

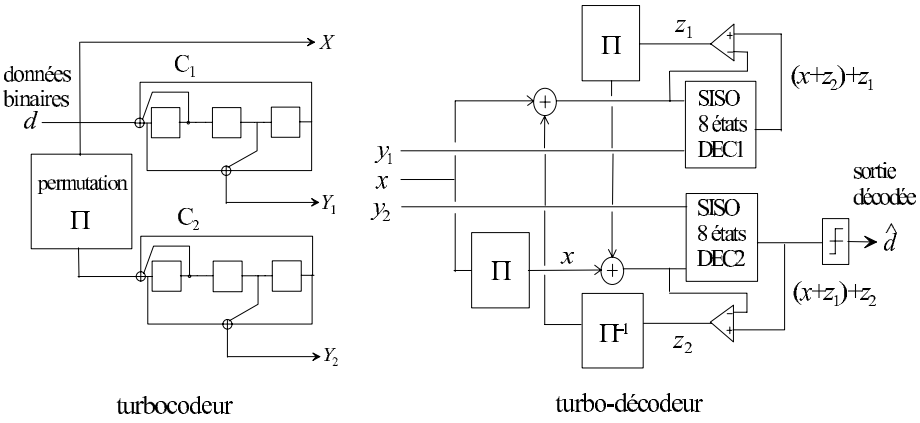


Figure 7.12 – Turbocodeur à 8 états et structure de principe du turbo-décodeur correspondant. Les deux décodeurs élémentaires s'échangent des informations probabilistes, dites extrinsèques (z)

.

À cause des effets de latence, l'échange des informations extrinsèques, dans un circuit de traitement numérique, doit être mis en œuvre à travers un processus itératif : premier décodage par DEC_1 et mise en mémoire des informations extrinsèques z_1, deuxième décodage par DEC_2 et mise en mémoire des informations extrinsèques z_2 (fin de la première itération), de nouveau appel à DEC_1 et mise en mémoire de z_1, etc. Différentes architectures matérielles, avec des degrés plus ou moins élevés de parallélisme, sont envisageables pour accélérer le décodage itératif.

Si l'on voulait décoder le turbocode à l'aide d'un seul décodeur, qui prendrait en compte tous les états possibles du codeur, on obtiendrait pour chaque élément du message décodé une probabilité et une seule, d'avoir une valeur binaire égale à 0 ou à 1. La structure composite de la figure 7.12, quant à elle, fait appel à deux décodeurs travaillant conjointement. Par analogie avec le résultat que fournirait le décodeur unique, il leur faut donc *converger vers les mêmes décisions, avec les mêmes probabilités*, pour chacune des données considérées. C'est le principe fondamental du traitement « turbo », qui justifie la structure du décodeur, comme le montre le raisonnement suivant.

Le rôle d'un décodeur *SISO* (*Soft-In/Soft-Out*, voir section 7.4.2) est de traiter les LRV à son entrée pour essayer de les rendre plus fiables, grâce à la redondance locale (c'est-à-dire y_1 pour DEC1, y_2 pour DEC2). Le LRV produit par un décodeur de code binaire, relatif à la donnée d, peut s'écrire simplement

comme :

$$\mathrm{LRV}_{\mathrm{sortie}}(d) = \mathrm{LRV}_{entre}(d) + z(d) \tag{7.19}$$

où $z(d)$ est l'information extrinsèque propre à d. Le LRV est amélioré quand z est négative et d est un 0, ou quand z est positive et d est un 1.

Après p itérations, la sortie de DEC1 est :

$$\mathrm{LLR}^p_{\mathrm{sortie},1}(d) = (x + z_2^{p-1}(d)) + z_1^p(d)$$

et celle de DEC2 est :

$$\mathrm{LLR}^p_{\mathrm{sortie},2}(d) = (x + z_1^{p-1}(d)) + z_2^p(d)$$

Si le processus itératif converge vers une solution stable, $z_1^p(d) - z_1^{p-1}(d)$ et $z_2^p(d) - z_2^{p-1}(d)$ tendent vers zéro lorsque p tend vers l'infini. En conséquence, les deux LRV relatifs à d deviennent identiques, satisfaisant ainsi le critère fondamental de commune probabilité énoncé plus haut. Quant à la preuve de la convergence, elle fait toujours l'objet de travaux approfondis et on peut consulter à ce propos, par exemple [7.28, 7.29].

Outre les fonctions de permutation et de permutation inverse, la figure 7.13 détaille les opérations effectuées lors du turbo-décodage :

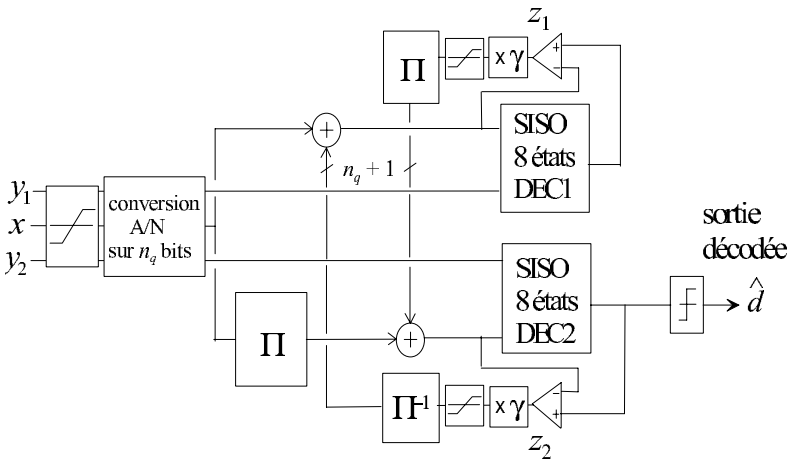


Figure 7.13 – Opérations détaillées (écrêtage, quantification, atténuation de l'information extrinsèque) dans le turbo-décodeur de la figure 7.12.

1. La conversion analogique-numérique (A/N) transforme les informations en provenance du démodulateur en échantillons exploitables par le décodeur numérique. Deux paramètres interviennent dans cette opération : n_q, le nombre de bits de quantification, et Q, le facteur d'échelle, c'est-à-dire le rapport entre la valeur absolue moyenne du signal quantifié et sa valeur absolue maximale. n_q est fixé à une valeur de compromis entre

la précision requise et la complexité du décodeur. Avec $n_q = 4$, la performance du décodeur est très proche de celle que l'on obtient avec des échantillons réels. La valeur de Q dépend de la modulation, du rendement de codage et du type de canal. Elle est par exemple plus grande pour un canal de Rayleigh que pour un canal gaussien.

2. Le décodage *SISO* a pour rôle d'augmenter le rapport signal bruit équivalent du LRV, c'est-à-dire de fournir à la sortie une information extrinsèque z_{sortie} plus fiable qu'à l'entrée (z_{entre}). De la fonction de transfert $\mathrm{SNR}(z_{sortie}) = G(\mathrm{SNR}(z_{entre}))$ de chacun des décodeurs dépendra la convergence du procédé itératif (voir section 7.6).

   Lorsqu'une information n'est pas disponible à l'entrée du décodeur *SISO*, du fait du poinçonnage par exemple, une valeur neutre (zéro analogique) se substitue à cette information manquante.

3. Lorsque l'algorithme élémentaire de décodage n'est pas l'algorithme optimal (*MAP*) mais une version simplifiée sous-optimale, l'information extrinsèque doit subir quelques transformations avant d'être utilisée par un décodeur :
   - la multiplication de l'information extrinsèque par le facteur γ, inférieur à 1, garantit la stabilité de la structure bouclée. γ peut varier au fil des itérations, par exemple de 0,7 au début du processus itératif jusqu'à 1 pour la dernière itération.
   - l'écrêtage de l'information extrinsèque répond à la fois au souci de limiter la taille des mémoires et au souci de participer à la stabilité du processus. Une valeur typique de dynamique maximale de l'information extrinsèque est de 2 fois la dynamique d'entrée du décodeur.

4. La prise de décision binaire s'effectue par un seuillage à la valeur 0. Le nombre d'itérations requis par le turbo-décodage dépend de la taille du bloc et du rendement de codage. En général, la DMH du code est d'autant plus élevée que le bloc décodé est grand et la convergence en est plus lente. Il en est de même lorsque les rendements de codage sont faibles. En pratique, on limite le nombre d'itérations à une valeur comprise entre 4 et 10, selon les contraintes de vitesse, de latence et de consommation imposées par l'application.

La figure 7.14 donne un exemple de performance de turbocode binaire, tiré de la norme UMTS [7.15]. On peut y observer une décroissance du Taux d'Erreurs par Paquets (TEP), tout près de la limite théorique (qui est environ 0,5 dB, en tenant compte de la taille du bloc), mais aussi un changement de pente assez prononcé, dû à une DMH qui n'est pas extraordinaire ($d_{\min} = 26$) pour un rendement de 1/3.

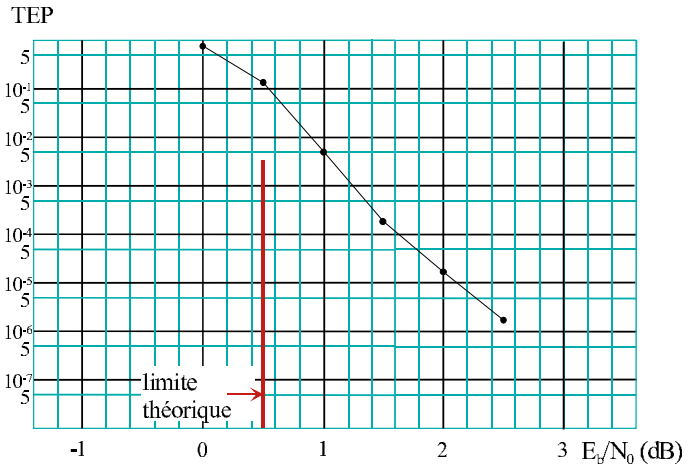


Figure 7.14 – Performance en Taux d'Erreurs de Paquets (TEP) du turbocode de la norme UMTS pour $k = 640$ et $R = 1/3$ sur canal gaussien avec modulation MDP-4. Décodage selon l'algorithme Max-Log-*MAP* avec 6 itérations.

7.4.2 Le décodage *SISO* et l'information extrinsèque

Sont ici développés les traitements effectués en pratique dans un décodeur *SISO* utilisant l'algorithme *MAP* [7.30] ou sa version simplifiée Max-log-*MAP*, encore appelée *SubMAP* [7.31], pour décoder les codes CSR m-binaires et mettre en œuvre le décodage itératif. Pour les codes et turbocodes binaires, toutes ces équations sont à simplifier en prenant $m = 1$.

Notations

Une séquence de données $\mathbf{d}$ est définie par $\mathbf{d} \equiv \mathbf{d}_0^{k-1} = (\mathbf{d}_0 \cdots \mathbf{d}_i \cdots \mathbf{d}_{k-1})$, où $\mathbf{d}_i$ est le vecteur de données m-binaires appliqué à l'entrée du codeur à l'instant i : $\mathbf{d}_i = (d_{i,1} \cdots d_{i,l} \cdots d_{i,m})$. La valeur de $\mathbf{d}_i$ pourra également être représentée par la grandeur scalaire entière $j = \sum_{l=1}^{m} 2^{l-1} d_{i,l}$, comprise entre 0 et $2^m - 1$ et l'on écrira alors $\mathbf{d}_i \equiv j$.

Dans le cas d'une modulation à deux ou quatre états de phase (MDP-2, MDP-4), la séquence codée et modulée $\mathbf{u} \equiv \mathbf{u}_0^{k-1} = (\mathbf{u}_0 \cdots \mathbf{u}_i \cdots \mathbf{u}_{k-1})$ est constituée de vecteurs $\mathbf{u}_i$ de taille $m + m'$: $\mathbf{u}_i = (u_{i,1} \cdots u_{i,l} \cdots u_{i,m+m'})$, où $u_{i,l} = \pm 1$ pour $l = 1 \cdots m + m'$ et m' est le nombre de bits de redondance ajoutés aux m bits d'information. Le symbole $u_{i,l}$ est donc représentatif d'un bit systématique pour $l \leq m$ et d'un bit de redondance pour $l > m$.

La séquence observée en sortie du démodulateur est notée $\mathbf{v} \equiv \mathbf{v}_0^{k-1} = (\mathbf{v}_0 \cdots \mathbf{v}_i \cdots \mathbf{v}_{k-1})$, avec $\mathbf{v}_i = (v_{i,1} \cdots v_{i,l} \cdots v_{i,m+m'})$. La suite des états du codeur entre les instants 0 et k est notée $\mathbf{S} = \mathbf{S}_0^k = (\mathbf{S}_0 \cdots \mathbf{S}_i \cdots \mathbf{S}_k)$. Ce qui suit s'appuie sur les résultats présentés dans le chapitre sur les codes convolutifs.

Décodage suivant le critère du *Maximum A Posteriori* (*MAP*)

À chaque instant i, les estimations pondérées (probabilistes) fournies par le décodeur *MAP* sont les 2^m probabilités *a posteriori* (*APP* pour *A Posteriori Probabilities*) $\Pr(\mathbf{d}_i \equiv j \mid \mathbf{v})$, $j = 0 \cdots 2^m - 1$. La décision dure correspondante, $\hat{\mathbf{d}}_i$, est la représentation binaire de la valeur j qui maximise l'*APP*.

Chaque *APP* peut s'exprimer en fonction des vraisemblances conjointes $p(\mathbf{d}_i \equiv j, \mathbf{v})$:

$$\Pr(\mathbf{d}_i \equiv j \mid \mathbf{v}) = \frac{p(\mathbf{d}_i \equiv j, \mathbf{v})}{p(\mathbf{v})} = \frac{p(\mathbf{d}_i \equiv j, \mathbf{v})}{\sum_{l=0}^{2^m-1} p(\mathbf{d}_i \equiv l, \mathbf{v})} \tag{7.20}$$

En pratique, on calcule les vraisemblances conjointes $p(\mathbf{d}_i \equiv j, \mathbf{v})$ pour $j = 0 \cdots 2^m - 1$ puis chaque *APP* est obtenue par normalisation.

Le treillis représentatif d'un code de mémoire ν possède 2^ν états, prenant leur valeur scalaire s dans $(0, 2^\nu - 1)$. Les vraisemblances conjointes sont calculées à partir des probabilités récurrentes avant (*forward*) $\alpha_i(s)$ et arrière (*backward*) $\beta_i(s)$ et des vraisemblances de branches $g_i(s', s)$:

$$p(\mathbf{d}_i \equiv j, \mathbf{v}) = \sum_{(s',s)/\mathbf{d}_i(s',s) \equiv j} \beta_{i+1}(s)\, \alpha_i(s')\, g_i(s', s) \tag{7.21}$$

où $(s', s)/\mathbf{d}_i(s', s) \equiv j$ désigne l'ensemble des transitions d'état à état $s' \to s$ associées à la donnée d'information m-binaire j. Cet ensemble est, bien sûr, toujours le même dans un treillis invariant dans le temps.

La grandeur $g_i(s', s)$ s'exprime comme :

$$g_i(s', s) = \Pr{}^a(\mathbf{d}_i \equiv j, \mathbf{d}_i(s', s) \equiv j).p(\mathbf{v}_i \mid \mathbf{u}_i) \tag{7.22}$$

où $\mathbf{u}_i$ est le jeu de symboles d'information systématique et redondante associé à la transition $s' \to s$ du treillis à l'instant i et $\Pr^a(\mathbf{d}_i \equiv j, \mathbf{d}_i(s', s) \equiv j)$ est la probabilité *a priori* d'émettre le m-uplet d'information et que cela corresponde à la transition $s' \to s$ à l'instant i. Si la transition $s' \to s$ n'existe pas dans le treillis pour $\mathbf{d}_i \equiv j$, alors $\Pr^a(\mathbf{d}_i \equiv j, \mathbf{d}_i(s', s) \equiv j) = 0$, sinon celle-ci est donnée par la statistique de la source (le plus souvent uniforme en pratique).

Dans le cas d'un canal gaussien à entrées binaires, la grandeur $p(\mathbf{v}_i \mid \mathbf{u}_i)$ s'écrit :

$$p(\mathbf{v}_i \mid \mathbf{u}_i) = \prod_{l=1}^{m+m'} \left(\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(v_{i,l} - u_{i,l})^2}{2\sigma^2} \right) \right) \tag{7.23}$$

où σ^2 est la variance du bruit additif blanc gaussien. En pratique, on ne retient que les termes spécifiques à la transition considérée et qui ne s'éliminent pas par division dans l'expression (7.20) :

$$p'(\mathbf{v}_i \mid \mathbf{u}_i) = \exp\left(\frac{\sum_{l=1}^{m+m'} v_{i,l} \cdot u_{i,l}}{\sigma^2} \right) \tag{7.24}$$

Les probabilités récurrentes avant et arrière sont calculées de la manière suivante :

$$\alpha_i(s) = \sum_{s'=0}^{2^\nu - 1} \alpha_{i-1}(s')\, g_{i-1}(s', s) \quad \text{pour } i = 1 \cdots k \tag{7.25}$$

et :

$$\beta_i(s) = \sum_{s'=0}^{2^\nu - 1} \beta_{i+1}(s')\, g_i(s, s') \quad \text{pour } i = k-1 \cdots 0 \tag{7.26}$$

Pour éviter les problèmes de précision ou de débordement dans la représentation de ces grandeurs, il convient de les normaliser régulièrement. L'initialisation des récursions dépend de la connaissance ou non de l'état du codeur en début et en fin de codage. Si l'état $\mathbf{S}_0$ de départ du codeur est connu, alors $\alpha_0(\mathbf{S}_0) = 1$ et $\alpha_0(s) = 0$ pour tout autre état, sinon tous les $\alpha_0(s)$ sont initialisés à la même valeur. La même règle est appliquée pour l'état final $\mathbf{S}_k$. Pour les codes circulaires, l'initialisation est effectuée automatiquement après l'étape de prologue, qui démarre à partir de valeurs identiques pour tous les états du treillis.

Dans le contexte du décodage itératif, le décodeur composite fait appel à deux décodeurs élémentaires s'échangeant des *probabilités extrinsèques*. Par conséquent, la brique de base de décodage décrite précédemment doit être reconsidérée :

1. pour tenir compte dans l'expression (7.22) d'une probabilité extrinsèque, $\Pr^{ex}(\mathbf{d}_i \equiv j \,\big|\, \mathbf{v}')$, calculée par l'autre décodeur élémentaire du décodeur composite, à partir de sa propre séquence d'entrée $\mathbf{v}'$,
2. pour produire sa propre probabilité extrinsèque $\Pr^{ex}(\mathbf{d}_i \equiv j \,\big|\, \mathbf{v}')$ qui sera utilisée par l'autre décodeur élémentaire.

En pratique, pour chaque valeur de j, $j = 0 \cdots 2^m - 1$:

1. dans l'expression (7.22), la probabilité *a priori* $\Pr^a(\mathbf{d}_i \equiv j, \mathbf{d}_i(s', s) \equiv j)$ est remplacée par la probabilité *a priori* modifiée $\Pr^{@}(\mathbf{d}_i \equiv j, \mathbf{d}_i(s', s) \equiv j)$, ayant pour expression, à un facteur de normalisation près :

$$\Pr^{@}(\mathbf{d}_i \equiv j, \mathbf{d}_i(s', s) \equiv j) = \Pr^a(\mathbf{d}_i \equiv j, \mathbf{d}_i(s', s) \equiv j).\Pr^{ex}(\mathbf{d}_i \equiv j \mid \mathbf{v}') \tag{7.27}$$

1. $\Pr^{ex}(\mathbf{d}_i \equiv j \mid \mathbf{v})$ est donnée par :

$$\Pr^{ex}(\mathbf{d}_i \equiv j \mid \mathbf{v}) = \frac{\sum\limits_{(s',s)/\mathbf{d}_i(s',s)\equiv j} \beta_{i+1}(s)\, \alpha_i(s')\, g_i^*(s', s)}{\sum\limits_{(s',s)} \beta_{i+1}(s)\, \alpha_i(s')\, g_i^*(s', s)} \tag{7.28}$$

Les termes $g_i^*(s', s)$ sont non nuls si $s' \to s$ correspond à une transition du treillis et sont alors déduits de l'expression de $p(\mathbf{v}_i \mid \mathbf{u}_i)$ en éliminant la partie systématique de l'information. Dans le cas d'une transmission sur un canal

gaussien à entrées binaires et en partant de l'expression simplifiée (7.24) de $p'(\mathbf{v}_i \mid \mathbf{u}_i)$, on a :

$$g_i^*(s',s) = \exp\left(\frac{\sum_{l=m+1}^{m+m'} v_{i,l}u_{i,l}}{\sigma^2}\right) \tag{7.29}$$

L'algorithme simplifié Max-Log-*MAP* ou *SubMAP*

Le décodage suivant le critère *MAP* requiert un grand nombre d'opérations, dont des calculs d'exponentielles et des multiplications. La réécriture de l'algorithme de décodage dans le domaine logarithmique simplifie les traitements. Les estimations pondérées fournies par le décodeur sont alors des grandeurs proportionnelles aux logarithmes des *APP*, dites *Log-APP* et notées L :

$$L_i(j) = -\frac{\sigma^2}{2}\ln \Pr(\mathbf{d}_i \equiv j \mid \mathbf{v}) \ , \quad j = 0 \cdots 2^m - 1 \tag{7.30}$$

On définit $M_i^{\alpha}(s)$ et $M_i^{\beta}(s)$, les métriques avant et arrière relatives au nœud s à l'instant i ainsi que $M_i(s',s)$, la métrique de branche relative à la transition $s' \rightarrow s$ du treillis à l'instant i par :

$$\begin{array}{l} M_i^{\alpha}(s) = -\sigma^2 \ln \alpha_i(s) \\ M_i^{\beta}(s) = -\sigma^2 \ln \beta_i(s) \\ M_i(s',s) = -\sigma^2 \ln g_i(s',s) \end{array} \tag{7.31}$$

Introduisons les grandeurs $A_i(j)$ et B_i calculées comme :

$$A_i(j) = -\sigma^2 \ln \left[\sum_{(s',s)/\mathbf{d}_i(s',s)\equiv j} \beta_{i+1}(s)\alpha_i(s')g_i(s',s) \right] \tag{7.32}$$

$$B_i = -\sigma^2 \ln \left[\sum_{(s',s)} \beta_{i+1}(s)\alpha_i(s')g_i(s',s) \right] \tag{7.33}$$

$L_i(j)$ peut alors s'écrire, par référence à (7.20) et (7.21), de la manière suivante :

$$L_i(j) = \frac{1}{2}\left(A_i(j) - B_i\right) \tag{7.34}$$

Les expressions (7.32) et (7.33) peuvent être simplifiées en appliquant l'approximation dite *Max-Log* :

$$\ln(\exp(a) + \exp(b)) \approx \max(a,b) \tag{7.35}$$

On obtient pour $A_i(j)$:

$$A_i(j) \approx \min_{(s',s)/\mathbf{d}_i(s',s)\equiv j} \left(M_{i+1}^{\beta}(s) + M_i^{\alpha}(s') + M_i(s',s) \right) \tag{7.36}$$

et pour B_i :

$$B_i \approx \min_{(s',s)} \left(M_{i+1}^{\beta}(s) + M_i^{\alpha}(s') + M_i(s',s) \right) = \min_{l=0\cdots 2^m-1} A_i(l) \tag{7.37}$$

et l'on a finalement :

$$L_i(j) = \frac{1}{2} \left(A_i(j) - \min_{l=0\cdots 2^m-1} A_i(l) \right) \tag{7.38}$$

Remarquons que ces grandeurs sont toujours positives ou nulles.

Introduisons les grandeurs L^a proportionnelles aux logarithmes des probabilités *a priori* $\Pr^a$:

$$L_i^a(j) = -\frac{\sigma^2}{2} \ln \Pr{}^a(\mathbf{d}_i \equiv j) \tag{7.39}$$

Les métriques de branches $M_i(s',s)$ s'écrivent, d'après (7.22) et (7.31) :

$$M_i(s',s) = 2L_i^a(\mathbf{d}(s',s)) - \sigma^2 \ln p(\mathbf{v}_i \mid \mathbf{u}_i) \tag{7.40}$$

Si la statistique d'émission *a priori* des m-uplets $\mathbf{d}_i$ est uniforme, le terme $2L_i^a(\mathbf{d}(s',s))$ peut être omis dans la relation précédente car c'est la même valeur qui intervient dans toutes les métriques de branche. Dans le cas d'une transmission sur un canal gaussien à entrées binaires, on a d'après (7.24) :

$$M_i(s',s) = 2L_i^a(\mathbf{d}(s',s)) - \sum_{l=1}^{m+m'} v_{i,l} \cdot u_{i,l} \tag{7.41}$$

Les métriques avant et arrière sont alors calculées à partir des relations de récurrence suivantes :

$$M_i^{\alpha}(s) = \min_{s'=0,\cdots,2^{\nu}-1} \left(M_{i-1}^{\alpha}(s') - \sum_{l=1}^{m+m'} v_{i-1,l} \cdot u_{i-1,l} + 2L_{i-1}^a(\mathbf{d}(s',s)) \right) \tag{7.42}$$

$$M_i^{\beta}(s) = \min_{s'=0,\cdots,2^{\nu}-1} \left(M_{i+1}^{\beta}(s') - \sum_{l=1}^{m+m'} v_{i,l} \cdot u_{i,l} + 2L_i^a(\mathbf{d}(s,s') \right) \tag{7.43}$$

L'application de l'algorithme Max-Log-*MAP* revient en fait à réaliser un double décodage de Viterbi, dans les sens aller et retour. Il est, pour cette raison, également appelé algorithme *dual Viterbi*.

Si l'état de départ du codeur, $\mathbf{S}_0$, est connu, alors $M_0^{\alpha}(\mathbf{S}_0) = 0$ et $M_0^{\alpha}(s) = +\infty$ pour tout autre état, sinon tous les $M_0^{\alpha}(s)$ sont initialisés à la même valeur. La même règle est appliquée pour l'état final. Pour les codes circulaires, toutes les métriques sont initialisées à la même valeur en début de prologue.

Finalement, en tenant compte de (7.36) et en remplaçant $M_i(s',s)$ par son expression (7.41), on obtient :

$$A_i(j) = \min_{(s',s)/\mathbf{d}_i(s',s)\equiv j} \left(M_{i+1}^{\beta}(s) + M_i^{\alpha}(s') - \sum_{l=1}^{m+m'} v_{i,l} \cdot u_{i,l} \right) + 2L_i^a(j) \tag{7.44}$$

La décision dure prise par le décodeur est la valeur de j, $j = 0 \cdots 2^m - 1$, qui minimise $A_i(j)$. Notons j_0 cette valeur. D'après (7.38), $L_i(j)$ s'écrit :

$$L_i(j) = \frac{1}{2}\left[A_i(j) - A_i(j_0)\right] \text{ pour } j = 0 \cdots 2^m - 1 \tag{7.45}$$

On notera que la présence du coefficient σ^2 dans la définition (7.30) de $L_i(j)$ permet de s'affranchir de la connaissance de ce paramètre pour le calcul des métriques et par conséquent pour tout le décodage. C'est un avantage important de la méthode Max-Log-*MAP* sur la méthode *MAP*.

Dans le contexte du décodage itératif, le terme $L_i^a(j)$ est modifié afin de prendre en compte l'information extrinsèque $L_i^*(j)$ en provenance de l'autre décodeur élémentaire :

$$L_i^{@}(j) = L_i^a(j) + L_i^*(j) \tag{7.46}$$

D'autre part, l'information extrinsèque produite par le décodeur est obtenue en éliminant dans $L_i(j)$ les termes contenant l'information directe sur $\mathbf{d}_i$, c'est-à-dire les informations intrinsèques et *a priori* :

$$\begin{aligned} L_i^*(j) = \tfrac{1}{2}\Bigg[& \min_{(s',s)/\mathbf{d}_i(s',s)\equiv j}\left(M_{i+1}^{\beta}(s) + M_i^{\alpha}(s') - \sum_{l=m+1}^{m+m'} v_{i,l}\cdot u_{i,l}\right) \\ & - \min_{(s',s)/\mathbf{d}_i(s',s)\equiv j_0}\left(M_{i+1}^{\beta}(s) + M_i^{\alpha}(s') - \sum_{l=m+1}^{m+m'} v_{i,l}\cdot u_{i,l}\right)\Bigg] \end{aligned} \tag{7.47}$$

L'expression de $L_i(j)$ peut alors être formulée comme suit :

$$L_i(j) = L_i^*(j) + \frac{1}{2}\sum_{l=1}^{m} v_{i,l}\cdot\left[u_{i,l}|_{\mathbf{d}_i\equiv j} - u_{i,l}|_{\mathbf{d}_i\equiv j_0}\right] + \left[L_i^{@}(j) - L_i^{@}(j_0)\right] \tag{7.48}$$

Cette expression montre que l'information extrinsèque $L_i^*(j)$ peut, en pratique, être déduite de $L_i(j)$ par simple soustraction. Le facteur $\frac{1}{2}$ dans la définition (7.30) de $L_i(j)$ permet d'obtenir une décision pondérée et une information extrinsèque $L_i^*(j)$ à la même échelle que les échantillons bruités $v_{i,l}$.

7.4.3 Considérations pratiques

La manière la plus simple de réaliser le turbo-décodage est complètement séquentielle et fait appel aux opérations suivantes, ici fondées sur l'algorithme Max-Log-*MAP* et répétées autant de fois que nécessaire :

1. Récursion retour pour le code C_2 (figure 7.12), calcul et mémorisation des métriques $M_i^{\beta}(s)$, $i = k-1, ..., 0$ et $s = 0, ..., 2^{\nu}-1$,
2. Récursion avant pour le code C_2, calcul des métriques $M_i^{\alpha}(s)$, $i = 0, ..., k-1$ et $s = 0, ..., 2^{\nu}-1$. Calcul et mémorisation des informations extrinsèques,

3. Récursion retour pour le code C_1, calcul et mémorisation des métriques $M_i^{\beta}(s)$, $i = k-1, ..., 0$ et $s = 0, ..., 2^{\nu}-1$,
4. Récursion avant pour le code C_1, calcul des métriques $M_i^{\alpha}(s)$, $i = 0, ..., k-1$ et $s = 0, ..., 2^{\nu}-1$. Calcul et mémorisation des informations extrinsèques. Décisions binaires (à la dernière itération).

Le premier problème pratique réside dans la mémoire nécessaire au stockage des métriques $M_i^{\beta}(s)$. Traiter des messages codés de $k = 1000$ bits par exemple, avec des décodeurs à 8 états et une quantification des métriques sur 6 bits requiert à première vue une capacité de stockage de 48000 bits pour chaque décodeur. Dans un fonctionnement séquentiel (traitement alterné de C_1 et C_2), cette mémoire peut, bien sûr, être utilisée tour à tour par les deux décodeurs. La technique utilisée pour réduire notablement cette mémoire est celle de la fenêtre glissante (*sliding window*). Elle consiste (figure 7.15) à remplacer le traitement retour complet, de $i = k-1$ à 0, par une succession de traitements retour partiels, de $i = i_F$ à 0, puis de $i = 2i_F$ à i_F, de $i = 3i_F$ à $2i_F$ etc., où i_F est un intervalle de quelques dizaines de sections de treillis. Chaque traitement retour partiel comprend un « prologue » (trait en tiret), c'est-à-dire une étape sans mémorisation dont le seul but est d'estimer le plus correctement possible les métriques cumulées retour aux positions i_F, $2i_F$, $3i_F$, etc. Les parties en trait plein correspondent à des phases pendant lesquelles ces métriques sont mémorisées. La même mémoire peut être utilisée pour toutes les récursions retour partielles. Quant à la récursion aller, elle se déroule sans aucune discontinuité.

Le procédé réduit fortement la capacité de stockage nécessaire qui, par ailleurs, devient indépendante de la longueur des messages. L'inconvénient réside dans la nécessité d'effectuer des opérations supplémentaires – les prologues – qui peuvent augmenter la complexité de calcul totale de 10 à 20 %. Cependant, ces prologues peuvent être évités après la première itération si les estimations des métriques aux indices frontières sont mises en mémoire pour servir de points de départ aux calculs de l'itération suivante.

Le deuxième problème pratique est celui de la rapidité et de la latence de décodage. L'ampleur du problème dépend bien sûr de l'application et du rapport entre l'horloge du circuit de décodage et le débit des données. Si celui-ci est très élevé, les opérations peuvent être effectuées par une seule machine, suivant l'ordre séquentiel présenté plus haut. Dans les processeurs spécialisés de type DSP (*digital signal processor*), des co-processeurs câblés peuvent être disponibles pour accélérer le décodage. Dans les circuits dédiés de type *ASIC* (*application-specific integrated circuit*), l'accélération du décodage est obtenue en faisant appel au parallélisme, c'est-à-dire en multipliant le nombre d'opérateurs arithmétiques sans augmenter d'autant, si possible, la capacité des mémoires requises. Se posent alors généralement des problèmes d'accès à ces mémoires.

Notons d'abord que seule la connaissance de la permutation $i = \Pi(j)$ est nécessaire à la mise en œuvre du décodage itératif et non celle de la permutation

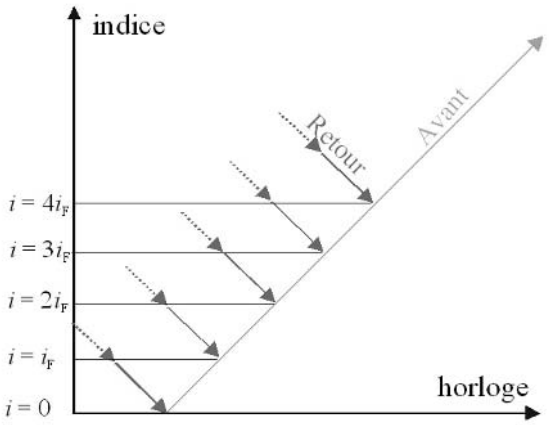


Figure 7.15 – Déroulement des récursions avant et retour dans la mise en œuvre de l'algorithme *MAP* par fenêtre glissante.

inverse Π^{-1}, comme le laisseraient faussement supposer les schémas de principe des figures 7.12 et 7.13. Considérons, par exemple, deux décodeurs *SISO* travaillant en parallèle au décodage des deux codes élémentaires du turbocode et s'appuyant sur deux mémoires à double port pour l'information extrinsèque (figure 7.16). Le décodeur DEC1 associé au premier code délivre et reçoit l'information extrinsèque dans l'ordre naturel i. Le décodeur DEC2 associé au deuxième code travaille selon l'indice j mais écrit et récupère ses données aux adresses $i = \Pi(j)$. La connaissance de Π^{-1}, qui pourrait poser problème selon le modèle de permutation retenu, n'est donc pas requise.

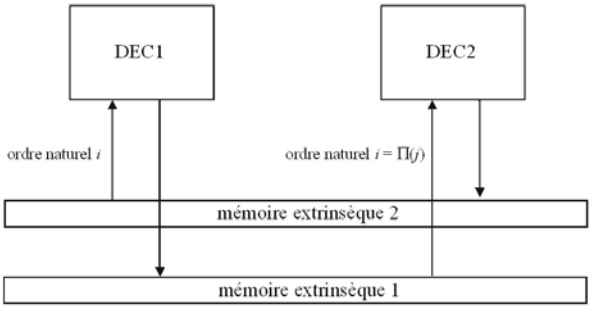


Figure 7.16 – La mise en œuvre du turbo-décodage ne requiert pas la connaissance explicite de Π^{-1}

En outre, les deux mémoires d'information extrinsèques peuvent être fusionnées en une seule en observant qu'une information extrinsèque qui vient d'être lue et exploitée par un décodeur n'a plus lieu d'être conservée. Elle peut donc

être remplacée aussitôt après par une autre donnée, qui peut être l'information extrinsèque sortante du même décodeur. La figure 7.17 illustre ce procédé qui impose une hypothèses légère : les indices de travail i et j ont même parité et la permutation $i = \Pi(j)$ inverse la parité. Par exemple, avec la permutation définie par (7.4), cette hypothèse est satisfaite si l'indice de départ i_0 est impair et la longueur du message k paire.

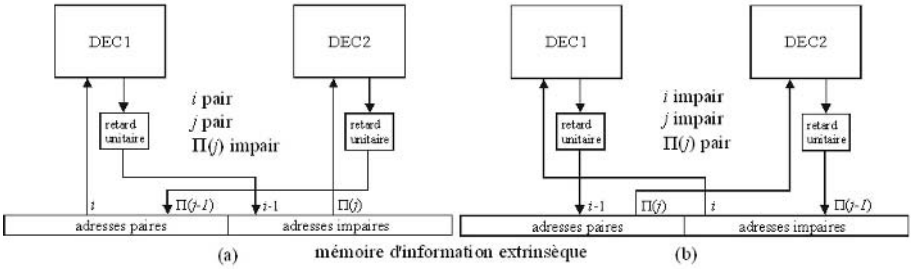


Figure 7.17 – En pratique, le stockage des informations extrinsèques ne fait appel qu'à une seule mémoire.

La mémoire d'information extrinsèque est partagée en deux pages correspondant aux sous-ensembles des adresses paires et impaires. L'accès à ces deux pages, en double port, est alterné régulièrement. En figure 7.17(a), aux cycles pairs, DEC1 lit à une adresse paire et écrit l'information extrinsèque produite lors du cycle précédent, à travers une mémoire tampon de retard unitaire, à une adresse impaire. Pendant ce temps, DEC2 lit à une adresse paire et écrit à une adresse impaire. En figure 7.17(b), durant les cycles impairs, les accès aux pages d'écriture-lecture sont échangés.

Pour augmenter encore le degré de parallélisme dans le décodeur itératif, les opérations de récursion avant et retour peuvent également être menées de front à l'intérieur de chacun des deux décodeurs (DEC1 et DEC2). Cela peut être mis en œuvre aisément en considérant le diagramme de la figure 7.15.

Enfin, suivant le modèle de permutation employé, le nombre de décodeurs élémentaires peut être augmenté au-delà de deux. Considérons par exemple la permutation circulaire définie par (7.14) et (7.16), avec un cycle $C = 4$ et k multiple de 4.

Les congruences de j et $\Pi(j)$ modulo 4, sont périodiques. Un parallélisme de degré 4 est alors possible suivant le principe décrit dans la figure 7.18 [7.26]. Pour chaque récursion aller ou retour (celles-ci pouvant également se faire en parallèle), quatre processeurs sont utilisés. Ces processeurs, au même instant, traitent des données dont les adresses ont des congruences modulo 4 différentes. Dans l'exemple de la figure, la récursion aller est considérée et on suppose que $k/4$ est aussi un multiple de 4. Alors, on fait débuter le premier processeur à l'adresse 0, le deuxième à l'adresse $k/4 + 1$, le troisième à l'adresse $k/2 + 2$ et enfin le quatrième à l'adresse $3k/4 + 3$. À chaque instant, les processeurs

avançant d'une place à chaque fois, les congruences modulo 4 des adresses sont toujours différentes. À travers un aiguilleur qui dirige les quatre processeurs vers quatre pages de mémoire correspondant aux quatre congruences possibles, les conflits d'adressage sont écartés. Si $k/4$ n'est pas un multiple de 4, les adresses de départ ne sont plus exactement 0, $k/4+1$, $k/2+2$, $3k/4+3$ mais le procédé est toujours applicable.

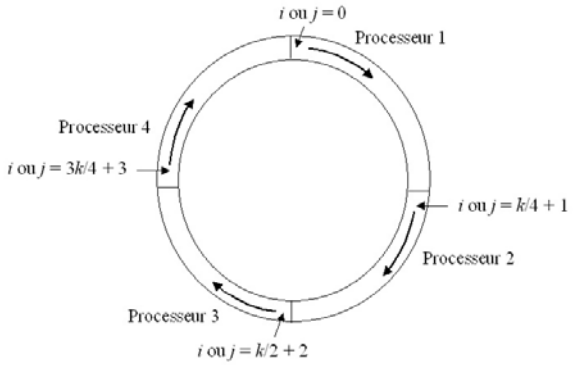


Figure 7.18 – Le cercle de la récursion avant est divisé en 4 quadrants.

Quelle que soit la valeur du cycle C, des degrés de parallélisme plus élevés, de valeur pC, peuvent être mis en œuvre. En effet, un quelconque multiple de C, cycle de base de la permutation, est aussi un cycle de la permutation, à la condition que pC soit un diviseur de k. C'est-à-dire que j modulo pC et $\Pi(j)$ modulo pC sont périodiques sur le cercle de longueur k, qui peut alors être découpé en pC fractions d'égales longueurs. Par exemple, un parallélisme de degré 64 est possible pour une valeur de k égale à 2048.

Toutefois, quel que soit le degré de parallélisme, une latence minimale est incontournable : le temps de réception et de mise en mémoire tampon d'un paquet reçu. Pendant que ce paquet est rangé en mémoire, le décodeur travaille sur les informations contenues dans le paquet précédent. Si ce décodage est réalisé en un temps au plus égal au temps de mémorisation, alors la latence totale de décodage est au maximum deux fois ce temps de mémorisation. Le niveau de parallélisme dans le décodeur est ajusté en fonction de cet objectif, qui peut être contraignant dans certains cas.

Pour en savoir plus sur l'implémentation des turbo-décodeurs et parmi toutes les publications sur ce sujet, [7.32] constitue un bon prolongement de cette section.

7.5 Les turbocodes m-binaires

Les turbocodes $m-$binaires sont construits à partir de codes Convolutifs Systématiques Récursifs (CSR) à m entrées binaires ($m \geq 2$). Il y a deux façons, au moins, de construire un code convolutif $m-$binaire : soit à partir du corps de Galois GF(2^m), soit à partir du produit cartésien $(GF(2))^m$. Il ne sera question ici que de la dernière manière de faire, plus commode. En effet, un code élaboré dans GF(2^m), avec une profondeur de mémoire ν, a $2^{\nu m}$ états possibles, alors que le nombre d'états du code défini dans $(GF(2))^m$, avec la même profondeur, peut être limité à 2^ν.

Les avantages de la construction m-binaire par rapport au schéma classique des turbocodes ($m = 1$), sont divers : meilleure convergence du processus itératif, plus grandes distances minimales, moindre poinçonnage, latence plus faible, robustesse envers la sous-optimalité de l'algorithme de décodage, en particulier quand l'algorithme *MAP* est simplifié en sa version Max-Log-*MAP* [7.33].

Le cas $m = 2$ a déjà été adopté dans les normes européennes de voie de retour du réseau satellite et du réseau terrestre ([7.19,7.20]) ainsi que dans la norme IEEE 802.16 [7.34]. Combiné avec la technique des treillis circulaires, ces turbocodes à 8 états dits double-binaires offrent de bonnes performances moyennes et une grande souplesse d'adaptation à différentes tailles de blocs et différents rendements, tout en gardant une complexité de décodage raisonnable.

7.5.1 Codeurs CSR $m-$binaires

La figure 7.19 représente la structure générale d'un codeur CSR $m-$binaire. Il utilise un générateur pseudo-aléatoire de mémoire de code ν et de matrice génératrice $\mathbf{G}$ (de taille $\nu \times \nu$). Le vecteur d'entrée $\mathbf{d}$ à m composantes est connecté aux différentes prises possibles grâce à une grille d'interconnexions dont la matrice binaire, de taille $\nu \times m$, est notée $\mathbf{C}$. Le vecteur $\mathbf{T}$ appliqué aux ν prises possibles du registre à l'instant i, est donné par :

$$\mathbf{T}_i = \mathbf{C}.\mathbf{d}_i \tag{7.49}$$

avec $\mathbf{d}_i = (d_{1,i} \dots d_{m,i})$.

Si l'on souhaite éviter les transitions parallèles dans le treillis du code, la condition $m \leq \nu$ doit être respectée et la matrice $\mathbf{C}$ doit être de rang plein. Excepté pour des cas très particuliers, ce codeur n'est pas équivalent à un codeur à une seule entrée sur lequel on présenterait successivement $d_1, d_2, \cdots, d_m$. Un codeur $m-$binaire n'est donc pas décomposable en général.

La sortie redondante de la machine (non représentée sur la figure) est calculée à l'instant i selon l'expression :

$$y_i = \sum_{j=1\dots m} d_{j,i} + \mathbf{R}^\mathbf{T}\mathbf{S}_i \tag{7.50}$$

où $\mathbf{S}_i$ est le vecteur d'état à l'instant i et $\mathbf{R}^\mathbf{T}$ est le vecteur transposé de redondance. La p-ième composante de $\mathbf{R}$ vaut 1 si la p-ième composante de $\mathbf{S}_i$

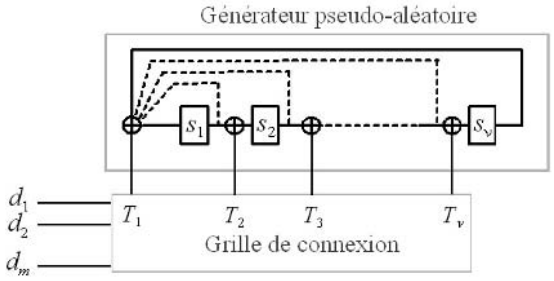


Figure 7.19 – Structure générale d'un codeur CSR m-binaire de mémoire de code ν. L'indice temporel n'est pas représenté.

est utilisée dans la construction de y_i et 0 sinon. On peut montrer que y_i peut aussi s'écrire comme :

$$y_i = \sum_{j=1\ldots m} d_{j,i} + \mathbf{R^T G^{-1} S}_{i+1} \tag{7.51}$$

à condition que :

$$\mathbf{R^T G^{-1} C} \equiv \mathbf{0} \tag{7.52}$$

L'expression (7.50) assure d'une part que le poids de Hamming du vecteur $(d_{1,i}, d_{2,i}, \cdots, d_{m,i}, y_i)$ est au moins égal à deux quand on s'écarte du chemin de référence (chemin « tout à zéro »), dans le treillis. En effet, inverser une seule composante de $\mathbf{d}_i$ modifie la valeur de y_i. D'autre part, l'expression (7.51) indique que le poids de Hamming du même vecteur est aussi au moins égal à deux lorsque l'on rejoint le chemin de référence. En conclusion, les relations (7.50) et (7.51) ensemble garantissent que la distance libre du code, dont le rendement est $R = m/(m+1)$, est au moins égale à 4, quel que soit m.

7.5.2 Turbocodes m-binaires

Nous considérons une concaténation parallèle de deux codeurs CSR m-binaires associés à une fonction de permutation de N mots de m bits ($k = mN$) (figure 7.20). Les blocs sont codés deux fois par ce code bi-dimensionnel, dont le rendement est $m/(m+2)$. Le principe des treillis circulaires est adopté pour permettre le codage des blocs sans séquence de terminaison et sans effet de bord.

Les avantages de cette construction par rapport aux turbocodes classiques sont les suivants :

- **Meilleure convergence**. Cet avantage, observé tout d'abord dans [7.35], commenté dans [7.36] et, d'une manière différente, dans [7.33], s'explique par une plus faible densité d'erreurs sur chacune des deux dimensions du processus itératif. Reprenons la relation (7.8) qui fournit la borne supérieure de la distance spatiale cumulée pour un code binaire et adaptons-la

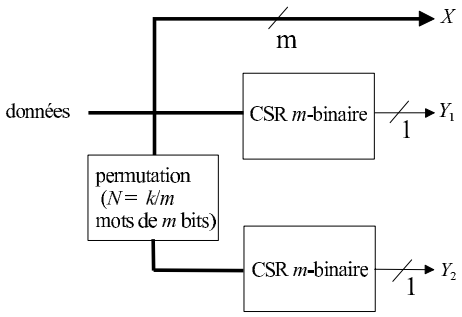


Figure 7.20 – Turbocodeur m-binaire.

à un code m-binaire :

$$\sup S_{\min} = \sqrt{\frac{2k}{m}} \tag{7.53}$$

Pour un rendement de codage R, le nombre de bits de parité produits par la séquence de longueur cumulée $\sup S_{\min}$ est :

$$n_{parit}(\sup S_{\min}) = \left(\frac{1-R}{R}\right)\frac{m}{2}\sup S_{\min} = \left(\frac{1-R}{R}\right)\sqrt{\frac{mk}{2}} \tag{7.54}$$

Ainsi, en remplaçant un turbocode binaire ($m = 1$) par un code double-binaire ($m = 2$), le nombre de bits de parité dans la séquence considérée est multiplié par $\sqrt{2}$, bien que la distance spatiale cumulée ait été réduite dans le même rapport. Parce que les bits de parité sont des informations locales pour les deux décodeurs élémentaires (et ne sont donc pas source de corrélation entre ceux-ci), en augmenter le nombre améliore la convergence. Augmenter m au delà de 2 améliore encore un peu le comportement vis-à-vis de la corrélation mais les effets sont moins visibles que lors du passage de $m = 1$ à $m = 2$.

- **Plus grandes distances minimales**. Comme expliqué ci-dessus, le nombre de bits de parité produits par des séquences *RTZ* de poids d'entrée 2 est augmenté en utilisant des codes m-binaires. Il en est de même pour toutes les séquences *RTZ* simples telles que définies dans la section 7.3.2. Le nombre de bits de parité pour ces séquences est au moins égal à $n_{parit}(\sup S_{\min})$. Les distances de Hamming correspondantes sont donc encore plus élevées que celles qui sont obtenues avec des codes binaires et contribuent d'autant moins à la DMH du turbocode. Quant aux distances associées aux motifs *RTZ* multiples, qui sont généralement ceux qui fixent la DMH, elles sont fonction de la qualité de la permutation mise en œuvre (voir section 7.3.2).
- **Moindre poinçonnage**. Pour obtenir des rendements de codage supérieurs à $m/(m+1)$ à partir du codeur de la figure 7.20, il n'est pas nécessaire de supprimer autant de symboles de redondance qu'avec un codeur binaire. La performance des codes élémentaires en est améliorée

comme le montre la figure 7.21. Dans cette figure sont comparés le pouvoir de correction de codes convolutifs de rendements 2/3 et 6/7, dans la version binaire ($m = 1$) et double-binaire ($m = 2$).

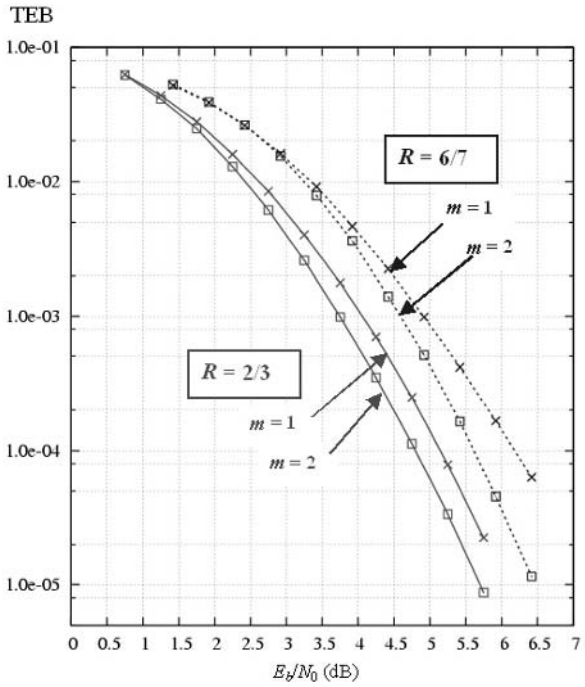


Figure 7.21 – Performance de codes convolutifs simples binaires ($m = 1$) et double-binaire ($m = 2$), pour $R = 2/3$ et $R = 6/7$.

- **Latence réduite**. Aussi bien du point de vue du codage que du décodage, la latence (c'est-à-dire le nombre de cycles d'horloge nécessaire au traitement) est divisée par m puisque les données sont traitées par groupes de m bits. Il se peut toutefois que le chemin critique du circuit de décodage soit augmenté par rapport au cas $m = 1$ car plus de données sont à considérer dans un cycle d'horloge. Des solutions de parallélisme telles que celles proposées dans [7.37] peuvent aider à augmenter la fréquence du circuit.
- **Robustesse du décodeur**. Pour les turbocodes binaires, la différence de performance entre l'algorithme *MAP* et ses versions simplifiées ou entre l'algorithme *MAP* et le *SOVA*, varie de $0,2$ à $0,6$ dB, suivant la taille des blocs et les rendements de codage. Cette différence est divisée par deux quand on utilise les turbocodes double-binaires et peut même être plus faible pour $m > 2$. Cette propriété favorable (et légèrement surprenante) peut s'expliquer de la manière suivante : pour un bloc de taille donnée (k bits), plus le nombre d'étapes dans le treillis est faible, plus le décodeur

est proche du décodeur à Maximum de Vraisemblance (MV), quel que soit l'algorithme sur lequel il s'appuie. À la limite, un treillis réduit à une seule étape et contenant donc tous les mots de code possibles est équivalent à un décodeur à MV.

Turbocode double-binaire à 8 états

La figure 7.22(a) donne quelques exemples de performances obtenues avec le turbocode de [7.19], pour un rendement 2/3. Les paramètres des codeurs constituants sont :

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, \mathbf{R} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

La fonction de permutation utilise à la fois un désordre inter et intra-symbole. On peut notamment observer :

- de bonnes performances moyennes pour ce code dont la complexité de décodage reste très raisonnable (environ 18000 portes par itération plus la mémoire) ;
- une certaine cohérence vis-à-vis de la variation de performance avec la taille des blocs (en accord avec les courbes des figures 3.6, 3.9, 3.10). La même cohérence pourrait aussi être observée sur la variation de la performance avec le rendement de codage ;
- une quasi-optimalité du décodage aux faibles taux d'erreurs. La courbe asymptotique théorique pour 188 octets a été calculée à partir de la seule connaissance de la distance minimale du code (c'est-à-dire 13 avec une multiplicité relative de $0,5$) et non à partir du spectre total des distances. Malgré cela, la différence entre la courbe asymptotique et la courbe obtenue par simulation n'est que de $0,2$ dB pour un TEP de 10^{-7}.

Turbocode double-binaire à 16 états

L'extension du schéma précédent vers des codeurs élémentaires à 16 états permet d'augmenter nettement les distances minimales. On peut par exemple choisir :

$$\mathbf{G} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{R} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Pour le turbocode de rendement 2/3, toujours avec des blocs de 188 octets, la distance minimale obtenue est égale à 18 (multiplicité relative de $0,75$) au lieu de 13 pour le code à 8 états. La figure 7.22(b) montre le gain obtenu pour de faible taux d'erreurs : environ 1 dB pour un TEP de 10^{-7} et 1,4 dB asymptotiquement en considérant les distances minimales respectives. On peut noter que le seuil de convergence est à peu près le même pour les décodeurs à 8

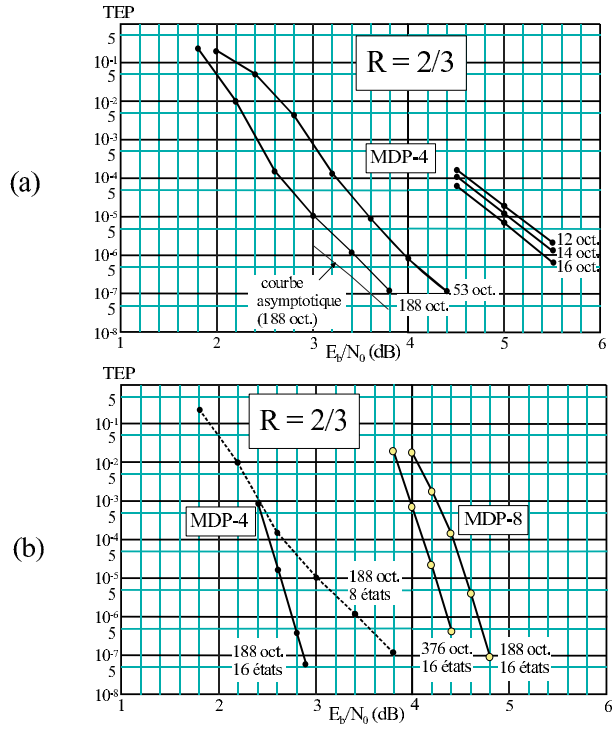


Figure 7.22 – (a) Performance en TEP d'un turbocode double-binaire à 8 états pour des blocs de 12, 14, 16, 53 et 188 octets. MDP-4, bruit BABG et rendement 2/3. Décodage Max-Log-*MAP* avec des échantillons d'entrée de 4 bits et 8 itérations. (b) Performance en TEP d'un turbocode double-binaire à 16 états pour des blocs de 188 octets (MDP-4 et MDP-8) et 376 octets (MDP-8), bruit BABG et rendement 2/3. Décodage Max-Log-*MAP* avec des échantillons d'entrée de 4 bits (MDP-4) ou 5 bits (MDP-8) et 8 itérations.

et à 16 états, les courbes étant pratiquement identiques pour un TEP supérieur à 10^{-4}. La Limite théorique (LT), pour $R = 2/3$ et pour une taille de blocs de 188 octets, est 1,7 dB. Les performances du décodeur dans ce cas sont : LT + 0,9 dB pour un TEP de 10^{-4} et LT + 1,3 dB pour un TEP de 10^{-7}. Ces écarts sont typiques de ce que l'on obtient dans la plupart des configurations de rendement et de taille de bloc.

Le remplacement de la modulation MDP-4 par une modulation MDP-8, suivant l'approche dite pragmatique, donne les résultats reportés en figure 7.22(b), pour des blocs de 188 et 376 octets. Là encore, de bonnes performances du code double-binaire peuvent être observées, avec des pertes par rapport aux limites théoriques (qui sont environ 3,5 et 3,3 dB, respectivement) proches de celles obtenues avec une modulation MDP-4. L'association des turbocodes avec différentes modulations est détaillée dans le chapitre 8.

Pour un système particulier, le choix entre un turbocode à 8 ou à 16 états dépend, outre de la complexité souhaitée pour le décodeur, du taux d'erreurs cible. Pour simplifier, disons qu'un turbocode à 8 états suffit pour des TEP supérieurs à 10^{-4}. C'est généralement le cas pour les transmissions avec possibilité de répétition (*ARQ : Automatic Repeat reQuest*). Pour des TEP inférieurs, typiques de la diffusion ou des applications de mémoire de masse, le code à 16 états est largement préférable.

7.6 Outils d'analyse

7.6.1 Performances théoriques

La figure 1.6 met en évidence deux paramètres essentiels permettant d'évaluer la performance d'un code correcteur d'erreurs et de son décodeur :

- le *gain asymptotique* mesurant le comportement à faible taux d'erreurs du système codé. Celui-ci est principalement dicté par la DMH du code (voir section 1.5). Une faible valeur de la DMH entraîne un fort changement de pente (*flattening*) dans la courbe de taux d'erreurs. Lorsque le gain asymptotique est atteint, la courbe TEB(E_b/N_0) avec codage devient parallèle à la courbe sans codage.
- le *seuil de convergence* défini comme le rapport signal à bruit à partir duquel le système codé devient plus performant que le système de transmission non codé ;

Dans le cas des turbocodes et du processus itératif de leur décodage, il n'est pas toujours aisé d'estimer les performances aussi bien du côté du gain asymptotique que de celui de la convergence. Les méthodes d'estimation ou de détermination de la distance minimale proposées par Berrou *et al* [7.38], Garello *et al* [7.39] et Crozier *et al* [7.40] sont présentées dans la suite de ce chapitre. La méthode du diagramme *EXIT* proposée par ten Brink [7.42] pour estimer le seuil de convergence est également introduite.

7.6.2 Comportement asymptotique

La qualification par simulation des performances à faibles taux d'erreurs des codes correcteurs d'erreurs demande une puissance de calcul importante. Il est cependant possible d'estimer ces performances lorsque la DMH $d_{\min}$ et la multiplicité sont connues (voir section 1.5). Ainsi le taux d'erreurs paquet à fort rapport signal à bruit E_b/N_0 est donné par le premier terme de la borne de l'union (*UB* pour *union bound*). L'expression de *UB* est décrite par la relation (3.21), et l'estimation du TEP, donnée par l'équation (1.16), est reproduite ici :

$$\text{TEP} \approx \frac{1}{2} N(d_{\min}) \text{erfc}\left(\sqrt{R d_{\min} \frac{E_b}{N_0}}\right) \tag{7.55}$$

où $N(d_{\min})$, la multiplicité, représente le nombre de mots de code à la distance minimale.

La distance minimale d'un code n'est, dans le cas général, pas simple à déterminer sauf si le nombre de mots du code est suffisamment petit pour pouvoir en dresser la liste exhaustive ou bien si des propriétés particulières du code permettent d'établir une expression analytique de cette grandeur (par exemple, la distance minimale d'un code produit est égale au produit des distances minimales des codes constituants). Dans le cas des turbocodes convolutifs, la distance minimale ne s'obtient pas de manière analytique, les seules méthodes proposées sont basées sur l'énumération, totale ou partielle [7.41], des mots de code dont le poids d'entrée est inférieur ou égal à la distance minimale. Ces méthodes ne sont applicables en pratique que pour des tailles de blocs et des distances minimales faibles, c'est pourquoi elles ne seront pas décrites ici.

Méthode de l'impulsion d'erreur

Cette méthode, proposée par Berrou *et al* [7.38], n'est pas basée sur l'analyse des propriétés du code mais sur la capacité de correction du décodeur. Son principe, illustré par la figure 7.23, consiste à superposer à la séquence d'entrée du décodeur une impulsion d'erreur dont on fait croître l'amplitude A_i jusqu'à ce le décodeur ne sache plus la corriger.

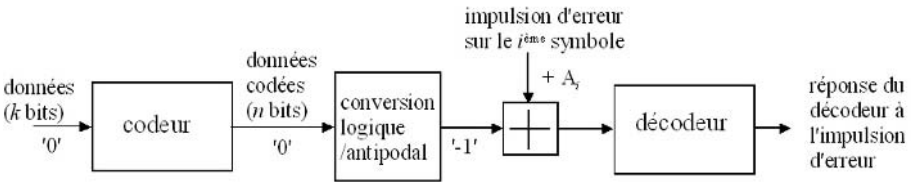


Figure 7.23 – Schéma de principe de la méthode d'impulsion d'erreur.

Le code considéré étant linéaire, la séquence transmise est supposée être la séquence « tout 0 ». L'opération de codage produit alors des mots de code qui ne contiennent, eux aussi, que des 0. Ceux-ci sont ensuite convertis en valeurs réelles égales à -1. Si cette succession de symboles était directement appliquée au décodeur, celui-ci ne rencontrerait aucune difficulté à retrouver le message d'origine car le canal de transmission est parfait.

La méthode proposée consiste à ajouter une impulsion d'erreur au i-ième symbole ($0 \leq i \leq k-1$) de la séquence d'information (partie systématique), c'est-à-dire à transformer un symbole « -1 » en un symbole ayant une valeur positive égale à $-1 + A_i$. Si l'amplitude A_i est suffisamment importante, le décodeur ne converge pas vers le mot « tout 0 ». Notons A_i^* l'amplitude maximale de l'impulsion à la position i ne mettant pas en défaut le décodeur. Il est montré dans [7.38] que, si le décodeur effectue un décodage à maximum de

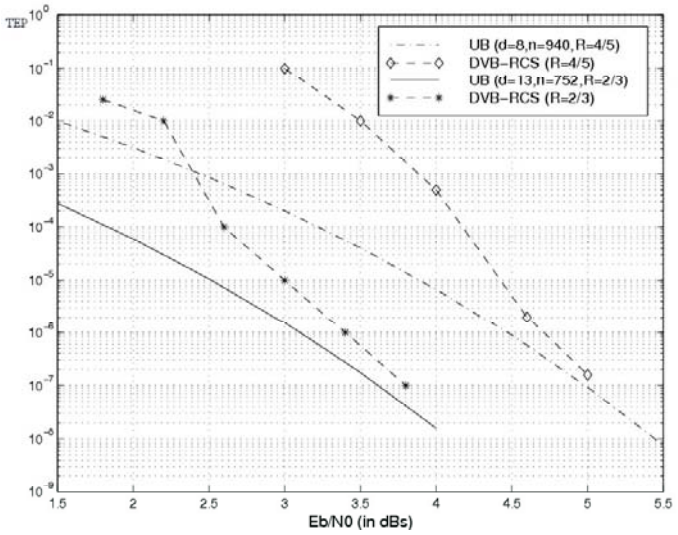


Figure 7.24 – TEP mesuré et estimé (*UB*) du turbocode DVB-RCS pour la transmission de blocs MPEG (188 octets) avec des rendements de codage 2/3 et 4/5. Modulation MDP-4 et canal gaussien.

vraisemblance, la *distance impulsionnelle* $d_{imp} = \min_{i=0,\cdots,k-1}(A_i^*)$, est aussi la distance minimale $d_{\min}$ du code.

Il n'est en général pas nécessaire de tester toutes les positions de la séquence. Pour un code invariant par décalage (c'est le cas des codes convolutifs), il suffit d'appliquer l'impulsion d'erreur sur une seule position du bloc de données. Pour un code présentant une périodicité de période P, il est nécessaire de tester P positions. Cette méthode est applicable à n'importe quel code linéaire, pour toute taille de bloc et tout rendement de codage et ne requiert que de quelques secondes à quelques minutes de calcul sur un ordinateur courant, le temps de calcul étant une fonction linéaire de la taille du bloc ou de sa période P.

Lorsque le décodage n'est pas à maximum de vraisemblance, la méthode n'est plus rigoureuse et ne délivre qu'une estimation de la distance minimale. En outre, la multiplicité des mots de code à la distance $d_{\min}$ n'est pas fournie et la formule (7.55) ne peut être appliquée sans hypothèses particulières sur les propriétés du code. Dans le cas des turbocodes, deux hypothèses réalistes peuvent être formulées pour estimer la multiplicité : un seul mot de code à la distance A_i^* a son i-ième bit d'information à 1 (unicité), les grandeurs A_i^* correspondant à toutes les positions i proviennent de mots de codes distincts (non recouvrement).

Une estimation du TEP est alors donnée par :

$$\text{TEP} \approx \frac{1}{2}\sum_{i=0}^{k-1} \text{erfc}(\sqrt{RA_i^* \frac{E_b}{N_0}}) \tag{7.56}$$

La première hypothèse (unicité) sous-évalue la valeur du taux d'erreurs au contraire de la seconde (non recouvrement) qui la surévalue et au total les deux effets se compensent. À titre d'exemple, la figure 7.24 compare les performances mesurées du turbocode DVB-RCS, pour deux rendements de codage, avec leur estimation déduite de (7.56). Les paramètres obtenus par la méthode de l'impulsion d'erreur sont :

- $d_{\min} = 13$ et $n(d_{\min}) = 752$ pour $R = 2/3$
- $d_{\min} = 8$ et $n(d_{\min}) = 940$ pour $R = 4/5$

À un taux d'erreurs paquet de 10^{-7}, moins de 0,2 dB séparent les courbes mesurées et estimées.

Méthode de l'impulsion d'erreur modifiée

L'approche de Garello *et al* [7.39] est similaire à la méthode de l'impulsion d'erreur présentée précédemment. En effet, il s'agit de placer une impulsion au rang i dans le mot de code « tout 0 ». Cette fois, l'amplitude de l'impulsion est suffisamment élevée pour que le décodeur ne converge pas vers le mot de code « tout 0 » mais vers une autre séquence qui contient un 1 à la position i. Par ailleurs, un bruit gaussien est ajouté à la séquence d'entrée du décodeur, ce qui tend à aider celui-ci à converger vers le mot concurrent de poids le plus faible. C'est ce qui se produit le plus souvent quand le niveau de bruit est bien ajusté. Dans tous les cas, le poids du mot de code délivré par le décodeur est une limite supérieure de la distance minimale de tous les mots de code contenant un 1 au rang i. La distance minimale et la multiplicité sont estimées en balayant toutes les positions. Cet algorithme fonctionne très bien pour des distances faibles et moyennes.

Méthode de la double impulsion d'erreur

La méthode proposée par Crozier *et al* [7.40] est une amélioration de la précédente, au prix d'un temps de calcul plus élevé. Il s'agit de placer une première impulsion de fort niveau au rang i et une seconde à un rang j à droite de i et tel que $j - i < R$. La limite supérieure de R est $2D$ où D est un majorant de la distance à évaluer. Il est ensuite appliqué un décodage semblable à celui décrit précédemment mais avec une plus forte probabilité d'obtenir un mot de code à la distance minimale. Le temps de calcul est accru dans un rapport R.

7.6.3 Convergence

Un décodeur *SISO* peut être vu comme un processeur qui transforme une de ses grandeurs d'entrée, le LRV de l'information extrinsèque servant d'information *a priori*, en un LRV extrinsèque de sortie. Dans un décodage itératif, les caractéristiques de l'information extrinsèque fournie par le décodeur 1 dépendent de l'information extrinsèque fournie par le décodeur 2 et vice-versa. Le degré de dépendance entre les informations extrinsèques d'entrée et de sortie peut être mesuré par l'information mutuelle moyenne (IMM).

L'idée mise en œuvre par Ten Brink [7.42] est de suivre sur un diagramme, appelé diagramme *EXIT* (*EXtrinsic Information Transfer chart*), l'échange d'information extrinsèque à travers les décodeurs *SISO* travaillant en parallèle. Pour élaborer le diagramme *EXIT*, il est nécessaire de connaître les caractéristiques de transfert de l'information extrinsèque de chaque décodeur *SISO* utilisé dans le décodage. Cette section présente l'établissement de la fonction de transfert de l'information extrinsèque pour un décodeur *SISO*, puis la construction du diagramme *EXIT*, et enfin l'analyse de la convergence du décodeur itératif.

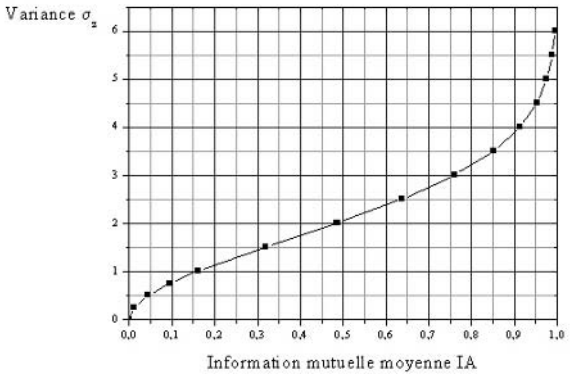


Figure 7.25 – Variation de la variance σ_z en fonction de l'information mutuelle moyenne IA

Fonction de transfert d'un décodeur *SISO* de l'information extrinsèque

a. Définition de l'information mutuelle moyenne (IMM)

Si l'information extrinsèque pondérée z sur l'élément binaire codé $x \in \{-1, +1\}$ suit une densité de probabilité conditionnelle $f(z|x)$, l'IMM $I(z, x)$

mesure la quantité d'information apportée en moyenne par z sur x et vaut :

$$\mathrm{I}(z,x) = \frac{1}{2} \sum_{x=-1,+1} \int_{-\infty}^{+\infty} f(z|x) \times \log_2 \left[\frac{2f(z|x)}{f(z|-1) + f(z|+1)} \right] dz \qquad (7.57)$$

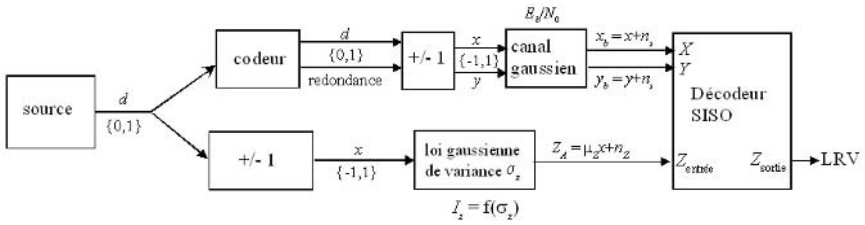


Figure 7.26 – algorithme de détermination de la fonction de transfert $IE = T(IA, Eb/N0)$

b. Définition de l'information mutuelle moyenne a priori

Hypothèses :

- Hyp. 1 : lorsque l'entrelacement est de taille suffisamment grande, la distribution de l'information extrinsèque d'entrée peut être approchée par une distribution gaussienne après quelques itérations.
- Hyp. 2 : la densité de probabilité $f(z|x)$ vérifie la condition de symétrie exponentielle, à savoir $f(z|x) = f(-z|x)exp(-z)$.

La première hypothèse permet de modéliser le LRV *a priori* Z_A d'un décodeur *SISO* par une variable affectée d'un bruit gaussien indépendant n_z, de variance σ_z et d'espérance μ_z, appliquée au symbole d'information transmis x selon l'expression :

$$Z_A = \mu_z x + n_z$$

La seconde hypothèse impose $\sigma_z^2 = 2\mu_z$. L'amplitude de l'information extrinsèque est donc modélisée par la distribution suivante :

$$f(\lambda|x) = \frac{1}{\sqrt{4\pi\mu_z}} \exp\left[-\frac{(\lambda - \mu_z x)^2}{4\mu_z} \right] \qquad (7.58)$$

De (7.57) et (7.58), en remarquant que $f(z|\ 1) = f(-z|-1)$, on déduit l'information mutuelle moyenne *a priori* :

$$\mathrm{IA} = \int_{-\infty}^{+\infty} \frac{1}{\sqrt{4\pi\mu_z}} \exp\left[-\frac{(\lambda - \mu_z)^2}{4\mu_z} \right] \times \log_2 \left[\frac{2}{1 + \exp(-\lambda)} \right] d\lambda$$

soit encore :

$$\mathrm{IA} = 1 - \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}\sigma_z} \exp\left[-\frac{\left(\lambda - \sigma_z^2/2\right)^2}{2\sigma_z^2} \right] \times \log_2 \left[1 + \exp(-\lambda) \right] d\lambda \qquad (7.59)$$

On peut noter que $\lim_{\sigma_z \to 0} \text{IA} = 0$ (l'information extrinsèque n'apporte aucune information sur la donnée x) et que $\lim_{\sigma_z \to +\infty} \text{IA} = 1$ (l'information extrinsèque détermine parfaitement la donnée x).
IA est une fonction croissante et monotone de σ_z, elle est donc inversible. La fonction $\sigma_z = f(\text{IA})$ est représentée en figure 7.25.

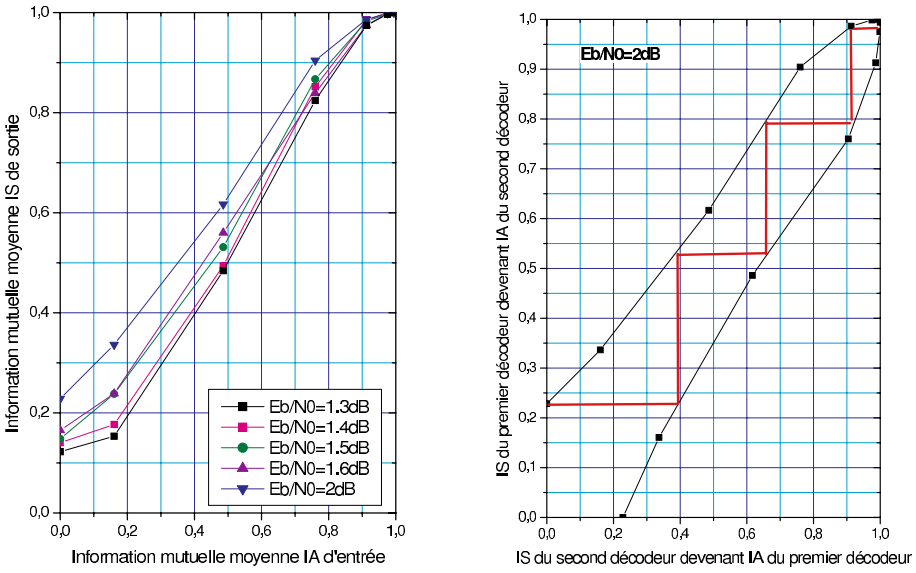


Figure 7.27 – (a) Caractéristique de transfert de l'information extrinsèque pour un codeur binaire 16 états de rendement 2/3 et un décodage *MAP* à différents Eb/N0. (b) Diagramme *EXIT* et trajectoire pour le turbocode correspondant, avec un entrelacement pseudo-aléatoire sur 20000 bits, pour $E_b/N_0 = 2dB$.

c. Calcul de l'information mutuelle moyenne de sortie

La relation (7.57) permet de calculer l'information mutuelle moyenne IS liée à l'information extrinsèque produite par le décodeur *SISO* :

$$\text{IS} = \frac{1}{2} \sum_{x=-1,+1} \int_{-\infty}^{+\infty} f_s\left(z|\, x\right) \times \log_2 \left[\frac{2 f_s\left(z|\, x\right)}{f_s\left(z|\, -1\right) + f_s\left(z|\, +1\right)} \right] dz \qquad (7.60)$$

On peut noter que $\text{IS} \in [0, 1]$.

La distribution f_s n'est pas gaussienne, il est donc nécessaire d'utiliser un outil de calcul numérique pour la déterminer, ce qui constitue le gros désavantage de cette méthode. Si on regarde l'IMM de sortie IS comme une fonction de IA et du rapport signal à bruit E_b/N_0, la fonction de transfert de l'information extrinsèque est définie par :

$$IE = T(IA, E_b/N_0) \qquad (7.61)$$

d. Méthode pratique pour obtenir la fonction de transfert de l'information extrinsèque

La figure 7.26 décrit le cheminement dans l'établissement de la caractéristique de transfert de l'information extrinsèque d'un décodeur *SISO*.

- Étape 1 : Génération du message pseudo aléatoire d à émettre ; au moins 10000 bits sont nécessaires pour que les propriétés statistiques soient représentatives.
- Étape 2 : Codage des données avec un rendement R puis modulation MDP-2 du signal ; les données systématiques et de redondance appartiennent à l'alphabet {-1,+1}.
- Étape 3 : Application d'un bruit gaussien de rapport signal à bruit E_b/N_0 (en dB), de variance

$$\sigma = \sqrt{\frac{1}{2} \cdot \frac{10^{-0,1 \times E_b/N_0}}{R}}$$

- Étape 4 : Application aux données émises (stockées dans un fichier) de la loi normale $N(\mu_z, \sigma_z)$ correspondant à l'information mutuelle moyenne souhaitée IA (voir figure 7.25) pour obtenir la distribution d'information extrinsèque *a priori*.
- Étape 5 : Initialisation du décodeur *SISO* avec les LRV *a priori* (il pourra s'avérer nécessaire suivant l'algorithme de décodage choisi de transformer les LRV en probabilités).
- Étape 6 : Récupération dans un fichier des LRV en sortie du décodeur *SISO* (correspondant à une demi-itération du processus de décodage).
- Étape 7 : Utilisation d'un logiciel de calcul numérique pour évaluer IS (relation (7.60)).
  - Tracer les histogrammes des répartitions des LRV de sortie en fonction du bit émis (d'où la nécessité de stocker ces informations dans deux fichiers).
  - Évaluer l'intégrale par la méthode des trapèzes.
- Le résultat est l'IMM de sortie IS correspondant à l'IMM d'entrée IA.

e. Un exemple

Les simulations ont été faites sur un turbocode binaire à 16 états de rendement 2/3, avec un entrelacement pseudo-aléatoire de 20000 bits. L'algorithme de décodage est l'algorithme *MAP*. La figure 7.27(a) fournit la relation entre IS et IA en fonction du rapport signal à bruit du canal gaussien.

Diagramme *EXIT*

La caractéristique de transfert de l'information extrinsèque est maintenant connue pour un décodeur *SISO*. Dans le cas du décodage itératif, la sortie du décodeur 1 devient l'entrée du décodeur 2 et vice versa. Les courbes IS1 = $f(I\text{A1} = \text{IS2})$ et IS2 = $f(\text{IA2} = \text{IS1})$, identiques à une symétrie près si les décodeurs *SISO* sont les mêmes, sont placées sur le même graphique comme le montre la figure 7.27(b). Dans le cas d'un rapport signal à bruit suffisamment élevé (ici 2 dB), les deux courbes ne présentent pas d'intersection en dehors du point de coordonnées (1,1) qui matérialise la connaissance du message reçu. Il est alors possible en partant d'une information mutuelle nulle de suivre l'échange des informations extrinsèques au fur et à mesure des itérations. Sur l'exemple de la figure 7.27(b), une arrivée au point (1,1) s'effectue en 3,5 itérations.

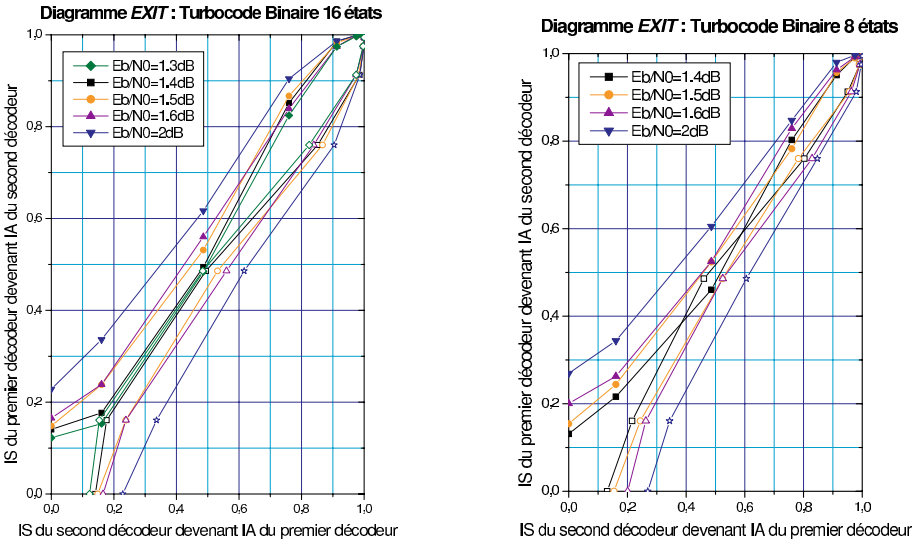


Figure 7.28 – Diagrammes *EXIT* pour différents E_b/N_0 dans le cas de turbocodes binaires, rendement 2/3, entrelacement pseudo-aléatoire de 20000 bits, décodage *MAP* (a) 16 états et (b) 8 états.

Lorsque le rapport signal à bruit est trop faible, les courbes présentent comme dans le cas $E_b/N_0 = 1,4$ dB dans la figure 7.28(b) des points d'intersection autres que le point (1,1). Le processus itératif démarrant d'une IMM nulle en entrée ne pourra donc pas aboutir à une détermination parfaite du message. Le rapport signal à bruit minimal pour lequel il n'existe pas d'intersection autre que le point (1,1) est le seuil de convergence du turbocodeur. Dans l'exemple simulé, cette convergence peut être estimée autour de 1,4 dB pour les turbocodes binaires à 16 états (figure 7.28(a)) et 8 états (figure 7.28(b)).

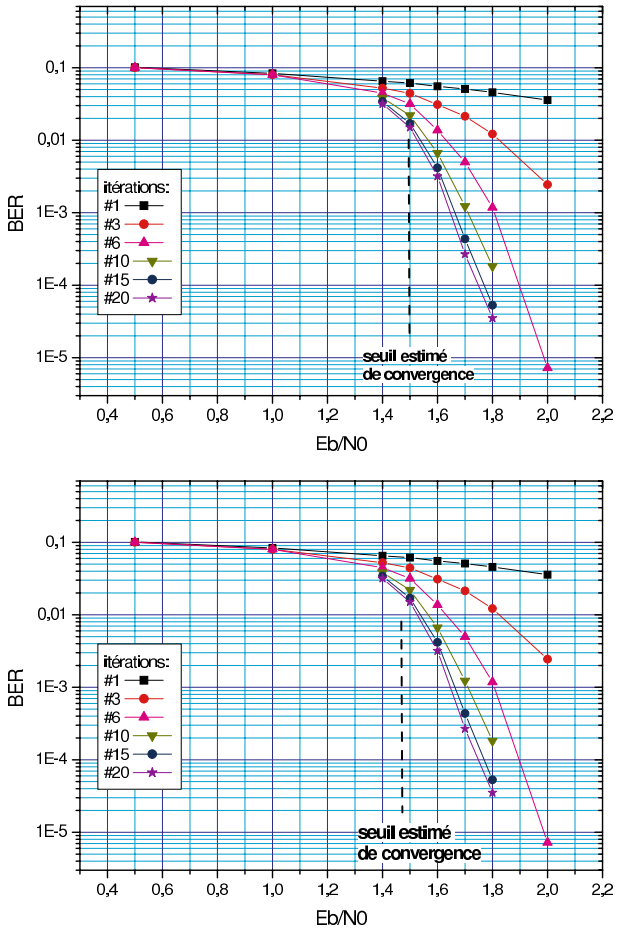


Figure 7.29 – Taux d'erreurs binaire d'un turbocode binaire 16 états (a) et 8 états (b) de rendement 2/3, avec entrelacement pseuso-aléatoire de 20000 bits. Décodage MAP avec 1, 3, 6, 10, 15 et 20 itérations et comparaison avec le seuil de convergence estimé par la méthode *EXIT*.

La figure 7.29 représente les performances de turbocodes binaires à 16 états et 8 états en fonction du nombre d'itérations et les compare avec le seuil de convergence estimé par la méthode du diagramme *EXIT*.

7.7 Bibliographie

[7.1] G. Battail, « Coding for the Gaussian channel : the promise of weighted-output decoding », *International Journal of Satellite Communications*, Vol. 7, pp. 183-192, 1989.

[7.2] G. Battail, « Pondération des symboles décodés par l'algorithme de Viterbi », *Ann. Télécommun.*, Fr., 42, N ° 1-2, pp. 31-38, Jan. 1987.

[7.3] J. Hagenauer and P. Hoeher, « A Viterbi algorithm with soft-decision outputs and its applications », *Proc. Globecom '89*, Dallas, Texas, pp. 47.11-47-17, Nov. 1989.

[7.4] J. Hagenauer and P. Hoeher, « Concatenated Viterbi-decoding », *Proc. Int. Workshop on Inf. Theory*, Gotland, Sweden, Aug-Sep. 1989.

[7.5] P. Elias, « Error-free coding », *IEEE Trans. Info. Theory*, vol. 4, N ° 4, pp. 29-39, Sept. 1954.

[7.6] R. M. Tanner, « A recursive approach to low complexity codes », *IEEE Trans. Inform. Theory*, Vol. IT-27, pp. 533-547, Sept. 1981.

[7.7] R. G. Gallager, « Low-density parity-check codes », *IRE Trans. Inform. Theory*, Vol. IT-8, pp. 21-28, Jan. 1962.

[7.8] C. Berrou, P. Adde, E. Angui and S. Faudeil, « A low complexity soft-output Viterbi decoder architecture », *Proc. of ICC '93*, Geneva, pp. 737-740, May 1993.

[7.9] C. Berrou, *Procédé de codage correcteur d'erreurs à au moins deux codages convolutifs systématiques en parallèle, procédé de décodage itératif, module de décodage et décodeur correspondants*, brevet France dépôt N° 91 05280, Europe N°92 460013.3, USA N°07/870,814, France Télécom et TDF, avril 1992

[7.10] C. Berrou, A. Glavieux and P. Thitimajshima, « Near Shannon limit error-correcting coding and decoding : turbo-codes », *Proc. ICC '93*, Geneva, pp. 1064-1070, May 1993.

[7.11] A. R. Calderbank, « The art of signaling : fifty years of coding theory », *IEEE Trans. Info. Theory*, vol. 44, N ° 6, pp. 2561-2595, Oct. 1998.

[7.12] C. Berrou and A. Glavieux, *Reflections on the Prize paper :* « Near optimum error correcting coding and decoding : turbo-codes », *IEEE IT Society Newsletter*, Vol. 48, N ° 2, June 1998, également disponible à www.ieeeits.org/publications/nltr/98_jun/reflections.html.

[7.13] C. Berrou, C. Douillard and M. Jézéquel, « Multiple parallel concatenation of circular recursive convolutional (CRSC) codes », *Ann. Télécomm.*, Tome 54, N ° 3-4, pp. 166-172, March-April 1999.

[7.14] Consultative Committee for Space Data Systems, *Recommendations for Space Data Systems. Telemetry Channel Coding*, BLUE BOOK, May 1998.

[7.15] 3GPP Technical Specification Group, *Multiplexing and Channel Coding (FDD)*, TS 25.212 v2.0.0, June 1999.

[7.16] C. Berrou and M. Jézéquel, « Frame-oriented convolutional turbo-codes », *Electronics letters*, Vol. 32, N ° 15, pp. 1362-1364, July 1996.

[7.17] C. Weiss, C. Bettstetter, S. Riedel and D.J. Costello, « Turbo decoding with tail-bitting trellises », *Proc. Signals, Systems and Electronics, URSI Int'l*

Symposium, pp. 343-348, Sept.-Oct. 1998.

[7.18] C. Weiss, C. Bettstetter and S. Riedel, « Code constuction and decoding of parallel concatenated tail-biting codes », *IEEE Trans. Info. Theory*, Vol. 47, Issue 1, pp. 366-386, Jan. 2001.

[7.19] DVB, *Interaction channel for satellite distribution systems*, ETSI EN 301 790, V1.2.2, pp. 21-24, Dec. 2000.

[7.20] DVB, *Interaction channel for digital terrestrial television*, ETSI EN 301 958, V1.1.1, pp. 28-30, Aug. 2001.

[7.21] Y. V. Svirid, « Weight distributions and bounds for turbo-codes », *European Trans. on Telecomm*, Vol. 6, N ° 5, pp. 543-55, Sept.-Oct. 1995.

[7.22] E. Boutillon and D. Gnaedig, « Maximum Spread of D-dimensional Multiple Turbocodes », *IEEE Trans. Commun.*, Vol. 53, N ° 8, pp 1237 - 1242, Aug. 2005.

[7.23] C. Berrou, C. Douillard and M. Jézéquel, « Designing turbocodes for low error rates », *IEE colloquium : Turbocodes in digital broadcasting – Could it double capacity ?*, pp. 1-7, London, Nov. 1999.

[7.24] H. R. Sadjadpour, N. J. A. Sloane, M. Salehi and G. Nebe, « Interleaver design for turbocodes », *IEEE Journal on Selected Areas in Commun.*, Vol. 19, N ° 5, pp. 831-837, May 2001.

[7.25] C. Heegard and S. B. Wicker, *Turbo coding*, Chap. 3, Kluwer Academic Publishers, 1999.

[7.26] C. Berrou, Y. Saouter, C. Douillard, S. Kerouédan and M. Jézéquel, « Designing good permutations for Turbocodes : towards a single model », *ICC'04*, Paris, France, June 2004.

[7.27] S. Crozier and P. Guinand, « Distance upper bounds and true minimum distance results for turbo-codes with DRP interleavers », *Proc. of 3rd Int'l Symposium on Turbocodes & Related Topics*, pp. 169-172, Brest, France, Sept. 2003.

[7.28] Y. Weiss and W. T. Freeman, « On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs », *IEEE Trans. Inform. Theory*, Vol. 47, Issue 2, pp. 736-744, Feb. 2001.

[7.29] L. Duan and B. Rimoldi, « The iterative turbo decoding algorithm has fixed points », *IEEE Trans. Inform. Theory*, Vol. 47, N ° 7, pp. 2993-2995, Nov. 2001.

[7.30] L.R. Bahl, J. Cocke, F. Jelinek and J. Raviv : « Optimal decoding of linear codes for minimizing symbol error rate », *IEEE Trans. Inform. Theory*, IT-20, pp. 248-287, Mar. 1974.

[7.31] P. Robertson, P. Hoeher and E. Villebrun, « Optimal and suboptimal maximum a posteriori algorithms suitable for turbo decoding », *European Trans. Telecommun.*, vol. 8, pp. 119-125, March-Apr. 1997.

[7.32] Z. Wang, Z. Chi and K. K. Parhi, « Area-efficient high-speed decoding schemes for turbo decoders », *IEEE Trans. VLSI Systems*, Vol. 10, N ° 6, pp. 902-912, Dec. 2002.

[7.33] C. Douillard and C. Berrou, « Turbo codes with rate-m/(m+1) constituent convolutional codes », *IEEE Trans. Commun.*, Vol. 53, N̊10, pp. 1630-

1638, Oct. 2005.

[7.34] IEEE Std 802.16a, *IEEE standard for local and metropolitan area networks*, 2003, available at
http ://standards.ieee.org/getieee802/download/802.16a-2003.pdf.

[7.35] C. Berrou, « Some clinical aspects of turbocodes », *Int'l Symposium on turbocodes & related topics*, pp. 26-31, Brest, France, Sept. 1997.

[7.36] C. Berrou and M. Jézéquel, « Non binary convolutional codes for turbo coding », *Elect. Letters*, Vol. 35, N ° 1, pp. 39-40, Jan. 1999.

[7.37] H. Lin and D. G. Messerschmitt, « Algorithms and architectures for concurrent Viterbi decoding », *Proc. ICC'89*, pp. 836-840, Boston, June 1989.

[7.38] C. Berrou, S. Vaton, M. Jézéquel and C. Douillard, « Computing the minimum distance of linear codes by the error impulse method », *IEEE Global Communication Conference*, Globecom'2002, Taipei, Taiwan, Nov. 2002, pp. 1017-1020.

[7.39] R. Garello and A. Vila Casado, « The all-zero iterative decoding algorithm for turbo code minimum distance computation », pp. 361-364, *ICC 2004.*, Paris, 20-24 June 2004.

[7.40] S. Crozier, P. Guinand and A. Hunt, « Computing the Minimum Distance of Turbo-Codes Using Iterative Decoding Techniques », *Proceedings of the 22nd Biennial Symposium on Communications*, Kingston, Ontario, Canada, pp. 306-308, May 31-June 3, 2004.

[7.41] R. Garello, P. Pierloni and S. Benedetto, « Computing the free distance of turbo codes and serially concatened codes with interleavers : Algorithms and Applications », *IEEE Journal on Select. Areas in Comm.*, May 2001.

[7.42] Stephan ten Brink, « Convergence behavior of iteratively decoded parallel concatenated codes », *IEEE trans. On Comm.*, vol. 49, N ° 10, Oct. 2001.

Chapitre 8

Turbocodes produits

8.1 Historique

En vertu de la borne de Gilbert-Varshamov, il est nécessaire d'avoir des codes longs pour obtenir des codes en blocs ayant une distance minimale de Hamming (DMH)importante et donc un fort pouvoir de correction. Or, sans structure particulière, il est presque impossible de décoder ces codes.

L'invention des codes produits, due à Elias [8.1], se place dans cette problématique : il s'agit d'un moyen simple d'obtenir des codes à haut pouvoir de correction aisément décodables à partir de codes élémentaires simples. Ces codes produits peuvent être vus comme une réalisation particulière du principe de concaténation (chapitre 6).

Le premier algorithme de décodage découle immédiatement de la construction de ces codes. Il s'agit de l'algorithme alternant le décodage, en décision dure, des codes élémentaires sur les lignes et les colonnes. Malheureusement cet algorithme ne permet pas d'atteindre le pouvoir de correction maximum de ces codes. L'algorithme de Reddy-Robinson [8.2] permet de l'atteindre. Mais sans doute en raison de sa complexité, il n'a jamais été implanté dans des applications pratiques.

Ce chapitre a pour but de faire une présentation assez complète des algorithmes de décodage des codes produits, que ce soit les algorithmes sur données dures ou sur données souples.

8.2 Les codes produits

Avec les constructions classiques, il est théoriquement possible de construire des codes ayant une DMH élevée. Cependant, la complexité de décodage devient prohibitive, même pour des codes possédant une structure algébrique, comme les codes de Reed-Solomon ou les codes BCH (voir chapitre 4). Par exemple, pour les codes Reed-Solomon sur GF(256), le décodeur le plus complexe ayant

été implanté sur circuit a un pouvoir de correction limité à 11 erreurs symboles. Ce qui est insuffisant pour la plupart des applications actuelles. La construction de codes produits permet de contourner ce problème : en utilisant des codes simples à faible pouvoir de correction mais dont le décodage est peu coûteux, il est possible de les assembler pour obtenir un code plus long dont le pouvoir de correction est important.

Définition :

Soit C_1 (resp. C_2) un code linéaire de longueur n_1 (resp. n_2) et de dimension[1] k_1 (resp. k_2). Le code produit $C = C_1 \otimes C_2$ est l'ensemble des matrices M de taille $n_1 \times n_2$ telles que :

- Chaque ligne est un mot de code de C_1,
- Chaque colonne est un mot de code de C_2.

Ce code est un code linéaire de longueur $n_1 \times n_2$ et de dimension $k_1 \times k_2$.

Exemple 8.1 :

Soit H le code de Hamming de longueur 7 et P le code de parité de longueur 3. La dimension de H est 4 et la dimension de P est 2. Le code $C = H \otimes P$ est donc de longueur $21 = 7 \times 3$ et de dimension $8 = 4 \times 2$. Soit à coder le mot d'information suivant :

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

Chaque ligne d'un mot de code de C doit être un mot de code de H. Donc pour coder I, on commence par multiplier chaque ligne de I par la matrice génératrice du code H :

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

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

Après le codage des lignes, le mot de code provisoire est donc :

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

[1] ou longueur du message.

Chaque colonne du mot de code final doit maintenant être un mot du code de parité P. Le mot de code final s'obtient donc en ajoutant une troisième ligne formée par les bits de parité de chaque colonne. Le mot de code complet est :

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

Pour que le mot de code soit valide, il faut encore vérifier que la troisième ligne du mot est bien un mot de code de H. Il faut donc multiplier ce vecteur ligne par la matrice de contrôle de parité de H :

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

En fait, cette vérification est inutile : elle est assurée par construction puisque les codes H et P sont linéaires. De plus, l'ordre d'encodage n'a pas d'importance : si on code d'abord par colonnes puis par lignes, le mot de code obtenu est le même.

8.3 Le décodage à entrée ferme des codes produits

8.3.1 Le décodage ligne-colonne

Le premier algorithme de décodage découle immédiatement de la construction du code : on alterne successivement le décodage des lignes par un décodeur du code C_1 et le décodage des colonnes par un décodeur du code C_2. Soit d_1 (resp. d_2) la distance minimale du code C_1 (resp. C_2). Alors, le décodage par syndrome de C_1 (resp. C_2) est t_1-correcteur (resp. t_2-correcteur) avec $t_1 = \lfloor d_1/2 \rfloor$ (resp. $t_2 = \lfloor d_2/2 \rfloor$).

Propriété :

Le décodage ligne-colonne est borné par un pouvoir de correction de $(t_1+1)\cdot(t_2+1)$ erreurs. En d'autres termes, le décodage ligne-colonne décode tout mot ayant au plus $(t_1+1)\cdot(t_2+1)$ erreurs (même s'il peut éventuellement décoder certains motifs ayant plus d'erreurs) et il existe des mots ayant exactement $(t_1+1)\cdot(t_2+1)$ erreurs qui ne seront pas décodés.

En effet, supposons que l'on ait un motif avec un nombre d'erreurs strictement inférieur à $(t_1+1)\cdot(t_2+1)$. Puisque l'algorithme de décodage en ligne

corrige jusqu'à t_1 erreurs, après la première étape de décodage, toute ligne en erreur contient au moins $t_1 + 1$ erreurs. Il y a donc au plus t_2 lignes en erreurs après le décodage ligne. Chaque colonne contient donc au plus t_2 erreurs et le décodage colonne élimine alors toutes les erreurs.

Il existe des motifs indécodables ayant exactement $(t_1+1)\cdot(t_2+1)$ erreurs. En effet, prenons un mot de code de $C_1 \otimes C_2$ dont on choisit $(t_2 + 1)$ lignes et $(t_1 + 1)$ colonnes au hasard. À chaque intersection entre une ligne et une colonne on insère une erreur dans le mot de code initial. Par construction, pour le mot ainsi obtenu, il existe bien un mot de code pour le code produit à une distance de $(t_1 + 1) \cdot (t_2 + 1)$ erreurs, mais le décodage ligne et le décodage colonne échouent.

On peut noter que le décodage ligne-colonne est moins puissant que le décodage par syndrome pour un code produit. En effet, un code produit est un code linéaire dont la distance minimale est $d_1 d_2$. Donc le décodage par syndrome permet de corriger tous les mots ayant au plus t erreurs avec $t = \lfloor (d_1 d_2)/2 \rfloor$. Or le décodage ligne-colonne permet seulement de corriger tous les mots ayant moins de $(t_1 + 1) \cdot (t_2 + 1) = (\lfloor d_1/2 \rfloor + 1)(\lfloor d_2/2 \rfloor + 1)$ erreurs. On perd donc environ un facteur 2 en pouvoir de correction.

Exemple 8.2 :

On suppose que l'on a un code produit dont le code en ligne et le code en colonne ont tous les deux une distance minimale égale à 5. Ils sont donc tous les deux 2-correcteurs. Le code produit, d'après ce qui précède, peut donc corriger tout mot ayant au plus 8 erreurs. La figure 8.1 montre un mot ayant 10 erreurs (représentées par les points) et pourtant corrigible par le décodage ligne-colonne. La figure 8.2 montre un motif ayant le même nombre d'erreurs mais non corrigible.

8.3.2 L'algorithme de Reddy-Robinson

L'algorithme de Reddy-Robinson [8.2] est un algorithme itératif plus sophistiqué qui suppose cette fois que l'on dispose pour les codes C_1 et C_2 de décodeurs avec erreurs et effacements. Un effacement est une position non fiable de la trame reçue pour laquelle on estime que le symbole peut éventuellement être erroné. La différence avec une erreur classique est que la position de l'effacement est connue au moment du décodage. Pour un code de DMH égale à d, le décodage par syndrome peut être adapté pour tenir compte des effacements et alors il est possible de décoder toute trame avec t erreurs et e effacements dès que l'on a

$$2t + e < d \tag{8.1}$$

Les algorithmes algébriques des codes BCH et Reed-Solomon peuvent aussi être adaptés pour traiter les effacements.

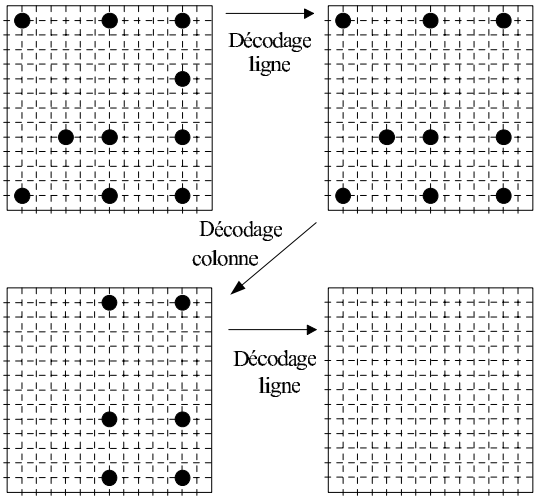


Figure 8.1 – Mot en erreur corrigible par un décodage ligne-colonne.

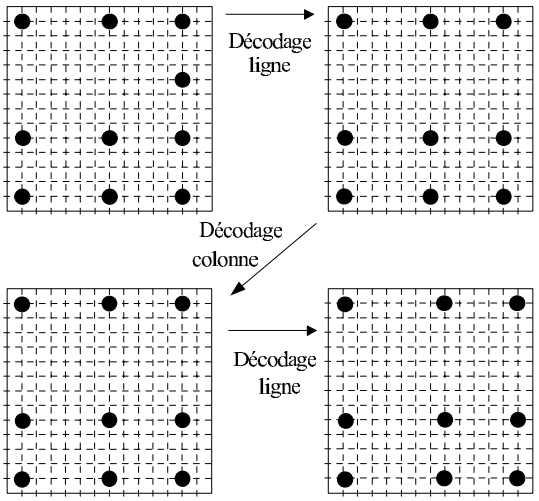


Figure 8.2 – Mot en erreur non corrigible par un décodage ligne-colonne.

L'algorithme de Reddy-Robinson est alors le suivant :

- Étape 1 : repérer les erreurs de chaque ligne i en appliquant le décodeur du code C_1 sans effacements et leur attribuer un poids l_i égal au nombre d'erreurs détectées. Si le décodage échoue, le poids attribué est $l_i = d_1/2$. Les lignes sont passées sans correction à l'étape 2.
- Étape 2 : décoder les colonnes par le décodeur du code C_2 avec effacements. Pour chaque colonne, on fait successivement tous les décodages en

effaçant les symboles les moins sûrs (*i.e.* ceux qui ont le poids de ligne le plus élevé). On a donc au plus $\lceil d_1/2 \rceil$ décodages par ligne. À chaque décodage on attribue un poids W tel que $W = \sum_{i=1}^{n} w_i$ où $w_i = l_i$ si le symbole de la ligne i est inchangé par le décodage et $w_i = \lceil d_2/2 \rceil$ sinon.

- Étape 3 : Comme décodage final, on choisit pour chaque colonne, le mot décodé donnant le poids W le plus petit.

Le décodage de Reddy-Robinson permet de corriger tout motif d'erreurs dont le poids est inférieur strictement à $d_1 d_2$ ([8.2]).

Exemple 8.3 :

Reprenons l'exemple précédent avec le mot de la figure 8.2. Au cours de la première étape, les lignes ayant 3 erreurs ne pourront pas être corrigées par le code ligne car celui-ci ne peut corriger que 2 erreurs au maximum (DMH de 5). Elles se verront donc attribuer un poids égal 2,5. La ligne avec une erreur aura un poids égal à 1, tandis que toutes les lignes restantes auront un poids égal à 0. La configuration est alors comme illustré en figure 8.3.

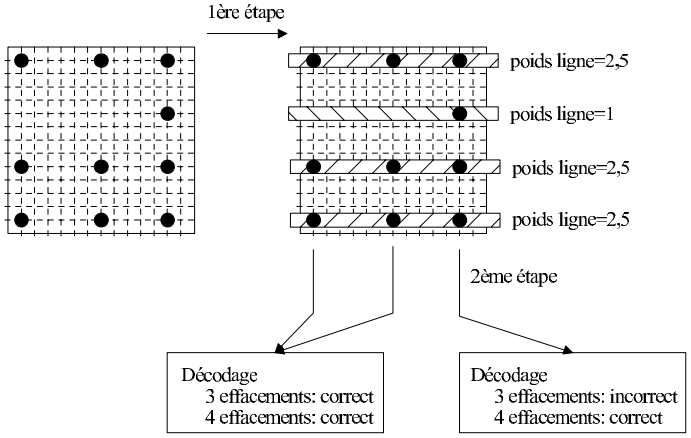


Figure 8.3 – Calcul des poids de ligne issus de la première étape de l'algorithme de Reddy-Robinson.

À la deuxième étape, la correction devient effective. Seules trois colonnes possèdent des erreurs. En ce qui concerne la colonne en erreur la plus à gauche, d'après les poids fournis par l'étape 1, trois symboles de cette colonne ont un poids égal à 2,5, un symbole a un poids égal à 1 et tous les autres ont un poids nul. La deuxième étape de l'algorithme pour cette colonne va donc consister en trois décodages successifs : un décodage sans effacement, un décodage avec trois effacements (les symboles ayant un poids égal à 2,5) et un décodage à quatre effacements (les trois symboles précédent plus le symbole ayant un poids de 1). Le premier décodage échoue puisque le code est seulement 2-correcteur. Le deuxième décodage réussit. En effet, le code colonne a une distance minimale

de 5. Il peut donc corriger t erreurs et e effacements dès que $2t + e < 5$. Or pour ce décodage, on a $e = 3$ (puisque 3 effacements sont placés) et $t = 0$ (puisqu'il n'y a pas d'erreurs supplémentaire dans la colonne). Le poids associé au second décodage est la somme des poids des symboles effacés, soit 7,5. De même le troisième décodage (avec 4 effacements) réussit et le poids associé au mot décodé est alors de 8,5. L'algorithme choisit alors parmi les décodages ayant réussi celui dont le poids est le plus faible, soit ici le second.

Les deux autres colonnes en erreur se décodent aussi. Cependant pour la colonne la plus à droite, le deuxième décodage échoue (puisque qu'il y a une erreur dans les symboles non effacés) et le mot décodé pour cette colonne est donc celui du troisième décodage. Finalement, toutes les erreurs sont corrigées.

Dans cet algorithme le rôle des lignes et des colonnes n'est pas symétrique. Ainsi, si le décodage complet échoue dans l'ordre initial, il est possible d'essayer à nouveau en intervertissant le rôle des lignes et des colonnes.

L'algorithme de décodage de Reddy-Robinson n'est pas itératif dans sa version initiale. On peut le rendre itératif, par exemple en recommençant un décodage avec le mot final de l'étape 2, si l'un des décodages de la deuxième étape a réussi. Il existe aussi des versions itératives plus sophistiquées [8.3].

8.4 Le décodage à entrée souple des codes produits

8.4.1 L'algorithme de Chase à sortie pondérée

L'algorithme de Chase permet, dans le cas d'un code en bloc, d'utiliser les valeurs reçues sur le canal de transmission pour faire un décodage au sens du maximum de vraisemblance ou, pour le moins, en approcher la performance. Dans sa version de base, il produit un décodage ferme. En suivant, l'idée du décodage des turbocodes convolutifs [8.4], la version améliorée par Pyndiah [8.5] de l'algorithme permet d'obtenir en sortie une valeur souple du bit décodé. À l'aide d'un décodage itératif, il est alors possible pour les décodeurs ligne et colonne d'un code produit de s'échanger des information extrinsèques sur les bits. On peut donc décoder les codes produits à la manière turbo.

Soit donc $r = (r_1, ..., r_n)$ le mot reçu après codage et transmission sur un canal gaussien. L'algorithme de Chase-Pyndiah à t places se décompose ainsi :

- Étape 1 : sélectionner les t places P_k de la trame contenant les symboles les moins fiables de la trame (i.e. les t places j pour lesquelles les valeurs r_j sont les plus petites en valeur absolue).
- Étape 2 : générer le vecteur des décisions fermes $h_0 = (h_{01}, ..., h_{on})$ tel que $h_{0j} = 1$ si $r_j > 0$ et 0 sinon. Générer les vecteurs $h_1, ..., h_{2^t-1}$ tels que $h_{ij} = h_{0j}$ si $j \notin \{P_k\}$ et $h_{iP_k} = h_{0P_k} \oplus Num(i,k)$ où $Num(i,k)$ est le k-ième bit dans l'écriture binaire de i.
- Étape 3 : décoder les mots $h_0, ..., h_{2^t-1}$ avec le décodeur ferme du code linéaire. On obtient alors les mots concurrents $c_0, ..., c_{2^t-1}$.

- Étape 4 : calculer les métriques des mots concurrents

$$M_i = \sum_{1 \leqslant j \leqslant n} r_j(1 - 2c_{ij})$$

- Étape 5 : déterminer l'indice pp tel que

$$M_{pp} = \min\{M_i\}$$

  Le mot de code c_{pp} est alors le mot de code le plus probable.
- Étape 6 : pour chaque bit j de la trame calculer la fiabilité

$$F_j = 1/4(\min\{M_i, c_{ij} \neq c_{pp,j}\} - M_{pp})$$

  S'il n'existe pas de concurrents pour lesquels le j-ième bit est différent de $c_{pp,j}$, alors la fiabilité F_j est à une valeur β fixée.
- Étape 7 : calculer la valeur extrinsèque pour chaque bit j,

$$E_j = (2 \times c_{pp,j} - 1) \times F_j - r_j$$

Les valeurs extrinsèques s'échangent alors entre décodeurs ligne et décodeurs colonne dans un processus itératif. La valeur β, ainsi que les valeurs des contre-réactions sont ici plus sensibles que dans le cas du décodage de turbocodes convolutifs. Des valeurs inadéquates peuvent dégrader de manière importante le pouvoir de correction. Cependant, il est possible de les déterminer de manière incrémentale. On cherche d'abord pour la première itération quelles sont les valeurs donnant les meilleures performances (par exemple, par dichotomie). Puis ces valeurs étant fixées, on procède à une recherche similaire pour la deuxième itération et ainsi de suite.

Exemple 8.4 :

Soit $r = (0,5; 0,7; -0,9; 0,2; -0,3; 0,1; 0,6)$ un échantillon reçu d'un mot de code de Hamming$(7,4,3)$ et le nombre de places de l'algorithme de Chase est égal à $t = 3$. On choisit $\beta = 0,6$. L'algorithme précédent donne alors :

– Étape 1 : $P_1 = 6, P_2 = 4, P_3 = 5$.

– Étape 2 :

I	h_i
0	(1 ;1 ;0 ;1 ;0 ;1 ;1)
1	(1 ;1 ;0 ;1 ;0 ;$\underline{0}$;1)
2	(1 ;1 ;0 ;$\underline{0}$;0 ;1 ;1)
3	(1 ;1 ;0 ;$\underline{0}$;0 ;$\underline{0}$;1)
4	(1 ;1 ;0 ;1 ;$\underline{1}$;1 ;1)
5	(1 ;1 ;0 ;1 ;$\underline{1}$;$\underline{0}$;1)
6	(1 ;1 ;0 ;$\underline{0}$;$\underline{1}$;1 ;1)
7	(1 ;1 ;0 ;$\underline{0}$;$\underline{1}$;$\underline{0}$;1)

Les bits soulignés correspondent aux inversions réalisées par l'algorithme de Chase.

– Étape 3 : Les bits avec une étoile dans la colonne des mots concurrents c_i correspondent aux places corrigées par le décodeur ferme dans le mot h_i.

I	h_i	c_i
0	(1 ;1 ;0 ;1 ;0 ;1 ;1)	(1 ;1 ;0 ;1 ;0 ;1 ;0*)
1	(1 ;1 ;0 ;1 ;0 ;$\underline{0}$;1)	(1 ;1 ;0 ;0* ;0 ;0 ;1)
2	(1 ;1 ;0 ;$\underline{0}$;0 ;1 ;1)	(1 ;1 ;0 ;0 ;0 ;0* ;1)
3	(1 ;1 ;0 ;$\underline{0}$;0 ;$\underline{0}$;1)	(1 ;1 ;0 ;0 ;0 ;0 ;1)
4	(1 ;1 ;0 ;1 ;$\underline{1}$;1 ;1)	(1 ;1 ;1* ;1 ;1 ;1 ;1)
5	(1 ;1 ;0 ;1 ;$\underline{1}$;$\underline{0}$;1)	(0* ;1 ;0 ;1 ;1 ;0 ;1)
6	(1 ;1 ;0 ;$\underline{0}$;$\underline{1}$;1 ;1)	(1 ;0* ;0 ;0 ;1 ;1 ;1)
7	(1 ;1 ;0 ;$\underline{0}$;$\underline{1}$;$\underline{0}$;1)	(1 ;1 ;0 ;0 ;0* ;0 ;1)

– Étape 4 :

I	c_i	M_i
0	(1 ;1 ;0 ;1 ;0 ;1 ;0*)	-(0.5)-(0.7)+(-0.9)-(0.2)+(-0.3)-(0.1)+(0.6)=-2,1
1	(1 ;1 ;0 ;0* ;0 ;0 ;1)	-(0.5)-(0.7)+(-0.9)+(0.2)+(-0.3)+(0.1)-(0.6)=-2,7
2	(1 ;1 ;0 ;0 ;0 ;0* ;1)	-(0.5)-(0.7)+(-0.9)+(0.2)+(-0.3)+(0.1)-(0.6)=-2,7
3	(1 ;1 ;0 ;0 ;0 ;0 ;1)	-(0.5)-(0.7)+(-0.9)+(0.2)+(-0.3)+(0.1)-(0.6)=-2,7
4	(1 ;1 ;1* ;1 ;1 ;1 ;1)	-(0.5)-(0.7)-(-0.9)-(0.2)-(-0.3)-(0.1)-(0.6)=-0,9
5	(0* ;1 ;0 ;1 ;1 ;0 ;1)	+(0.5)-(0.7)-(-0.9)-(0.2)-(-0.3)+(0.1)-(0.6)=0,3
6	(1 ;0* ;0 ;0 ;1 ;1 ;1)	-(0.5)+(0.7)+(-0.9)+(0.2)-(-0.3)-(0.1)-(0.6)=-0,9
7	(1 ;1 ;0 ;0 ;0* ;0 ;1)	-(0.5)-(0.7)+(-0.9)+(0.2)+(-0.3)+(0.1)-(0.6)=-2,7

– Étape 5 : On a $pp = 1$. Le mot décodé est $(1; 1; 0; 0; 0; 0; 1)$. Noter qu'on le rencontre plusieurs fois dans la liste des mots concurrents.
– Étapes 6 et 7 :

j	F_j	E_j
1	((0,3)-(-2,7))/4=0,75	0,75-0,5=0,25
2	((-0,9)-(-2,7))/4=0,475	0,475-0,7=-0,225
3	((-0,9)-(-2,7))/4=0,475	-0,475-(-0,9)=0,525
4	((-0,9)-(-2,7))/4=0,475	-0,475-0,2=-0,675
5	((-2,1)-(-2,7))/4=0,15	-0,15-(-0,3)=0,15
6	((-0,9)-(-2,7))/4=0,475	-0,475-0,1=-0,575
7	((-2,1)-(-2,7))/4=0,15	0,15-0,6=-0,4

8.4.2 Performances de l'algorithme de Chase-Pyndiah

L'algorithme de Chase-Pyndiah est le plus répandu des algorithmes à décodage pondérés pour les codes en blocs. La figure 8.4 donne les performances obtenues par cet algorithme dans le cadre d'un turbo-décodage pour différents codes produits.

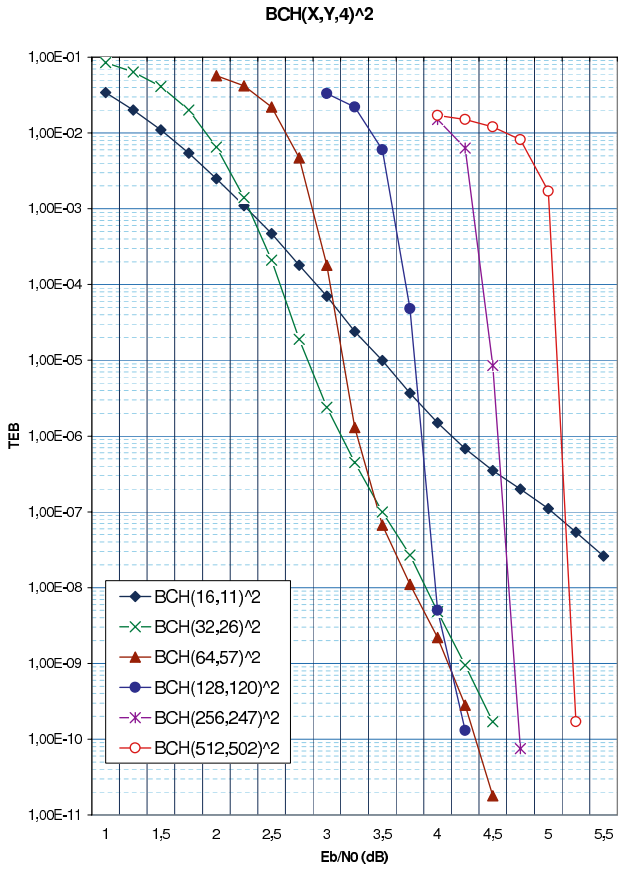


Figure 8.4 – Performances de l'algorithme de Chase Pyndiah en turbo-décodage itératif de différents codes 1-correcteurs (MDP-2 sur canal gaussien, 4 itérations).

La figure 8.5 montre l'évolution du taux d'erreurs binaire pendant le turbo-décodage du code BCH étendu, BCH$(64,57,4)^2$, avec l'algorithme de Chase-Pyndiah. Pour ce code, on voit que l'essentiel du gain en correction est obtenu au cours des 4 premières itérations.

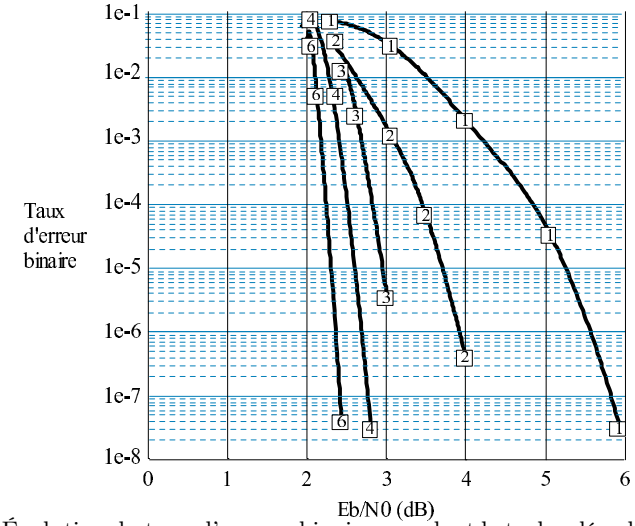


Figure 8.5 – Évolution du taux d'erreurs binaires pendant le turbo-décodage de Chase-Pyndiah pour 1, 2, 3, 4 et 6 itérations (code BCH(64,51,6), MDP-2 sur canal Gaussien).

8.4.3 L'algorithme de Fang et Battail

L'algorithme de Fang et Battail permet, comme l'algorithme de Chase-Pyndiah, de calculer la fiabilité des bits décodés, à partir de données souples échantillonnées en sortie du canal de transmission. On peut voir cet algorithme comme une variante de l'algorithme de Chase-Pyndiah pour laquelle seuls les bits les moins fiables sont modifiés. Soit $r = (r_1, ..., r_n)$ le mot reçu. Le code utilisé est un code linéaire de longueur n et de dimension k de matrice génératrice $G = \{g_{ij}, 0 \leqslant i < k, 0 \leqslant j < n\}$ et de matrice de contrôle de parité $H = \{h_{ij}, 0 \leqslant i < n-k, 0 \leqslant j < n\}$. L'algorithme procède par les étapes suivantes :

- Étape 1 : trier les valeurs r_i dans l'ordre des valeurs absolues décroissantes. On obtient alors une permutation P sur l'ensemble des indices $[1..n]$. On pose $r'_l = r_{P(l)}$ et on a donc $|r'_1| \geqslant |r'_2| \geqslant ... \geqslant |r'_n|$.
- Étape 2 : soit H^* la matrice obtenue en permutant les colonnes de H par P. Systématiser les k colonnes les plus à droite de H^* par réduction de Gauss. Soit H' la matrice obtenue.
- Étape 3 : soit s' le vecteur des décisions dures de r'. Soit M_1 la métrique cumulée associée aux $n-k$ premières valeurs de r'. Soit

$$s'' = H' \begin{bmatrix} s'_1 & s'_2 & ... & s'_{n-k} & 0 & 0 & ... & 0 \end{bmatrix}$$

  Soit M_2 la métrique cumulée associée à s'' dans les k dernières valeurs de r'. La métrique cumulée totale est alors $M = M_1 + M_2$.
- Étape 4 : énumérer la liste des concurrents dans l'ordre des métriques M_1 croissantes (on utilise un algorithme de génération de combinaison avec

exclusion). S'arrêter dès que la métrique totale M commence à croître.
- Étape 5 : après permutation inverse des métriques, calculer les valeurs extrinsèques à partir des concurrents en utilisant les mêmes formules que dans l'algorithme de Chase-Pyndiah.

Exemple 8.5 :

On considère le même exemple que pour l'algorithme de Chase.

Soit $r = (0,5; 0,7; -0,9; 0,2; -0,3; 0,1; 0,6)$ un échantillon reçu. La matrice de contrôle de parité du code de Hamming de dimension 4 et de longueur 7 est, comme nous l'avons déjà vu,

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

– Étape 1 : on a $r' = (-0,9; 0,7; 0,6; 0,5; -0,3; 0,2; 0,1)$ et $P = [3, 2, 7, 1, 5, 4, 6]$.
– Étape 2 :

$$H^* = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Après systématisation sur les trois dernières colonnes, on obtient :

$$H' = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

– Étape 3 : la décision dure sur les $n - k = 4$ valeurs reçues les plus fortes donne $s' = (0, 1, 1, 1)$. La matrice H' sans les k dernières colonnes, permet de retrouver les valeurs de redondance manquantes :

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

Le vecteur décodé initial est donc $(0, 1, 1, 1, 0, 0, 0)$ qui donne après permutation inverse le vecteur $(1, 1, 0, 0, 0, 0, 1)$. La métrique initiale est $M = M_1 + M_2 = -2,7$ car $M_1 = (-0,9) - 0,7 - 0,6 - 0,5 = -2,7$ et $M_2 = (-0,3) + 0,2 + 0,1 = 0,0$.
– Étape 4 : pour énumérer les concurrents, on applique des masques d'inversion aux $n - k$ premier bits (on peut avoir de 1 à 4 inversions au maximum). Chaque masque d'inversion va accroître la partie M_1 de la métrique. Pour une inversion, le bonus est au minimum de 1,0 et au maximum de 1,8. Le bonus minimum pour deux inversions est au minimum de $2 \times (0,6 + 0,5) = 2,2$. Les premiers concurrents à considérer sont donc tous ceux correspondant à une seule inversion. De plus, les modifications

sur la partie M_2 de la métrique peuvent éventuellement la diminuer. Cependant la diminution ne peut excéder $2 \times (0,2+0,1) = 0,6$ par rapport à la métrique initiale. On sait donc, d'ores et déjà, que le mot précédemment trouvé est le mot le plus vraisemblable. Si on décide tout de même d'énumérer, on trouve les mots concurrents suivants, qui contrairement à l'algorithme de Chase-Pyndiah, sont tous différents les uns des autres :

s_i	C'_i	M
(0,1,1,1)	(0,1,1,1,0,0,0)	-2,7
(0,1,1,0)	(0,1,1,0,1,1,0)	-2,7+1,0+0,2=-1,5
(0,1,0,1)	(0,1,0,1,0,1,1)	-2,7+1,2-0,6=-2,1
(0,0,1,1)	(0,0,1,1,1,0,1)	-2,7+1,4+0,2=-1,1
(1,1,1,1)	(1,1,1,1,1,1,1)	-2,7+1,8+0,0=-0,9
(0,1,0,0)	(0,1,0,0,1,0,1)	-2,7+2,2+0,2=-0,3
(0,0,1,0)	(0,0,1,0,0,1,1)	-2,7+2,4-0,6=-0,9
(0,0,0,1)	(0,0,0,1,1,1,0)	-2,7+2,6+0,2=0,1
(1,1,1,0)	(1,1,1,0,0,0,1)	-2,7+2,8-0,2=-0,1
(1,1,0,1)	(1,1,0,1,1,0,0)	-2,7+3,0+0,6=0,9
(1,0,1,1)	(1,0,1,1,0,1,0)	-2,7+3,2-0,2=0,3
(0,0,0,0)	(0,0,0,0,0,0,0)	-2,7+3,6+0,0=0,9
(1,1,0,0)	(1,1,0,0,0,1,0)	-2,7+4,0-0,2=1,1
(1,0,1,0)	(1,0,1,0,1,0,0)	-2,7+4,2+0,6=2,1
(1,0,0,1)	(1,0,0,1,0,0,1)	-2,7+4,4-0,2=1,5
(1,0,0,0)	(1,0,0,0,1,1,1)	-2,7+5,4+0,0=2,7

– Étape 5 :

j	$F_{P(j)}$
1	((-0,9)-(-2,7))/4=0,475
2	((-1,1)-(-2,7))/4=0,4
3	((-2,1)-(-2,7))/4=0,15
4	((-1,5)-(-2,7))/4=0,3
5	((-1,5)-(-2,7))/4=0,3
6	((-1,5)-(-2,7))/4=0,3
7	((-2,1)-(-2,7))/4=0,15

J	E_J
1	0,3-0,5=-0,2
2	0,4-0,7=-0,3
3	-0,475-(-0,9)=0,525
4	-0,3-0,2=-0,5
5	-0,3-(-0,3)=0,0
6	-0,15-0,1=-0,25
7	0,15-0,6=-0,4

L'algorithme de Fang-Battail est théoriquement à maximum de vraisemblance. Cependant, cela ne suffit pas nécessairement pour que l'algorithme appliqué au décodage des codes produits soit lui-même à maximum de vraisemblance.

8.4.4 L'algorithme de Hartmann-Nazarov

Hartmann *et al* [8.6] ont utilisé les propriétés de dualité pour décrire une méthode de décodage à maximum de vraisemblance pour les codes linéaires. Initialement, seule une décision ferme était fournie par l'algorithme. Hagenauer [8.7] a alors repris ces idées pour que les valeurs extrinsèques nécessaires au turbo-décodage soient disponibles. Enfin Nazarov *et al* [8.8] ont montré comment réduire le coût en complexité par l'utilisation de la transformée de Hadamard rapide. Ce paragraphe fait la synthèse de ces trois articles de référence.

Soit $r = (r_1, ..., r_n)$ le mot reçu. Le code utilisé est un code linéaire de longueur n et de dimension k de matrice génératrice $G = \{g_{ij}, 0 \leqslant i < k, 0 \leqslant j < n\}$ et de matrice de contrôle de parité $H = \{h_{ij}, 0 \leqslant i < n-k, 0 \leqslant j < n\}$. On suppose que la transmission a été faite sur un canal perturbé par un bruit gaussien de moyenne nulle et de variance égale à σ^2.

En posant $\rho_l = -\tanh(r_l/\sigma^2)$, on montre que le LLR à maximum de vraisemblance du l-ième bit de la trame est :

$$LLR_l = \ln\left(\frac{1 - C(l)}{1 + C(l)}\right) \tag{8.2}$$

avec :

$$C(l) = \frac{\rho_l}{2}\left(1 + \frac{F_D(h_l)}{F_D(0)}\right) + \frac{1}{2\rho_l}\left(1 - \frac{F_D(h_l)}{F_D(0)}\right)$$

où h_l est la l-ième colonne de la matrice de contrôle de parité et la fonction F_D est telle que

$$F_D(h_l) = \sum_{\nu=0}^{2^{n-k}-1} D_\nu(l) \exp\left\{j\pi \sum_{m=0}^{n-k-1} \nu_m h_{ml}\right\}$$

ν_m est le m-ième bit de la représentation binaire de l'entier ν ($\nu = \sum_{m=0}^{n-k-1} \nu_m 2^m$), et :

$$D_\nu(l) = \prod_{l=0}^{n-1} (\rho_l)^{t_\nu(l)}$$

$t_\nu(l)$ étant le l-ième bit du ν-ième vecteur du dual du code, soit donc :

$$t_\nu(l) = \left(\sum_{m=0}^{n-k-1} \nu_m h_{ml}\right) \bmod 2 = \langle \nu, h_l \rangle \bmod 2$$

Le calcul de $D_\nu(l)$ nécessite normalement n multiplications. Mais en passant par la base du code dual et en appliquant une transformée rapide de Hadamard, il est possible de ramener ce coût a un terme de l'ordre de $n-k$ multiplications. Si le rendement du code est élevé, alors n est bien supérieur à $n-k$ et le gain en terme de complexité de calcul est important.

Pour ce faire on ré-écrit le terme général en facteur dans $D_\nu(l)$ sous la forme :

$$(\rho_l)^{t_\nu(l)} = \exp\{t_\nu(l)\ln|\rho_l|\}\exp\{j\pi q_l t_\nu(l)\}$$

où q_l est tel que $\rho_l = (-1)^{q_l}|\rho_l|$.
On a alors :

$$D_\nu(l) = \exp\left\{\sum_{l=0}^{n-1}(t_\nu(l)\ln|\rho_l|) + (j\pi q_l t_\nu(l))\right\}$$

$$= \exp\left\{\sum_{l=0}^{n-1}(t_\nu(l)\ln|\rho_l|)\right\}\exp\left\{\sum_{l=0}^{n-1}(j\pi q_l t_\nu(l))\right\}$$

Posons $F_\rho(w) = \sum_{l=0}^{n-1}\ln(|\rho_l|)\exp\left\{j\pi\sum_{m=0}^{n-k-1}w_m h_{ml}\right\}$ pour tout entier $0 \leqslant w \leqslant 2^{n-k}-1$. On a donc :

$$F_\rho(w) = \sum_{l=0}^{n-1}\ln(|\rho_l|)\exp\{j\pi t_w(l)\}$$

avec en particulier, $F_\rho(0) = \sum_{l=0}^{n-1}\ln(|\rho_l|)$.

D'autre part, si $t=0$ ou 1, alors $\frac{1-\exp\{j\pi t\}}{2} = t$ et $\frac{F_\rho(0)-F_\rho(\nu)}{2} = \sum_{l=0}^{n-1}(t_\nu(l)\ln|\rho_l|)$.

De même, si on pose $F_q(w) = \sum_{l=0}^{n-1} q_l \exp\left\{j\pi\sum_{m=0}^{n-k-1}w_m h_{ml}\right\}$, on a :

$$\frac{F_q(0)-F_q(\nu)}{2} = \sum_{l=0}^{n-1}(q_l\ln|\rho_l|)$$

et donc :

$$D_\nu(l) = \exp\left(\frac{1}{2}(F_\rho(0)-F_\rho(\nu))\right)\exp\left(\frac{1}{2}j\pi(F_q(0)-F_q(\nu))\right)$$

Les deux termes $F_\rho(\nu)$et $F_q(\nu)$ ont une expression commune de la forme :

$$F(w) = \sum_{l=0}^{n-1} f_l \exp\left\{j\pi\sum_{m=0}^{n-k-1}w_m h_{ml}\right\}$$

où f_l sont des réels qui ne dépendent que de l.

On définit la fonction g sur l'ensemble $\left\{0, \cdots, 2^{n-k}-1\right\}$ à valeurs réelle par :

$$g(p) = \begin{cases} f_l & \text{si } \exists l, \quad p = \sum_{m=0}^{n-k-1} h_{ml} 2^m \\ 0 & \text{sinon} \end{cases}$$

La fonction g est bien définie car les colonnes de H sont linéairement indépendantes et donc *a fortiori* deux à deux distinctes. La transformée de Hadamard de G est alors une fonction à valeurs réelles définie sur l'intervalle $\left[0..2^{n-k}-1\right]$ par :

$$G(w) = \sum_{p=0}^{2^{n-k}-1} g(p)(-1)^{<p,w>}$$

Or la fonction g est nulle sauf pour les points $p_l = \sum_{l=0}^{n-k-1} h_{ml} 2^m$ pour $l \in [0, \cdots, n-1]$. Il vient alors :

$$G(w) = \sum_{l=0}^{n-1} f_l (-1)^{<\sum_{m=0}^{n-k-1} h_{ml} 2^m, w>} = \sum_{l=0}^{n-1} f_l \exp\left(j\pi \sum_{m=0}^{n-k-1} w_m h_{ml}\right) = F(w)$$

Les deux termes $F_\rho(\nu)$et $F_q(\nu)$ s'expriment donc comme des transformées de Hadamard et peuvent être calculés au moyen de la transformée rapide de Hadamard.

Transformée rapide de Hadamard

Soit $R = [R_0, R_1, ..., R_{2^n-1}]$ un vecteur à composantes réelles. Le vecteur obtenu à partir de R par transformée de Hadamard est le vecteur $\hat{R} = [\hat{R}_0, \hat{R}_1, \cdots, \hat{R}_{2^n-1}]$ tel que :

$$\hat{R}_j = \sum_{i=0}^{2^n-1} R_i (-1)^{<i,j>} \tag{8.3}$$

Le produit scalaire $< i, j >$ est comme précédemment le produit scalaire bit à bit des développements binaires de i et j. On écrit aussi sous forme vectorielle, $\hat{R} = RH_{2^n}$ où H_{2^n} est la matrice de Hadamard d'ordre 2^n dont le coefficient $(H_{2^n})_{i,j} = (-1)^{<i,j>}$.

Soit A une matrice de taille $a_1 \times a_2$ et B une matrice de taille $b_1 \times b_2$ à coefficients réels. Alors le produit de Kronecker de A par B, noté $A \otimes B$, est une matrice de taille $(a_1 b_1) \times (a_2 b_2)$ telle que $(A \otimes B)_{i,j} = A_{q_1 q_2} B_{r_1 r_2}$ où $i = b_1 q_1 + r_1$ et $j = b_2 q_2 + r_2$ avec $0 \leqslant r_1 < b_1$ et $0 \leqslant r_2 < b_2$.

Si $N = 2^n$, on montre que ([8.9]) :

$$H_N = (H_2 \otimes I_{N/2})(I_2 \otimes H_{N/2})$$

où I_k est la matrice unité d'ordre k.
En développant de manière récursive, on obtient :

$$H_N = \prod_{i=1}^{n} \left(I_{2^{i-1}} \otimes H_2 \otimes I_{N/2^i}\right)$$

La transformée rapide de Hadamard est en fait l'utilisation de cette factorisation pour le calcul de $\hat{R}$.

Exemple 8.6 :

Soit à calculer la transformée de Hadamard du vecteur :

$$R = [0,2; 0,5; -0,7; 1,3; 0,1; -1,1; 0,8; -0,3]$$

On a $\hat{R} = RH_8$ avec :

$$H_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

Le calcul direct donne :

$$R = [0,8; 0,0; -1,4; 1,8; 1,8; -4,6; 1,6; 1,6]$$

Or d'après ce qui précède, $H_8 = G_1G_2G_3$ où $G_i = I_i \otimes H_2 \otimes I_{8/2^i}$, on a :

$$G_1 = [1] \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix}$$

$$G_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$

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

$$G_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

On peut alors calculer $\hat{R}$ par trois multiplications matricielles :

$$\begin{aligned} \hat{R} &= [0,8;0,0;-1,4;1,8;-1,0;-1,8;0,0;3,2] \cdot G_1 \cdot G_2 \cdot G_3 \\ &= [0,8;0,0;-1,4;1,8;1,8;-4,6;1,6;1,6] \end{aligned}$$

Les matrices G_i sont des matrices creuses ayant seulement deux éléments non nuls par colonnes. De plus la factorisation de la matrice de Hadamard H_N est de longueur proportionnelle à $\log(N)$. Le coût total du calcul est donc en $N\log(N)$ par la transformée rapide, au lieu de N^2 par la méthode directe. La figure 8.6 représente le graphe des calculs pour la transformée rapide de Hadamard dans le cas $N = 8$. En termes de performance de correction d'erreurs, l'article [8.10] montre que l'on obtient les mêmes performances que l'algorithme de Chase-Pyndiah pour deux fois moins d'itérations.

8.4.5 Autres algorithmes à entrée souple

Il existe encore d'autres algorithmes de décodage des codes en blocs à entrée et sortie pondérées. L'un des plus ancien est l'algorithme de décodage par optimisation locale de Farrell *et al* [8.11]. Cette technique consiste à écrire le problème de la recherche du mot le plus probable comme un problème de

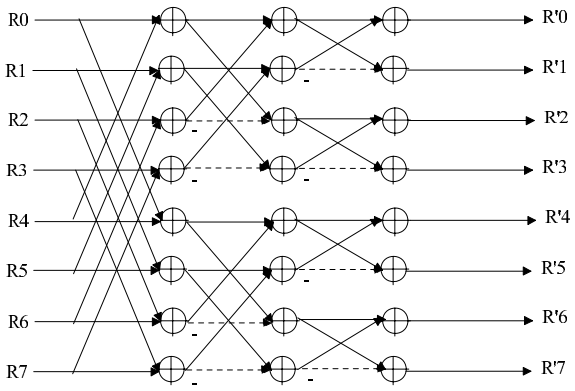


Figure 8.6 – Graphe du flot de calcul pour la transformée rapide de Hadamard $N = 8$.

minimisation d'une fonction de coût global ayant pour variables les bits d'information du mot. Cette optimisation doit se faire théoriquement en nombre entier ce qui rend le problème très difficile. La technique utilisée est de remplacer ces variables entières par des variables réelles. Le problème n'est alors plus strictement équivalent au problème initial mais il devient possible d'utiliser des techniques classiques d'optimisation non linéaires comme la méthode du gradient pour rechercher le minimum de la fonction de coût.

Une autre approche introduite par Kschischang *et al* [8.12] consiste à utiliser le fait que la plupart des codes en blocs utilisés dans les turbocodes produits sont en fait des codes en treillis. Il est alors possible d'utiliser les algorithmes classiques (MAP, Max-Log-MAP, ...) de décodage des codes en treillis pour obtenir les décisions souples du décodeur. Ce type d'algorithme est une application au cas des turbocodes produits des algorithmes de décodage à maximum de vraisemblance de Wolf [8.13].

Un autre algorithme plus récemment inventé est celui de Kötter *et al* [8.14]. Cet algorithme n'est applicable qu'à des codes très spécifiques dont essentiellement les codes de Reed-Solomon et les codes algébriques. Il est basé sur l'algorithme de Sudan [8.15] qui est capable de déterminer les mots de codes dans un voisinage proche du mot reçu. Il est alors possible d'utiliser les techniques de pondérations classiques. Si la version initiale de cet algorithme est relativement coûteuse en calcul, des améliorations théoriques ont été apportées et cet algorithme est d'autant plus prometteur que les codes de Reed-Solomon sont largement utilisés dans le domaine des télécommunications. Il existe aussi des algorithmes de décodage à base de sous-codes [8.16]. Bien qu'un peu complexe au niveau de l'implémentation, ces algorithmes donnent d'excellentes performances. Enfin, des travaux récents ont montré que les algorithmes de décodage à base de propagation de croyance peuvent être appliqués aux codes linéaires en général [8.17].

8.5 Implantation de l'algorithme de Chase-Pyndiah

L'algorithme de Fang-Battail requiert la systématisation de la matrice de parité qui est une opération très coûteuse. L'algorithme de Hartmann-Nazarov, quant à lui, a été implanté sur *DSP* [8.10] mais la précision nécessaire au calcul des métriques est trop importante pour pouvoir envisager une réalisation du décodeur sur un *ASIC* de taille raisonnable. C'est pourquoi, l'algorithme de Chase-Pyndiah est l'algorithme de décodage à sortie pondérée le plus couramment utilisé pour une réalisation sur circuit dédié [8.18]. D'autant plus que par une judicieuse utilisation des mémoires, il permet d'élaborer des architectures de turbocodes produits adaptées aux transmissions haut-débit [8.19].

Le turbo-décodage selon l'algorithme de Chase-Pyndiah alterne décodages pondérés des lignes et des colonnes. Ce processus itératif aboutit à l'architecture de la figure 8.7. Entre chaque demi-itération s'insère une phase de reconstruction de la matrice afin d'obtenir le mot décodé.

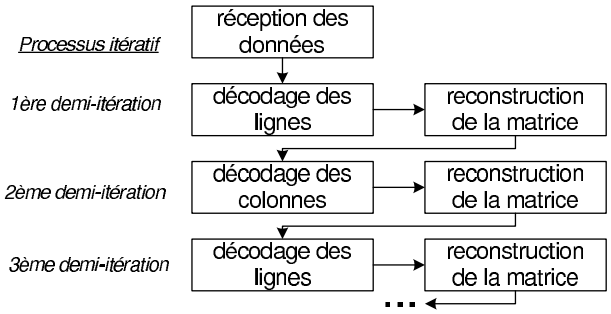


Figure 8.7 – Architecture globale d'un turbo-décodeur produit.

Chaque processeur de demi-itération prend en entrée les données canal ainsi que les valeurs extrinsèques produites par la demi-itération précédente. En sortie, le processeur reporte les données du canal (pour assurer un fonctionnement *pipeline* de la chaîne de décodage), les décisions fermes du mot de code le plus probable et les valeurs extrinsèque calculées par l'algorithme de Chase-Pyndiah. L'architecture du processeur est illustrée en figure 8.8. Les *FIFO* (*Fist-In/First-Out*) servent à synchroniser les données entrantes et sortantes du décodeur SISO qui possède une certaine latence. Si les *FIFO* de petite taille sont le plus souvent réalisées par des lignes de bascules D, pour des raisons d'encombrement matériel et de consommation, il devient assez vite intéressant d'utiliser une mémoire RAM et deux pointeurs (un pour l'écriture et un pour la lecture), incrémentés à chaque nouvelle donnée et dont l'adressage est géré circulairement. Une remarque peut être faite aussi en ce qui concerne la multiplication des valeurs extrinsèques par α. Dans une implémentation classique, les valeurs W_k sont généralement des nombres entiers de taille réduite (5 ou

6 bits) mais le coût matériel d'un vrai multiplieur est prohibitif, et on préfère généralement lui substituer une simple table.

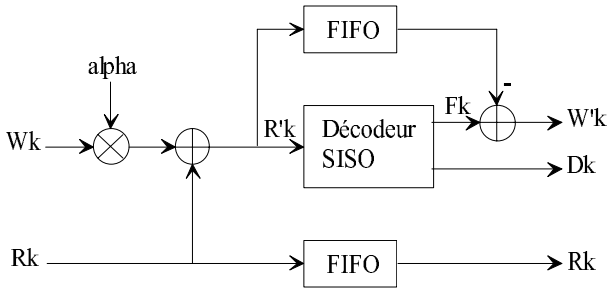


Figure 8.8 – Architecture interne du processeur de demi-itération.

Le décodeur *SISO*, décrit par la figure 8.9, réalise les étapes de l'algorithme de Chase-Pyndiah. Le décodeur est formé de cinq parties :

- *Le module de traitement séquentiel des données* calcule de manière parallèle le syndrome du mot de code entrant et les positions les moins fiables de la trame.
- *Le module de décodage algébrique* effectue le décodage algébrique des mots construits à partir des données $R'_{k'}$ entrantes et de la connaissance des places les moins fiables.
- *Le module de sélection* détermine le mot de code le plus probable ainsi que le ou les mots concurrents les plus proches.
- *Le module de calcul des pondérations* détermine la fiabilité des bits décodés.
- *Le module mémoire* stocke les pondérations totales entrantes qui sont utilisées pour le calcul des pondérations.

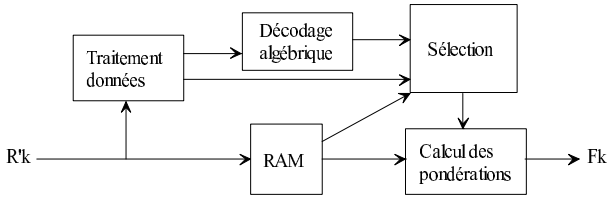


Figure 8.9 – Architecture interne du décodeur SISO.

Le module de traitement des données reçoit les bits d'échantillons les uns après les autres. Si le code est cyclique (BCH, par exemple) le calcul du syndrome se fait alors de manière très simple en utilisant la factorisation du polynôme générateur suivant le schéma de Hörner. La détermination des positions les moins fiables se fait souvent en gérant de manière séquentielle la liste des positions peu fiables dans une petite RAM locale. Il existe aussi d'autre solu-

tions plus économiques en taille comme la pile systolique de Leiserson, mais le gain obtenu est faible.

Le module de décodage algébrique utilise la valeur du syndrome pour déterminer les places en erreur dans les vecteurs concurrents. Dans le cas des codes BCH, il est possible d'utiliser l'algorithme de Berlekamp-Massey ou l'algorithme d'Euclide étendu pour faire la correction. Il faut quand même noter que cette solution n'est vraiment économique que pour des codes en blocs à fort pouvoir de correction. Pour les codes à faible pouvoir de correction, il est moins coûteux de stocker, dans une ROM locale, pour chaque valeur possible du syndrome, les bits à corriger.

Le module de sélection doit faire le tri entre les mots générés par le module de décodage algébrique pour déterminer ceux qui sont les plus probables. Il doit donc calculer la métrique de chacun de ces mots (ce qu'il fait de manière séquentielle par des additions) et déterminer par minimum puis mémoriser les plus probables d'entre eux (leur nombre est généralement limité par souci de place).

Enfin le module de calcul des pondérations utilise la liste des mots concurrents précédemment retenu pour générer les pondérations à partir de la formule de l'étape 6 de l'algorithme de Chase-Pyndiah. Ce module est de faible complexité car les calculs à réaliser sont relativement simples et pour chaque bit, il ne doit conserver séquentiellement que deux valeurs (la plus petite métrique parmi les mots concurrents ayant 0 comme valeur correspondante pour ce bit dans son développement binaire et la plus petite métrique pour les mots candidats avec 1 comme valeur binaire). Ce module contient en outre la valeur β. Dans le cas d'une implantation sur *FPGA* (*Field Programmable Gate Array*), toutes les itérations sont généralement exécutées sur le même matériel qui est réutilisé de demi-itération en demi-itération. Il faut donc prévoir une procédure de chargement de la valeur β.

8.6 Bibliographie

[8.1] P. Elias,« Error-free coding », *IEEE Transactions on Information Theory*, vol. IT-4, 1954, pp. 29-37.

[8.2] S.M. Reddy and J.P. Robinson, « Random error and burst corrections by iterated codes », *IEEE Transactions on Information Theory*, vol. 18, N^o 1, 1972, pp. 182-185.

[8.3] N. Sendrier, *Codes correcteurs à haut pouvoir de correction*, Thèse de l'Université Paris VI, 1991.

[8.4] C. Berrou , A. Glavieux and P. Thitimajshima, « Near Shannon limit error correcting coding and decoding : Turbo-codes », *Intl. Conf. on Communications*, vol. 2, 1993, Geneva, Switzerland, pp. 1064-1070.

[8.5] R. Pyndiah, A. Glavieux , A. Picart and S. Jacq , « Near optimum decoding of product codes », *IEEE GLOBECOM'94*, San Francisco, 1994.

[8.6] C.R.P. Hartmann and L.D. Rudolph , « An optimum symbol-by-symbol

decoding rule for linear codes », *IEEE Transactions on Information Theory*, vol. IT-22, 1974, pp. 514-517.

[8.7] J. Hagenauer, E. Offer and L. Papke , « Iterative decoding of binary block and convolutional codes », *IEEE Transactions on Information Theory*, vol. IT-42, 1996, pp. 429-445.

[8.8] L.E. Nazarov and V.M. Smolyaninov, « Use of fast Walsh-Hadamard transformation for optimal symbol-by-symbol binary block-code decoding », *Electronics Letters*, vol. 34, 1998, pp. 261-262.

[8.9] M. Lee and M. Kaveh, « Fast Hadamard transform based on a simple matrix factorization », *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, 1986, pp. 1666- 1667.

[8.10] A. Goalic , K. Cavalec-Amis and V. Kerbaol , « Real-Time Turbo Decoding of Block Turbo Codes using the Hartmann-Nazarov algorithm on the DSP TEXAS TMS320C6201 », *Intl. Conf. on Communications*, New-York, 2002.

[8.11] K. Farrell , L. Rudolph , C. Hartmann and L. Nielsen , « Decoding by local optimisation », *IEEE Trans. Info. Theory*, vol. 29, N ° 5, pp 740-743, Sept. 1983.

[8.12] V. Sorokine , F.R. Kschischang and V. Durand , « Trellis based decoding of binary linear block codes », *Lecture Notes in Computer Science*, vol. 793, Springer-Verlag Publisher, 1994, pp. 270-286.

[8.13] J. Wolf , « Efficient maximum likelihood decoding of linear block codes using a trellis »,*IEEE Transactions on Information Theory*, vol. 24, 1978, pp. 76-80.

[8.14] R. Kotter and A. Vardy , « Algebraic soft-decision decoding of Reed-Solomon codes », *IEEE International Symposium on Information Theory*, 2000, p. 61.

[8.15] M. Sudan , « Decoding of Reed-Solomon codes beyond the error correction bound », *Journal of Complexity*, vol. 12, 1997, pp. 180-193.

[8.16] C.Y. Liu and S. Lin , « Turbo encoding and decoding of Reed-Solomon codes through binary decomposition and self-concatenation »,*IEEE Transactions on Communications*,vol. 52, issue 9, Sept. 2004, pp. 1484 – 1493.

[8.17] A. Kothiyal , O.Y. Takeshita , « A comparison of adaptative belief propagation and the best graph algorithm for the decoding of linear block codes », *IEEE International Symposium on Information Theory*, 2005, pp. 724-728.

[8.18] P. Adde, R. Pyndiah and O. Raoul, « Performance and complexity of block turbo decoder circuits », *Third International Conference on Electronics, Circuits and System* ICECS'96, Rodos, Greece, 1996, pp. 172-175.

[8.19] J. Cuevas, P. Adde and S. Kerouedan, « Turbo decoding of product codes for Gigabit per second applications and beyond », *European Tansactions on Telecommunications*, vol.17, N ° 1, Jan. 2006, pp. 45-55.

Chapitre 9

Codes *LDPC*

Les codes *LDPC* (*Low Density Parity Check codes* pour codes à faible densité) forment une classe de codes en bloc qui se caractérisent par une matrice de contrôle creuse. Ils ont été décrits pour la première fois dans la thèse de Gallager au début des années 60 [9.1]. Outre le décodage à entrée ferme des codes *LDPC*, cette thèse proposait déjà un décodage itératif basé sur la propagation de croyance (en anglais *BP* pour *Belief Propagation*). Ces travaux ont été oubliés pendant 30 ans. Seules quelques rares études y font référence durant cette période de sommeil, notamment, celle de Tanner qui proposa une généralisation des codes de Gallager et une représentation par graphe bipartite [9.2].

Après l'invention des turbocodes, les codes *LDPC* furent redécouverts au milieu des années 90 par MacKay *et al* [9.3], Wilberg [9.4] et Sipser *et al* [9.5]. Depuis, des progrès considérables sur les règles de construction de bons codes *LDPC*, sur les techniques d'encodage et de décodage, ont permis aux codes *LDPC* d'être utilisés, tout comme les turbocodes, dans des applications pratiques.

Le principe de fonctionnement des codes *LDPC* et de leur décodage font l'objet de ce chapitre qui considère également les réalisations matérielles.

9.1 Principe des codes *LDPC*

Les codes *LDPC* sont des codes construits à partir du code élémentaire le plus simple : le code de parité. Nous commencerons donc ce chapitre en détaillant le code de parité et son décodage à entrée et sortie souples avant de poursuivre la construction des codes *LDPC*.

9.1.1 Code de parité

Définition

Une équation de parité, représentée graphiquement par la figure 9.1, est une équation reliant n données binaires entre elles par l'opérateur *ou exclusif*, noté $\oplus$. Elle est vérifiée si le nombre total de 1 dans l'équation est pair ou nul.

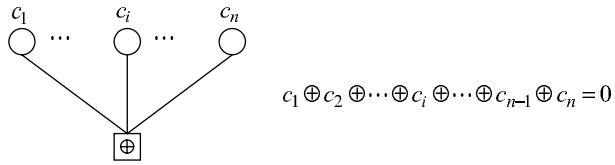


Figure 9.1 – Représentation graphique d'une équation de parité.

Les cercles représentent les données binaires c_i, aussi appelées *variables*. Le rectangle contenant l'opérateur *ou exclusif* représente l'équation de parité (appelée aussi contrainte de parité ou parité). Les liaisons entre les variables et l'opérateur indiquent les variables impliquées dans l'équation de parité.

Code de parité à trois bits

On considère que les variables binaires c_1, c_2 et c_3 sont liées par la contrainte de parité $c_1 \oplus c_2 \oplus c_3 = 0$, et qu'elles forment le mot de code (c_1, c_2, c_3). On suppose que l'on connaît le logarithme du rapport de vraisemblance (LRV) $L(c_1)$ et $L(c_2)$ des variables c_1 et c_2 : que peut-on dire alors du LRV $L(c_3)$ de la variable c_3 ?

On rappelle que $L(c_j)$ est défini par l'équation :

$$L\left(c_j\right) = \ln\left(\frac{\Pr\left(c_j = 0\right)}{\Pr\left(c_j = 1\right)}\right) \tag{9.1}$$

Il y a deux mots de code dans lequel le bit c_3 est égal à 0 : les mots de code (0,0,0) et (1,1,0). De même, il existe deux mots de code dans lequel le bit c_3 est égal à 1 : les mots de code (1,0,1) et (0,1,1). On en déduit les deux équations suivantes dans le domaine des probabilités :

$$\begin{cases} \Pr(c_3 = 1) = \Pr(c_1 = 1) \times \Pr(c_2 = 0) + \Pr(c_1 = 0) \times \Pr(c_2 = 1) \\ \Pr(c_3 = 0) = \Pr(c_1 = 0) \times \Pr(c_2 = 0) + \Pr(c_1 = 1) \times \Pr(c_2 = 1) \end{cases} \tag{9.2}$$

En utilisant l'expression de chaque probabilité en fonction du rapport de vraisemblance, déduite de l'équation (9.1) :

$$\begin{cases} \Pr(c_j = 1) = \frac{\exp(L(c_j))}{1+\exp(L(c_j))} \\ \\ \Pr(c_j = 0) = 1 - \Pr(c_j = 1) = \frac{1}{1+\exp(L(c_j))} \end{cases}$$

il vient :

$$L(c_3) = \ln\left[\frac{1+\exp(L(c_2)+L(c_1))}{\exp(L(c_2))+\exp(L(c_1))}\right] \triangleq L(c_1) \oplus L(c_2) \tag{9.3}$$

L'équation (9.3) permet de définir l'opérateur commutatif $\oplus$ entre les deux LRV des variables c_1 et c_2.

En appliquant la fonction $\tanh(x/2) = \frac{\exp(x)-1}{\exp(x)+1}$ à l'équation (9.3), celle-ci devient :

$$\begin{aligned}\tanh\left(\tfrac{L(c_3)}{2}\right) &= \tfrac{\exp(L(c_0))-1}{\exp(L(c_0))+1} \times \tfrac{\exp(L(c_1))-1}{\exp(L(c_1))+1} \\ &= \prod_{j=0}^{1} \tanh\left(\tfrac{L(c_j)}{2}\right)\end{aligned} \tag{9.4}$$

Il est pratique (et fréquent) de séparer les traitements du signe et du module dans l'équation (9.4)qui peut alors être remplacée par les deux d'équations suivantes :

$$\operatorname{sgn}(L(c_3)) = \prod_{j=0}^{1} \operatorname{sgn}(L(c_j)) \tag{9.5}$$

$$\tanh\left(\frac{|L(c_3)|}{2}\right) = \prod_{j=0}^{1} \tanh\left(\frac{|L(c_j)|}{2}\right) \tag{9.6}$$

La fonction $\operatorname{sgn}(x)$ est telle que $\operatorname{sgn}(x) = +1$ si $x \geq 0$ et $\operatorname{sgn}(x) = -1$ sinon.

Le traitement du module donné par l'équation (9.6) peut être simplifié en prenant l'inverse du logarithme de chacun des membres de l'équation. Ce qui donne :

$$|L(c_3)| = f^{-1}\left(\sum_{j=1,2} f(|L(c_j)|)\right) \tag{9.7}$$

où la fonction f, vérifiant $f^{-1}(x) = f(x)$, est définie par :

$$f(x) = \ln(\tanh(x/2)) \tag{9.8}$$

Ces différents aspects du calcul de la fonction $\oplus$ seront développés dans la partie architecture de ce chapitre.

L'expression (9.6) correspond en fait au calcul de (9.3) dans le domaine de Fourier. Enfin, il existe aussi une troisième écriture du LRV de la variable c_2 [9.6,9.7] :

$$\begin{aligned}L(c_3) = \operatorname{sign}(L(c_1))\operatorname{sign}(L(c_2))\min(|L(c_1)|, |L(c_2)|) \\ -\ln(1+\exp(-|L(c_1)-L(c_2)|)) \\ +\ln(1+\exp(-|L(c_1)+L(c_2)|))\end{aligned} \tag{9.9}$$

Cette autre expression de l'opérateur $\oplus$ permet d'exprimer sous forme de table, la fonction g définie par

$$g(x) = \ln(1+\exp(-|x|)) \tag{9.10}$$

Exemple pratique
Supposons que $\Pr(c_1 = 1) = 0,8$ et $\Pr(c_2 = 1) = 0,1$. On a alors,

$$\Pr(c_1 = 0) = 0,2 \quad \text{et} \quad \Pr(c_2 = 0) = 0,9$$

Il est donc plus probable que $c_1 = 1$ et $c_2 = 0$. Une application directe de la formule (9.2) donne alors

$$\Pr(c_3 = 0) = 0,26 \quad \text{et} \quad \Pr(c_3 = 1) = 0,74$$

c_3 est donc plus probablement égal à 1, ce qui intuitivement est justifié puisque le nombre de 1 arrivant à la parité doit être pair. L'utilisation de la formule (9.3) donne

$$L(c_3) = L(c_1) \oplus L(c_2) = (-1,386) \oplus (2,197) = -1,045$$

soit de nouveau $\Pr(c_3 = 0) = 0,26$. On retrouve le même résultat en utilisant (9.7) et (9.9).

Code de parité à n bits

Le cas d'une équation de parité à n bits peut maintenant être traité. On considère que les variables binaires $c_1, \cdots, c_n$ sont liées par la contrainte de parité $c_1 \oplus \cdots \oplus c_n = 0$ et qu'elles forment le mot de code $(c_1, \cdots, c_n)$. Le LRV des variables $\{c_j\}_{j=1..n, j\neq i}$ est supposé connu et on cherche celui de la variable c_i. Il est alors simple de généraliser les équations obtenues pour le code de parité à 3 bits. Ainsi, en reprenant l'opérateur défini par (9.2) :

$$L(c_i) = L(c_1) \oplus L(c_2) \oplus \cdots \oplus L(c_{j\neq i}) \oplus \cdots \oplus L(c_n) = \bigoplus_{j\neq i} L(c_j) \tag{9.11}$$

De même, la règle de la tangente hyperbolique s'exprime par :

$$\tanh\left(\frac{L(c_i)}{2}\right) = \prod_{j\neq i} \tanh\left(\frac{L(c_j)}{2}\right) \tag{9.12}$$

ou bien en séparant le signe et le module :

$$\operatorname{sgn}\left(L(c_i)\right) = \prod_{j\neq i} \operatorname{sgn}\left(L(c_j)\right) \tag{9.13}$$

$$|L(c_i)| = f^{-1}\left(\sum_{j\neq i} f\left(|L(c_j)|\right)\right) \tag{9.14}$$

où f est définie par l'équation (9.8).

9.1.2 Définition d'un code *LDPC*

Codes en bloc linéaires

Les codes en bloc linéaires (voir aussi chapitre 4) peuvent être définis par une matrice de contrôle $\mathbf{H}$ de taille $m \times n$, où $m = n - k$. Cette matrice peut être vue comme un système linéaire de m équations de parité. Les mots $\mathbf{c}$ du code défini par $\mathbf{H}$ sont les mots binaires dont les n bits vérifient simultanément les m équations de parité. Ce système d'équations linéaires est représenté graphiquement dans la figure 9.2 pour le cas du code en bloc binaire de Hamming de taille $n = 7$.

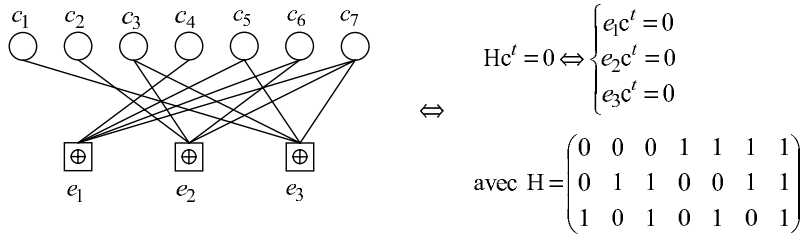


Figure 9.2 – Représentation graphique d'un code en bloc : exemple du code de Hamming de taille 7.

Une telle représentation est appelée *graphe bipartite* du code. Dans ce graphe, des branches sont reliées à deux nœuds appartenant à deux classes différentes :

- La première classe de nœuds appelés *nœuds de variable*, correspond aux bits des mots de code ($c_j,\ j \in \{1, \cdots, n\}$), et donc aux colonnes de $\mathbf{H}$.
- La seconde classe de nœuds, appelés *nœuds de parité*, correspond aux équations de parité ($e_p,\ p \in \{1, \cdots, m\}$), et donc aux lignes de $\mathbf{H}$.

Ainsi, à chaque branche reliant un nœud de variable c_j à un nœud de parité e_p correspond le 1 qui se situe à l'intersection de la j-ième colonne et de la p-ième ligne de la matrice de contrôle.

Par convention, on notera $P(j)$ (respectivement $J(p)$) l'ensemble des indices des nœuds de parité (respectivement de variable) connectés à la variable d'indice j (respectivement à la parité d'indice p). On désignera par $P(j)\backslash p$ (respectivement $J(p)\backslash j$) l'ensemble $P(j)$ privé de l'indice p (respectivement, l'ensemble $J(p)$ privé de l'indice j). Ainsi, sur l'exemple de la figure 9.2, nous avons

$$P(5) = \{1, 3\} \quad et \quad J(1)\backslash 5 = \{4, 6, 7\}$$

Un *cycle* sur un graphe bipartite est un chemin sur le graphe qui permet de partir d'un nœud et de revenir à ce même nœud sans passer par la même branche. La taille d'un cycle est donnée par le nombre de branches contenues dans le cycle. Le graphe étant bipartite, la taille des cycles est paire. En anglais, la taille du cycle le plus court dans un graphe est appelée *girth*. La présence

de cycles dans le graphe pourra dégrader les performances de décodage par un phénomène d'auto-confirmation lors de la propagation des messages. La figure 9.3 illustre deux cycles de tailles 4 et 6.

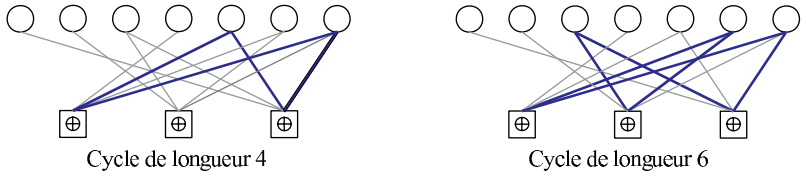


Figure 9.3 – Cycles dans un graphe bipartite.

Codes à faible densité

Les codes à faible densité (*LDPC*) sont des codes en blocs linéaires, le terme faible densité venant du fait que la matrice de contrôle $\mathbf{H}$ contient un faible nombre de valeurs non nulles : c'est une matrice *creuse*. Dans le cas particulier des codes *LDPC* binaires, étudiés ici, la matrice de contrôle contient un faible nombre de 1. Autrement dit, le graphe bipartite associé contient un faible nombre de branches. Le qualificatif « faible » signifie mathématiquement que lorsque la longueur n d'un mot de code augmente, le nombre de 1 dans la matrice augmente en $O(n)$ (contre une augmentation en $O(n^2)$ du nombre d'éléments de la matrice si le rendement reste fixe).

La classe des codes *LDPC* engendre un très grand nombre de codes. Il est commode de les distinguer en deux sous-classes :

– les codes *LDPC* réguliers
– les codes *LDPC* irréguliers

Un code *LDPC* est dit régulier dans le cas particulier où la matrice de contrôle $\mathbf{H}$ contient un nombre constant d_c de 1 dans chaque ligne, et un nombre constant d_v de 1 dans chaque colonne. On dit alors que les variables sont de degré d_v et que les parités sont de degré d_c. Le code est noté code *LDPC* régulier (d_v,d_c). Par exemple, la matrice de contrôle d'un code *LDPC* régulier (3,6) contient seulement 3 valeurs non nulles dans chaque colonne, et 6 valeurs non nulles dans chaque ligne. La figure 9.4 présente un exemple de code régulier (3,6) de taille $n = 256$ obtenu par tirage aléatoire. Sur les 256x128 entrées de la matrice, seules 3x256 sont non nulles, soit environ $2,3\%$. Ce pourcentage tend vers 0 si la taille du code, pour un rendement fixé, tend vers l'infini.

Le profil d'irrégularité des variables d'un code *LDPC* irrégulier est défini par le polynôme $\lambda(x) = \sum \lambda_j x^{j-1}$ où le coefficient λ_j est égal au rapport entre le nombre cumulé de 1 des colonnes (ou variable) de degré j et le nombre total E de 1 de la matrice $\mathbf{H}$. Par exemple, $\lambda(x) = 0,2x^4 + 0,8x^3$ indique un code où 20% des 1 sont associés à des variables de degré 5 et 80% à des variables de degré 4. Notons que, par définition, $\lambda(1) = \sum \lambda_j = 1$. De plus, la proportion

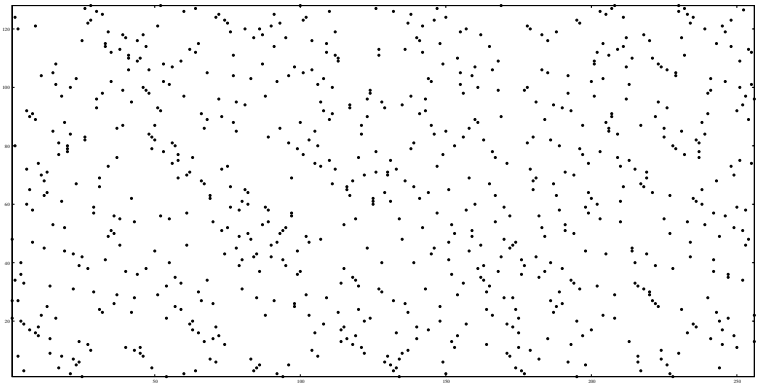


Figure 9.4 – Matrice de contrôle d'un code *LDPC* régulier (3,6) de taille $n = 256$ et de rendement $R = 0,5$.

de colonnes de degré j de la matrice est donnée par

$$\bar{\lambda}_j = \frac{\lambda_j / j}{\sum_k \lambda_k / k}$$

De façon symétrique, le profil d'irrégularité des parités est représenté par le polynôme $\rho(x) = \sum \rho_p x^{p-1}$, avec le coefficient ρ_p égal au rapport entre le nombre cumulé de 1 des lignes (ou parité) de degré p et le nombre total de 1 noté E. De même, on obtient $\rho(1) = \sum \rho_j = 1$. La proportion $\bar{\rho}_p$ de colonnes de degré p de la matrice $\mathbf{H}$ est donnée par

$$\bar{\rho}_p = \frac{\rho_p / p}{\sum_k \rho_k / k}$$

Les codes irréguliers possèdent plus de degrés de liberté que les codes réguliers et il est ainsi possible de les optimiser plus efficacement : leurs performances asymptotiques sont meilleures que celles des codes réguliers.

Rendement du code

Considérons une équation de parité de degré d_c. Il est possible de fixer arbitrairement les valeurs des $d_c - 1$ premiers bits, seul le dernier bit est contraint et correspond à de la redondance. Ainsi, dans une matrice de parité $\mathbf{H}$ de taille (m,n), chacune des m lignes correspond à 1 bit de redondance. Si les m lignes de $\mathbf{H}$ sont indépendantes, le code possède alors m bits de redondance. Le nombre total de bits du code étant de n, le nombre de bits d'information est alors $k = n - m$ et le rendement du code est $R = (n - m)/n = 1 - m/n$. Notons que dans le cas où les m lignes ne sont pas indépendantes (par exemple deux lignes identiques), le nombre de bits contraints est inférieur à m. On a alors $R > 1 - m/n$.

Dans le cas d'un code *LDPC* régulier (d_v,d_c), chacune des m lignes possède d_c valeurs non nulles, soit un total de $E = md_c$ valeurs non nulles dans la matrice $\mathbf{H}$. De façon symétrique, chacune des n colonnes contient d_v valeurs non nulles. On en déduit que E vérifie $E = nd_v = md_c$, soit $m/n = d_v/d_c$. Le rendement d'un tel code vérifie alors $R \geqslant (1 - d_v/d_c)$.

Dans le cas d'un code irrégulier, l'expression du rendement se généralise en tenant compte de chaque degré pondéré par son poids :

$$R \geqslant 1 - \frac{\sum_p \rho_p/p}{\sum_j \lambda_j/j} \tag{9.15}$$

L'égalité est atteinte si la matrice $\mathbf{H}$ est de rang m.

9.1.3 Encodage

L'encodage d'un code *LDPC* peut se révéler relativement complexe si la matrice $\mathbf{H}$ n'a pas de structure particulière. Il existe des solutions génériques d'encodage, dont un algorithme de complexité en $O(n)$, nécessitant un prétraitement complexe sur la matrice $\mathbf{H}$. Une autre solution consiste à construire directement la matrice $\mathbf{H}$ de façon à obtenir un code systématique très simple à encoder. C'est notamment cette solution qui a été adoptée pour le code du standard DVB-S2 de transmission numérique de télévision par satellite.

Encodage générique

Encodage par matrice génératrice

Les codes *LDPC* étant des codes linéaires en bloc, l'encodage peut se faire par le biais de la matrice génératrice $\mathbf{G}$ de taille $k \times n$ du code, telle que définie dans le chapitre 4. Comme nous l'avons vu, les codes *LDPC* se définissent à partir de leur matrice de contrôle $\mathbf{H}$, qui n'est généralement pas systématique. Une transformation de $\mathbf{H}$ en matrice systématique $\mathbf{H}_{sys}$ est possible, par exemple avec l'algorithme d'élimination gaussienne. Cette technique, relativement simple, a cependant un inconvénient majeur : la matrice génératrice $\mathbf{G}_{sys}$ du code systématique n'est généralement pas creuse. La complexité d'encodage augmente rapidement en $\mathrm{O}\left(n^2\right)$, ce qui rend cette opération trop complexe pour des codes de taille usuelle.

Encodage à complexité linéaire

Richardson *et al* [9.8] ont proposé une solution permettant un encodage quasi-linéaire (complexité en $\mathrm{O}\left(n\right)$), ainsi que des algorithmes que l'on pourrait traduire littéralement de « gourmands » (en anglais *greedy*) permettant d'effectuer un prétraitement de la matrice de contrôle $\mathbf{H}$. Le but du prétraitement est de mettre $\mathbf{H}$ sous une forme presque triangulaire inférieure, comme illustrée dans la figure 9.5, en utilisant uniquement des permutations de lignes ou de colonnes. Cette matrice est composée de 6 sous-matrices, toujours creuses, notées *A,B,C,D,E* et d'une matrice T sous matrice triangulaire inférieure. Une

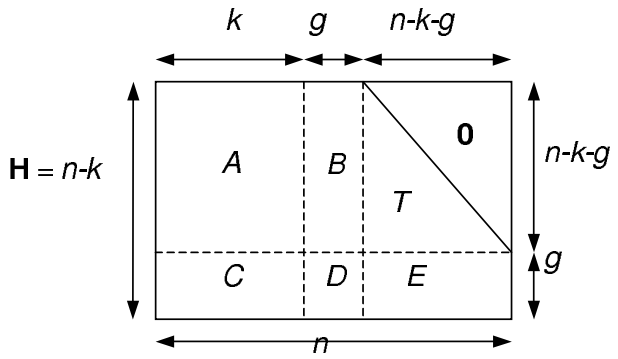


Figure 9.5 – Représentation sous forme pseudo triangulaire inférieure de la matrice de contrôle H.

fois le prétraitement de $\mathbf{H}$ achevé, le principe de l'encodage est basé sur la résolution du système représenté par l'équation matricielle suivante :

$$\mathbf{c}\mathbf{H}^T = 0 \tag{9.16}$$

Le mot de code recherché est décomposé en trois parties : $\mathbf{c} = (\mathbf{d}, \mathbf{r}_1, \mathbf{r}_2)$, où $\mathbf{d}$ est la partie systématique qui est connue et où les bits de redondance cherchés sont séparés en deux vecteurs $\mathbf{r}_1$ et $\mathbf{r}_2$, de tailles respectives g et $n-k-g$. Après multiplication à droite par la matrice $\begin{pmatrix} I & 0 \\ -ET^{-1} & I \end{pmatrix}$, l'équation (9.16) devient :

$$A\mathbf{d}^t + B\mathbf{r}_\mathbf{1}^t + T\mathbf{r}_\mathbf{2}^t = 0 \tag{9.17}$$

$$\left(-ET^{-1}A + C\right)\mathbf{d}^t + \left(-ET^{-1}B + D\right)\mathbf{r}_\mathbf{1}^t = 0 \tag{9.18}$$

L'équation (9.18) permet de trouver $\mathbf{r}_1$ en inversant $\Phi = -ET^{-1}B + D$. Puis l'équation (9.17) permet de trouver $\mathbf{r}_2$ en inversant T. De nombreuses opérations coûteuses en temps peuvent être faites une fois pour toute dans un prétraitement. Toutes les opérations répétées au cours de l'encodage ont une complexité en $\mathrm{O}(n)$ sauf la multiplication de $\left(-ET^{-1}A + C\right)\mathbf{d}^t$ par la matrice carrée $\left(-\Phi^{-1}\right)$ de taille $g \times g$ qui après inversion n'est plus creuse d'où une complexité en $\mathrm{O}\left(g^2\right)$. Il est montré dans [9.8] que l'on peut obtenir une valeur de g égale à une faible fraction de n : $g = \alpha n$ où α est un coefficient suffisamment faible pour que $\mathrm{O}\left(g^2\right) << \mathrm{O}(n)$ pour des valeurs de n allant jusqu'à 10^5.

Constructions spécifiques

Encodage par matrice génératrice creuse

Une idée proposée par Oenning *et al* [9.9] consiste à construire directement une matrice génératrice creuse et systématique, ainsi l'encodage est réalisé par simple multiplication et la matrice de contrôle reste creuse. Ces codes sont

appelés *LDGM* (*Low-Density Generator-Matrix*). Leurs performances sont cependant médiocres [9.10], même s'il est possible d'optimiser leur construction [9.11] et de diminuer le plancher d'erreur.

Encodage par résolution du système $cH^T = 0$ obtenue par substitution

Mackay *et al.* [9.12] proposent de contraindre la matrice de parité pour qu'elle soit composée des trois sous-matrices A, B et C disposées comme dans la figure 9.6.

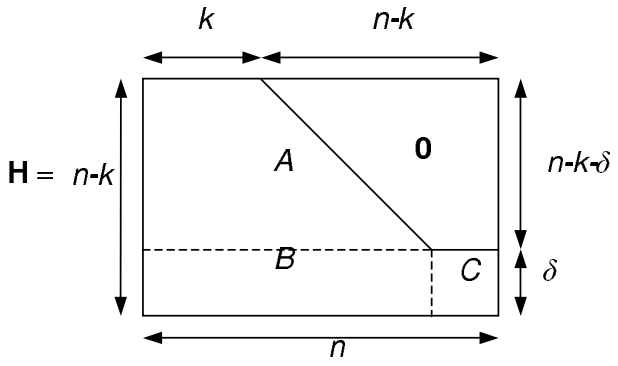


Figure 9.6 – Construction spécifique de la matrice de contrôle H facilitant l'encodage.

L'encodage systématique est réalisé par résolution de l'équation (9.16), qui se traduit par la résolution par substitution de $(n - k - \delta)$ équations. Chaque ligne de la matrice de parité contenant un faible nombre de 1, cette opération est linéaire avec n. Les δ équations restantes sont résolues par inversion de la matrice C définie dans la figure 9.4. Cela entraîne la multiplication par une matrice non creuse et donc une complexité en $\mathrm{O}\left(\delta^2\right)$. Bond *et al* [9.13] ainsi que Hu *et al* [9.14] ont proposé de construire des matrices de contrôle avec $\delta = 0$. Dans [9.15] Haley *et al* définissent une classe de codes permettant la résolution de l'équation 9.12 par un algorithme itératif analogue à celui utilisé pour le décodage.

Encodage cyclique

Les classes de codes *LDPC* définies par la géométrie finie ou par la géométrie projective [9.14, 9.16-9.19] permettent d'obtenir des codes cycliques ou pseudo cycliques. Les codes ainsi obtenus peuvent être encodés de façon efficace à l'aide de registres à décalages. Ils offrent en outre de bonnes propriétés en terme de distribution de longueur de cycle ($\geqslant 6$). L'inconvénient principal est que le cardinal de ces classes de code est relativement petit. Ces classes n'offrent donc qu'un nombre très restreint de combinaisons *taille – rendement – profil d'irrégularité* possibles.

Récapitulatif

La table 9.1 récapitule les différents types d'encodage possibles rencontrés dans la littérature. En pratique, l'encodage classique des codes en blocs par matrice génératrice n'est pas utilisé pour les codes *LDPC* en raison de la taille importante des mots de codes. Les codes obtenus par géométrie projective ou finie ne peuvent pas être optimisés (conception optimale des profils d'irrégularité). Il ne reste donc que les codes construits pour faciliter l'encodage par résolution de l'équation $cH^T = 0$ par substitution, tel que celui choisi pour la norme DVB-S2.

Type d'encodage		Complexité	Remarques
Générique	Matrice Génératrice	$\sim \mathrm{O}(n^2)$	Non utilisé en pratique
	Encodage pseudo-linéaire	$\sim \mathrm{O}(n)$	Prétraitement important
Construction ad hoc	Résolution de $\mathbf{cH}^T = 0$ par substitution	$\sim \mathrm{O}(n)$ (si $\delta = 0$)	Perte de performance possible
	Cyclique ou pseudo cyclique	$\sim \mathrm{O}(n)$	Nombre restreint de combinaisons possibles des différents paramètres

Table 9.1 – Récapitulatif des différents encodages possibles.

Analogie entre un code *LDPC* et un turbocode

La figure 9.7 donne la représentation sous forme d'un graphe bipartite proposé par Tanner [9.2] d'un turbocode, montrant ainsi la relation profonde qui lie la famille des turbocodes et celle des codes *LDPC*.

Dans le cas d'un turbocode, les contraintes sont plus grandes que dans le cas d'un code *LDPC* car les codes élémentaires sont des code convolutifs. Mais de la même façon que pour les codes *LDPC*, un mot est un mot du code si et seulement si les deux contraintes du graphe sont respectées. Notons ici que le degré des bits d'un turbocode est de deux pour les bits d'information et de un pour les bits de redondance.

De façon similaire, un code produit peut aussi se représenter par un graphe bipartite. Le nombre de nœuds de contraintes est alors de $2\sqrt{n}$ (contre 2 pour le turbocode et $n/2$ pour un code *LDPC* de rendement 0,5) et ceux-ci ont une complexité intermédiaire entre celui du turbocode et celui du code *LDPC*. Les turbocodes et les codes *LDPC* représentent ainsi les deux extrémités du spectre des codes « composés ». Le premier ne contient que deux nœuds de contraintes très complexes, le dernier, une multitude de nœud de contraintes, chaque nœud étant constitué du code linéaire le plus simple possible (code de parité). Notons

qu'à partir de cette représentation du graphe bipartite, une infinité de codes plus ou moins exotiques peuvent être construits.

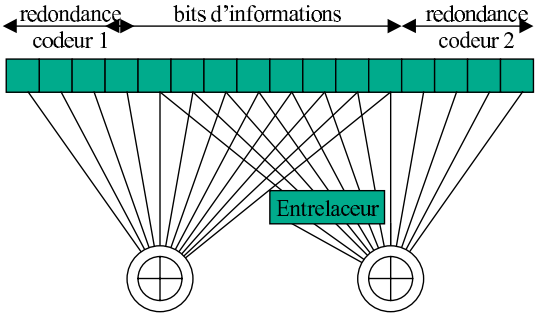


Figure 9.7 – Représentation d'un turbocode sous forme d'un graphe bipartite.

La similitude entre turbocodes et codes *LDPC* est encore plus profonde que ne le laisse supposer les représentations sous forme de graphes bipartites. En effet, il est montré dans [9.10], [9.20] et dans la section 6.2 qu'il est possible de représenter un turbocode sous forme d'une matrice *LDPC*. La ressemblance s'arrête ici. En effet, la matrice de contrôle **H** d'un turbocode contient de nombreux motifs rectangulaires (quatre 1 formant un rectangle dans la matrice **H**), c'est-à-dire de nombreux cycles de longueur 4, qui rendent inefficaces les algorithmes de décodage des codes *LDPC* qui seront décrits ci-après.

9.1.4 Décodage des codes *LDPC*

Le décodage d'un code *LDPC* s'effectue selon le même principe que le décodage d'un turbocode par un algorithme itératif dit algorithme à propagation de croyance. Chaque nœud de variable envoie aux nœuds de parité auxquels il est associé un message sur la valeur estimée de la variable (information *a priori*). L'ensemble des message *a priori* reçus permet à la contrainte de parité de calculer, puis de retourner les informations extrinsèques. Le traitement successif des nœuds de variable puis de parité constitue une itération. À chaque itération, il y a donc un échange bilatéral de messages entre les nœuds de parité et les nœuds de variable, sur les arcs du graphe bipartite représentant le code *LDPC*.

Au niveau du récepteur, la méthode de quantification de la séquence reçue, **X**, détermine le choix de l'algorithme de décodage.

Algorithme à entrée ferme

Une quantification sur un bit consiste à traiter seulement le signe des échantillons reçus. Les algorithmes de décodage à entrée ferme sont basés sur celui proposé par Gallager sous le nom d'algorithme A [9.1]. Ces décodeurs offrent bien sûr des performances moindres que celles des décodeurs à entrée souple.

Il ne sont mis en œuvre que pour des applications très particulières comme les communications par fibre optique par exemple [9.21]. Ces algorithmes ne seront pas considérés dans la suite du chapitre.

Algorithme à propagation de croyance

Lorsque la quantification est faite sur plus d'un bit, le décodage est à entrée souple et utilise la probabilité *a priori* des symboles reçus. Dans le cas des codes binaires et en se plaçant dans le domaine logarithmique, on utilise le logarithme du rapport de vraisemblance (LRV) *a priori* des échantillons X_j :

$$L\left(X_j\,|c_j\right) = \ln\left(\frac{p\left(X_j\,|c_j=0\right)}{p\left(X_j\,|c_j=1\right)}\right) \tag{9.19}$$

où c_j est le j-ième bit du mot de code et $X_j = c_j + b_j$. Dans le cas du canal à bruit blanc additif gaussien, les échantillons de bruit b_j suivent une loi gaussienne centrée de variance σ^2, soit :

$$p\left(X_j\,|c_j\right) \quad = \quad \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[\frac{\left(X_j-(2c_j-1)\right)^2}{2\sigma^2}\right] \tag{9.20}$$

En combinant (9.19) et (9.20), l'information intrinsèque I_j peut être définie :

$$I_j \triangleq L\left(X_j\,|c_j\right) = -\frac{2X_j}{\sigma^2} \tag{9.21}$$

Chaque itération de l'algorithme *BP* est décomposée en deux étapes :

1. Le traitement des parités :

$$L_{j,p} = 2\tanh^{-1}\left[\prod_{p'\in P(j)/p}\tanh\frac{Z_{j,p'}}{2}\right] \tag{9.22}$$

2. Le traitement des variables :

$$L_{j,p} = I_j + \sum_{j'\in J(p)/p} Z_{j',p} \tag{9.23}$$

Les itérations sont répétées jusqu'à ce que le nombre d'itérations maximum N_{it} soit atteint. Il est possible d'arrêter les itérations avant N_{it} lorsque toutes les équations de parités sont satisfaites. Cela permet soit de gagner en débit moyen, soit de limiter la consommation.

On appelle L_j l'information totale ou le LRV du bit j. C'est la somme de l'information intrinsèque I_i et de l'information extrinsèque totale Z_j qui est par définition la somme des informations extrinsèques de branches $Z_{j,p}$:

$$Z_j \triangleq \sum_{p\in P(j)} Z_{j,p} \tag{9.24}$$

On a donc $L_j = I_j + Z_j$ et l'équation (9.23) peut alors s'écrire :

$$L_{j,p} = L_j - Z_{j,p} = I_j + Z_j - Z_{j,p} \tag{9.25}$$

L'algorithme *BP* est optimal dans le cas où le graphe du code ne contient aucun cycle : tous les séquencements[1] (en anglais *schedules*) donnent le même résultat. Les codes *LDPC* comportant des cycles, leur décodage par l'algorithme *BP* peut entraîner des phénomènes d'auto-confirmation des messages qui dégradent la convergence et rendent l'algorithme *BP* nettement sous-optimal. Toutefois, ces phénomènes peuvent être limités si les cycles sont suffisamment grands.

Le premier séquencement qui fut proposé est appelé « séquencement par inondation » (en anglais *flooding schedule*) [9.22]. Il consiste à traiter successivement toutes les parités puis toutes les variables.

Algorithme de séquencement par inondation

Initialisation :

1- $n_{it} = 0$, $Z_{j,p}^{(0)} = 0 \quad \forall p \quad \forall j \in J(p)$, $I_j = 2y_j/\sigma^2 \quad \forall j$

Répéter jusqu'à ce que $n_{it} = N_{it}$ ou que le système ait convergé vers un mot de code :

2- $n_{it} = n_{it} + 1$

3- $\forall\, j \in \{1, \cdots, n\}$ faire : {Calcul des messages variable vers parité}

4- $Z_j^{(n_{it})} = \sum_{p \in P(j)} Z_{j,p}^{(n_{it}-1)}$ et $L_j^{(n_{it})} = I_j + Z_j^{(n_{it})}$

5- $\forall\, p \in P(j)$:

$$L_{j,p}^{(n_{it})} = I_j + \sum_{p' \in P(j)/p} Z_{j,p'}^{(n_{it}-1)} = I_j + Z_j^{(n_{it})} - Z_{j,p}^{(n_{it}-1)}$$

6- $\forall\, p \in \{1, \cdots, m\}$ faire : {Calcul du message parité vers variable}

7- $\forall\, j \in J(p) : Z_{j,p}^{(n_{it})} = \bigoplus_{p' \in P(j)/p} L_{j,p'}^{(n_{it})}$

Les bits décodés sont alors estimés par $\operatorname{sgn}\left(-L_j^{(n_{it})}\right)$.

Il est intéressant de noter qu'il est possible de modifier l'algorithme en « ordonnant » le séquencement par inondation suivant les noeuds de parité. Ceux-ci sont alors traités en série, et l'algorithme devient :

3'. $\forall j \in \{1, \cdots, n\} : Z_j^{(n_{it}+1)} = 0$

4'. $\forall p \in \{1, \cdots, m\}$ faire :

5'. calcul des messages d'entrées

$$\forall j \in J(p) \quad L_{j,p}^{(n_{it})} = I_j + Z_j^{(n_{it})} - Z_{j,p}^{(n_{it}-1)}$$

[1] On entend par séquencement l'ordre dans lequel est effectué le traitement de chaque parité et de chaque variable.

6'. calcul des informations extrinsèques

$$\forall\, j \in J(p) \quad Z_{j,p}^{(n_{it})} = \bigoplus_{p' \in P(j)/p} L_{j,p'}^{(n_{it})}$$

7'. mise à jour pour l'itération suivante

$$\forall\, j \in J(p) \quad Z_{j}^{(n_{it}+1)} = Z_{j}^{(n_{it}+1)} + Z_{j,p}^{(n_{it})}$$

Une organisation similaire des calculs pour les nœuds de variable sera appelée « calcul distribué » car les calculs liés à un nœud de variable seront distribués pendant une itération. Dans la section 9.2, les différents types de séquencement seront détaillés puis généralisés.

Il faut aussi noter que la notion d'itération (le calcul de tous les messages du graphe en une et une seule fois) n'est pas stricte. Ainsi, Mao *et al* [9.23] ont proposé une variante du séquencement par inondation afin de limiter l'impact de l'effet des cycles sur la convergence. Cette variante appelée « séquencement probabiliste » (*probabilistic scheduling*) consiste à omettre de traiter certaines variables à chaque itération. Le choix de ces variables est aléatoire et dépend de la taille du cycle le plus petit associé à cette variable : plus celui-ci est petit, plus la probabilité de traiter la variable est faible. Cette méthode limite ainsi les phénomènes d'auto-confirmation introduits par les cycles courts. Elle permet d'obtenir une convergence plus rapide que celle du séquencement par inondation. Les architectures liées à ce séquencement ne seront pas abordées.

9.1.5 Construction d'un code *LDPC*

La construction d'un code *LDPC* (ou d'une famille de code *LDPC*) doit naturellement être effectuée de façon à optimiser les performances du code tout en minimisant la complexité matérielle du décodeur associé.

Cet ouvrage ayant déjà présenté les principes du décodage itératif (chapitre 7), nous nous contentons ici de rappeler les principes de construction d'un code *LDPC* permettant d'être efficacement décodé par l'algorithme à propagation de croyance. Nous renvoyons le lecteur souhaitant approfondir le sujet aux références données dans ce chapitre. Le problème de construction de code *LDPC* adapté à un décodage matériel sera abordé dans la section 9.2.

L'optimisation d'un code *LDPC* s'effectue en trois étapes :

- optimisation *a priori* des profils d'irrégularité des nœuds de parité et de variable ;
- construction de matrices **H** de taille adéquate respectant les profils d'irrégularité et maximisant la longueur des cycles ;
- éventuellement, choix ou rejet des codes sur le critère de la distance minimale ou sur les performances calculées par simulation.

Optimisation des profils d'irrégularités

On fait l'hypothèse de codes de taille infinie et d'un nombre infini d'itérations. Cela permet en effet l'optimisation de leurs caractéristiques asymptotiques (profil d'irrégularité, rendement) en fonction du canal visé. Deux techniques existent : l'algorithme d'évolution de densité et son approximation gaussienne, et les diagrammes de transfert d'information extrinsèque.

L'algorithme d'évolution de densité (*density evolution*) a été proposé par Richardson [9.24]. Cet algorithme calcule la densité de probabilité des messages $L_{j,p}$ et $Z_{j,p}$ après chaque nouvelle itération. L'algorithme est initialisé avec la densité de probabilité des échantillons entrants, qui dépend du niveau de bruit σ^2 du canal. L'utilisation de cet algorithme permet de connaître la valeur maximale de σ^2 en dessous de laquelle l'algorithme converge, c'est-à-dire telle que la probabilité d'erreur est plus faible qu'un seuil fixé au départ. Il est aussi possible de déterminer par programmation linéaire un profil d'irrégularité qui permette d'avoir le seuil le plus faible possible.

Une simplification de l'algorithme d'évolution de densité proposée par Chung *et al* [9.25, 9.26], est obtenue en remplaçant les densités de probabilité réelles par des densités gaussiennes voisines. L'intérêt de l'approximation par des densités gaussiennes est qu'il suffit de calculer l'évolution d'un seul paramètre en faisant l'hypothèse que ces densités gaussiennes sont consistantes, c'est-à-dire que la variance est égale à deux fois la moyenne. En effet, en supposant que le mot « tout 0 » a été envoyé, on a à l'initialisation ($n_{it} = 0$) :

$$L_{j,p}^{(0)} = -\frac{2X_j}{\sigma^2} \quad \text{avec} \quad X_j \sim N\left(-1, \sigma^2\right) \tag{9.26}$$

$$\text{donc } L_{j,p}^{(0)} \sim N\left(\frac{2}{\sigma^2}, \frac{4}{\sigma^2}\right)$$

On note :

$m_j^{(0)} = \frac{2}{\sigma^2}$ la moyenne de la densité de probabilité gaussienne consistante envoyée de la variable c_j de degré d_v vers les parités e_p de degré d_c qui y sont connectées,

$\mu_p^{(n_{it})}$ la moyenne des message $Z_{j,p}^{(n_{it})}$.

Pour suivre l'évolution de la moyenne $m_j^{(n_{it})}$ au cours des itérations n_{it} , il suffit alors de prendre l'espérance mathématique des équations (9.22) et (9.23) relatives au traitement des variables et des parités, ce qui donne :

$$\Psi\left(\mu_p^{(n_{it})}\right) = \Psi\left(m_j^{(n_{it})}\right)^{d_c-1} \text{avec } \Psi(x) = E\left[\tanh\left(x/2\right)\right],\ x \sim N\left(m, 2m\right) \tag{9.27}$$

$$m_j^{(n_{it}+1)} = \frac{2}{\sigma^2} + \left(d_v - 1\right)\mu_p^{(n_{it})} \tag{9.28}$$

Ainsi pour un code *LDPC* régulier (d_v, d_c) et pour un bruit donné de variance σ^2, les équations (9.27) et (9.28) permettent, par un calcul itératif, de

savoir si la moyenne des messages tend vers l'infini ou pas. Si tel est le cas, un décodage sans erreur avec un mot de code de taille infinie et un nombre infini d'itération est possible. Dans le cas d'un code irrégulier, il suffit de faire la moyenne pondérée sur les différents degrés des équations (9.27) et (9.28).

La valeur maximale de σ pour laquelle la moyenne tend vers l'infini, et donc pour laquelle la probabilité d'erreur tend vers 0, est le seuil du code (*threshold* en anglais). Par exemple, le seuil d'un code régulier (3,6), obtenu avec l'algorithme d'évolution de densité, est $\sigma_{\max} = 0.8809$ [9.26], ce qui correspond à un rapport signal sur bruit minimum de $\frac{E_b}{N_0}_{\min} = 1.1\text{dB}$.

Une autre technique dérivée des diagrammes de transfert d'information extrinsèque(*EXIT charts*[2]) proposés par Ten Brink [9.27, 9.28] permet d'effectuer une optimisation des profils d'irrégularité. Si l'algorithme d'évolution de densité s'intéresse à l'évolution au cours des itérations des densités de probabilité des messages, ces diagrammes s'intéressent quant à eux au transfert d'information mutuelle entre l'entrée et la sortie des décodeurs des codes constituants [9.28]. Le principe de ces diagrammes a été aussi utilisé avec d'autres paramètres que l'information mutuelle, comme le rapport signal sur bruit ou la probabilité d'erreur [9.29, 9.30]. Il a aussi été appliqué à d'autres types de canaux [9.31].

Optimisation de la taille des cycles

L'optimisation des profils d'irrégularité étant asymptotique, il s'agit maintenant de construire une matrice de contrôle de taille finie. Cette phase peut être réalisée de façon aléatoire : on tire au hasard les entrées non nulles de la matrice de contrôle en respectant au mieux le profil d'irrégularité des nœuds. Il est aussi possible de construire des codes en tirant au hasard des permutations d'une matrice élémentaire qui sont ensuite concaténées. Une autre façon de construire des codes *LDPC* est la construction déterministe de matrice (géométrie finie et projective).

Dans tous les cas, il faut prendre garde aux cycles présents dans le graphe du code, et cela d'autant plus qu'ils sont petits. L'algorithme de décodage par propagation de croyance suppose en effet que les cycles qui détérioreraient l'indépendance des messages entrant dans un nœud n'existent pas. Dans la pratique, la présence de cycles dans le graphe est inévitable. Mais s'ils sont suffisamment grands, l'indépendance des messages reste une bonne approximation. La construction de bons codes *LDPC* doit donc s'assurer de l'absence des plus petits cycles, ceux de taille 4. De très nombreuses solutions sont proposées dans la littérature pour construire des codes *LDPC*. Par exemple, Campello *et al* [9.32] proposent d'optimiser la taille du cycle minimum pour un rendement donné. Hu *et al* [9.33] suggèrent de construire le graphe branche par branche afin d'éviter au maximum les tailles de cycles les plus faibles (*Progressive Edge Geometry* ou *PEG*). Zhang *et al* [9.34] construisent des codes *LDPC* dont les cycles les plus petits sont de tailles 12, 16 ou 20, mais les variables ne sont que de degré 2. Tian *et al* [9.35] se basent sur le fait que tous les cycles de

[2] le principe de construction des diagrammes *EXIT* est décrit à la section 7.6.3

petites tailles n'ont pas la même influence et suppriment uniquement les plus pénalisants.

Sélection du code par la méthode impulsionnelle

Si les performances de décodage par l'algorithme de propagation de croyance sont améliorées par la suppression des cycles de petites tailles, il est également important aussi d'avoir de « bons » codes correcteurs d'erreurs, c'est-à-dire qui possèdent une distance minimale importante. La méthode impulsionnelle a d'abord été proposée par Berrou *et al*[9.36, 9.37] pour évaluer la distance minimale de Hamming d'un turbocode. Elle a ensuite été adaptée au cas des codes LPDC par Hu *et al* [9.38]. Elle permet ainsi de vérifier simplement que la distance minimale du code conçu soit suffisante pour atteindre le taux d'erreur cible pour l'application visée.

Sélection du code par simulation

Enfin, le choix final du code est obtenu par la simulation. En effet, deux codes de même taille et de même rendement, construits avec les mêmes profils d'irrégularités, n'ayant pas de cycle court et ayant la même distance minimale peuvent néanmoins avoir des performances sensiblement différentes. Ces différences peuvent s'expliquer par deux phénomènes : l'existence de points fixes « parasites » introduits par la sous-optimalité de l'algorithme de décodage itératif qui remonte le taux d'erreur binaire par rapport à la valeur théorique [9.39]. Le nombre de mots de code à distance minimale influe aussi sur les performances du code. La figure 9.8 montre les performances d'un code *LDPC* pour différentes tailles et différents rendements dans le cas d'un décodeur DVB-S2 implanté sur un FPGA Altera Stratix80.

Les codes *LDPC* ont donc des performances théoriques excellentes. Ceci doit toutefois se traduire par une simplicité de mise en œuvre matérielle pour que ces codes puissent être utilisés en pratique. C'est pourquoi une attention particulière doit être portée aux architectures et aux implémentations des décodeurs *LDPC*.

9.2 Architecture de décodage de codes *LDPC* pour le canal Gaussien

Lorsque l'algorithme à propagation de croyance est mis en œuvre, l'architecture générale des décodeurs de codes *LDPC* peut être réalisée à l'aide de processeurs de nœuds génériques (PNG) modélisant soit le traitement des parités, soit le traitement des variables. Cette section décrit les différentes réalisations possibles de ces processeurs après une analyse de la complexité du décodage des codes *LDPC*. Les différentes possibilités de contrôle de cette architecture à base de PNG permettent de définir trois classes de séquencement de l'algorithme à

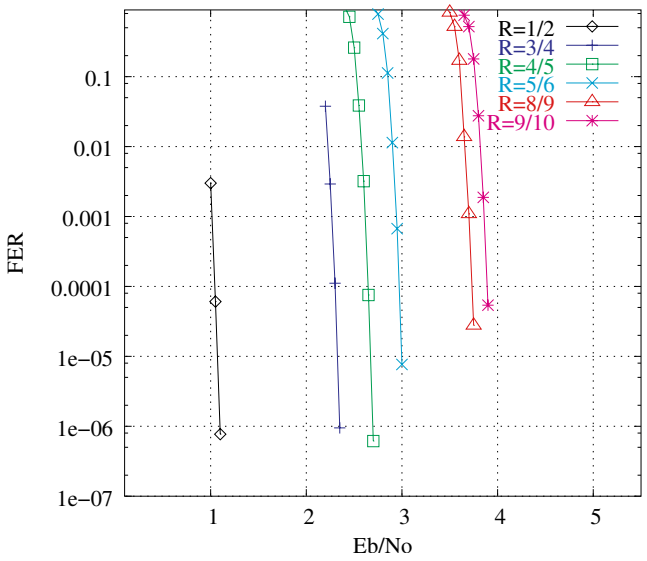


Figure 9.8 – Taux d'erreur paquet (*FER : Frame Error Rate*) obtenu pour des tailles de mot de code de 64 kbits et différents rendements de la norme DVB-S2 (50 itérations, virgule fixe). Avec la permission de TurboConcept S.A.S, France.

propagation de croyance : séquencement en deux passes, séquencements « vertical » et « horizontal ». Cette présentation originale et unifiée des architectures de décodeurs de codes *LDPC* permet de couvrir de nombreuses architectures existantes et publiées à ce jour et de synthétiser des architectures novatrices.

9.2.1 Analyse de la complexité

La complexité du décodage des codes *LDPC* est directement liée au nombre de branches dans le graphe bipartite du code, ou au nombre de 1 dans la matrice de contrôle. L'algorithme de décodage itératif à propagation de croyance comporte deux étapes. Dans chaque étape, il s'agit de calculer l'information $L_{j,p}$ ou $Z_{j,p}$ qui est associée à la branche reliant la variable j à la parité p. Notons B le nombre de branche du graphe bipartite du code *LDPC*. Dans le cas par exemple d'un code régulier (d_v, d_c) de taille n, le nombre de branches B est donné par :

$$B = d_v n = d_c m \tag{9.29}$$

La puissance de calcul P_c nécessaire au décodage des codes *LDPC* est alors définie comme le nombre de branches à traiter par cycle d'horloge. Ce paramètre dépend :

- du nombre k de bits d'information à transmettre par mot de code,
- du nombre de branches B,
- du débit D d'information souhaité,

- du nombre maximum d'itération N_{it},
- de la fréquence d'horloge f_{clk}.

En une seconde, le nombre de mots de code à traiter pour obtenir un débit d'information D est égale à D/k (mots/seconde). Dans le pire des cas, le décodage d'un mot de code requiert le calcul de $B \times N_{it}$ branches. Pour garantir un débit D, une architecture doit fournir une puissance de calcul de $D \times B \times N_{it}/k$ branches par seconde. La puissance de calcul minimale P_c à fournir par cycle d'horloge est donc :

$$P_c = \frac{BN_{it}D}{kf_{clk}} \text{ (branches/cycle)} \tag{9.30}$$

Remarquons que, pour une architecture entièrement parallèle dans laquelle chaque nœud du graphe est associé à un processeur, toutes les branches du graphe sont traitées en un cycle d'horloge. La puissance de calcul est alors $(P_c)_{\max} = B$. Vouloir dépasser cette puissance en pratique ne présente pas d'intérêt puisque le chemin critique devient alors plus long.

9.2.2 Architecture d'un Processeur de Nœud Générique (PNG)

Les calculs effectués dans un processeur de nœud de variable (PNV) et dans un processeur nœud de parité (PNP) ont une dépendance identique entre les entrées et les sorties. En effet, tant pour le PNP que pour le PNV, les d sorties sont calculées à partir des d entrées, avec la i-ième sortie dépendant de toutes les entrées moins la i-ième entrée. Il est ainsi possible de représenter les différentes architectures de processeurs de façon abstraite par un processeurs de nœud générique. Celui-ci sera ensuite spécialisé en fonction de l'algorithme de décodage utilisé. Le PNG reçoit donc en entrée d messages $(e_i)_{i=1..d}$ et produit en sortie d messages $(s_j)_{j=1..d}$ définis par :

$$s_j = \bigotimes_{i \neq j} e_i \tag{9.31}$$

L'opérateur $\otimes$ est un opérateur générique de calcul, associatif et commutatif, dont la réalisation sera spécifiée par la suite. L'expression condensée 9.31 signifie que l'opérateur s'applique à toutes les variables e_i pour $i \neq j$.
La figure 9.9 illustre les trois principales versions d'architectures parallèles de PNG :

- *Architecture directe* : les calculs des d messages de sortie sont effectués indépendamment (figure 9.9(a)). Les calculs des différentes sorties peuvent aussi être factorisés. Le nombre de composants $\otimes$ traversés est de l'ordre de $\log_2(d)$.
- *Architecture treillis (type Forward-Backward)* : cette architecture correspond à une forme factorisée particulière de l'architecture parallèle qui présente une grande régularité, mais dont le nombre d'opérateurs $\otimes$ est linéaire avec d.

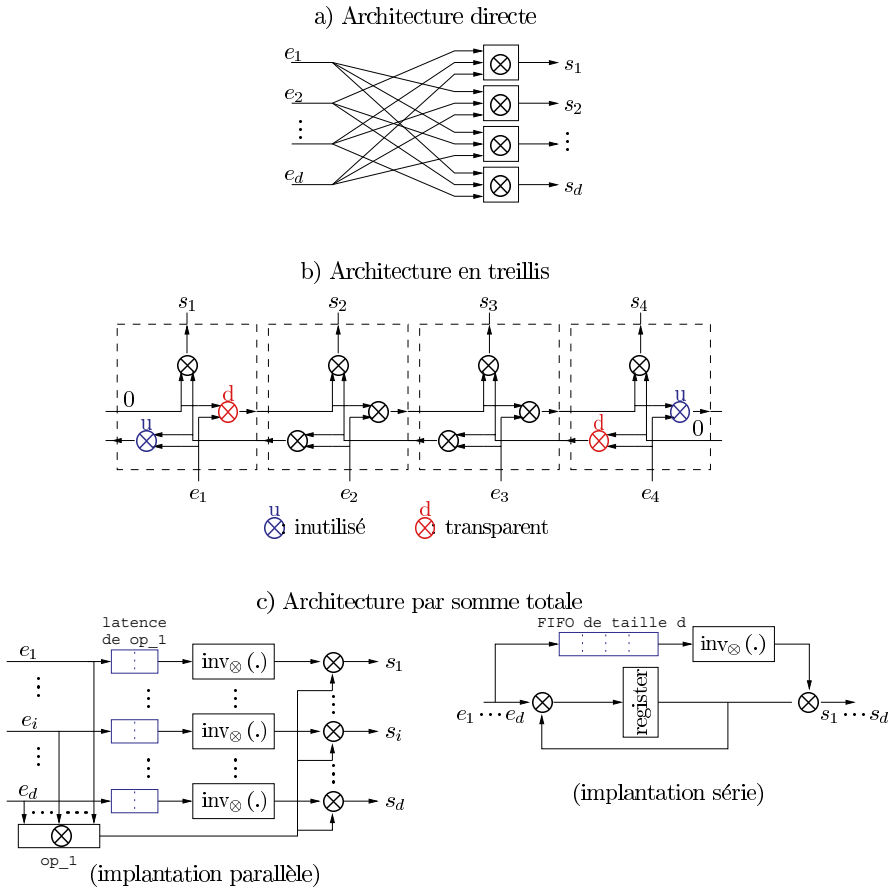


Figure 9.9 – Les différentes architectures « mode compact » pour l'implantation de l'opérateur générique $\otimes$.

- *Architecture par somme totale* : cette architecture n'est possible que si l'opérateur générique $\otimes$ admet un inverse noté $inv_\otimes$. Dans ce cas, l'opérateur générique est appliqué à toutes les entrées (somme totale) puis chaque sortie est calculée en éliminant la contribution de l'entrée correspondante à l'aide de l'opérateur inverse.

Il est possible de modifier ces architectures pour introduire des registres intermédiaires de *pipeline* permettant de réduire le chemin critique. Il existe également des architectures de type série (figure 9.9(c)).

Dans la suite, le degré de parallélisme d'un PNG sera noté α_g. Il s'agit du nombre de cycles nécessaires pour traiter un nœud (sans considérer la latence due au traitement *pipeline*). Ainsi, pour une architecture parallèle capable de traiter un nœud à chaque cycle d'horloge, $\alpha_g = 1$, tandis que pour une archi-

tecture série, $\alpha_g = d$.

Notons que dans toutes les architectures de PNG présentées, nous avons fait implicitement l'hypothèse que toutes les entrées étaient disponibles et que toutes les sorties devaient être générées soit simultanément (architecture parallèle), soit de façon groupée dans le temps (architecture série). Un tel mode de contrôle du PNG est appelé « mode compact ».

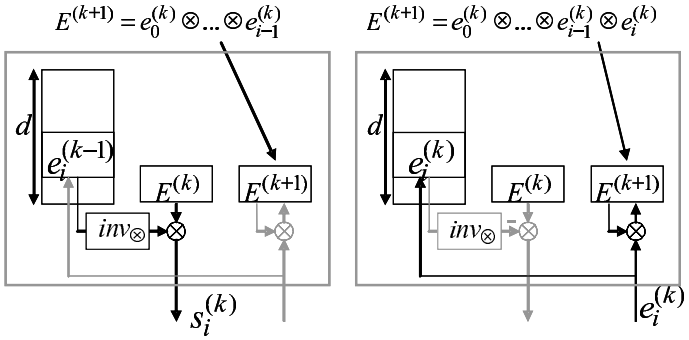


Figure 9.10 – Principe du mode distribué (mise à jour différée).

Il est possible d'imaginer des modes d'exécution différents, comme le « mode distribué », dans lesquels entrées et sorties du PNG sont réparties le long d'une itération de décodage.

La figure 9.10 permet d'illustrer le fonctionnement en mode distribué d'un processeur :

- Pendant l'itération courante n_{it}, on considère que les variables d'entrée e_i proviennent de l'itération précédente $n_{it} - 1$ tandis que les variables de sortie appartiennent à l'itération courante.
- À la fin d'une itération, on suppose que les d variables d'entrées $(e_i^{(n_{it}-1)})_{i=1..d}$ sont mémorisées dans une mémoire (interne ou externe au PNG) de même que la valeur de $E^{(n_{it})} = \underset{i=1..d}{\otimes} e_i^{(n_{it}-1)}$.
- Le PNG peut donc, à la requête du système, calculer la i-ième sortie $s_i^{(n_{it})} = E^{(n_{it})} \otimes inv_\otimes(e_i^{(n_{it}-1)})$.
- Cette sortie est envoyée, à travers l'entrelaceur, au nœud opposé qui, une fois le calcul terminé, renvoie $e_i^{(n_{it})}$.
- Cette nouvelle valeur remplace alors $e_i^{(n_{it}-1)}$ dans la mémoire et est aussi accumulée pour obtenir la valeur de $E^{(n_{it}+1)}$ à la fin de l'itération. Deux modes d'accumulation sont possibles :

  1. Le premier mode – mise à jour différée (figure 9.10) – consiste à utiliser un registre d'accumulation initialisé à zéro lors de chaque nouvelle itération. Ce registre permet de calculer directement $E^{(n_{it})} = \underset{i=1..d}{\otimes} e_i^{(n_{it})}$. Cette architecture possède donc $d + 2$ mots mémoire, d pour les entrées $(e_i^{(n_{it}-1)})_{i=1..d}$, un mot pour $E^{(n_{it})}$ et un mot pour

l'accumulation de $E^{(n_{it}+1)}$.

2. Le deuxième mode – mise à jour immédiate – consiste, dès qu'une nouvelle entrée $e_i^{(n_{it})}$ arrive, à remplacer la contribution de $e_i^{(n_{it}-1)}$ dans $E^{(n_{it}-1)}$ par celle de $e_i^{(n_{it})}$, soit :

$$E^{(n_{it}-1)} = E^{(n_{it}-1)} \otimes e_i^{(n_{it})} \otimes inv_{\otimes}\left(e_i^{(n_{it}-1)}\right) \tag{9.32}$$

À la fin de l'itération, on a ainsi $E^{(n_{it})} = E^{(n_{it}-1)}$. Cette solution offre deux avantages par rapport à la mise à jour différée :
- un mot de mémoire en moins ;
- une accélération de la convergence de l'algorithme car les nouvelles valeurs des entrées sont prises en compte plus tôt.

Choix de l'opérateur générique

La figure 9.11 donne une vue « en coupe » de l'algorithme à propagation de croyance sur le graphe bipartite du code *LDPC*. On suppose que chaque branche est dédoublée pour différencier les messages des variables vers les parités et les messages des parités vers les variables. Cette vue montre la grande ressemblance entre le traitement des variables et le traitement des parités et permet d'imaginer d'autres positions du réseau d'interconnexion dans le cycle de calcul. Chaque position du graphe d'interconnexion se traduit alors par un traitement différent des nœuds de parité et des nœuds de variable. Le tableau 9.2. donne les différents calculs à effectuer en fonction de la position du réseau d'interconnexion.

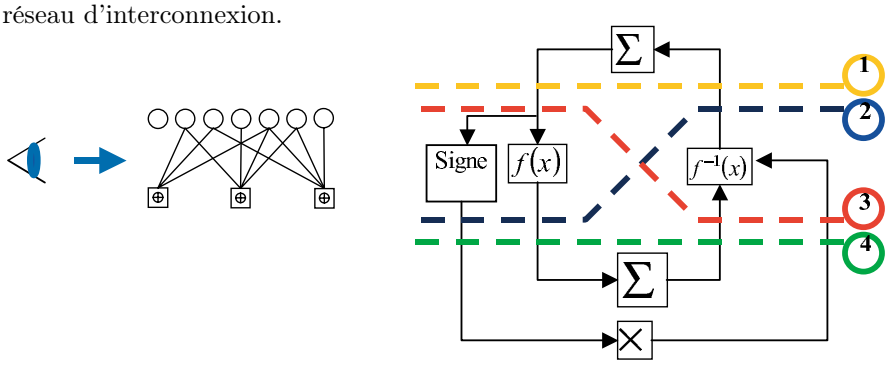


Figure 9.11 – Différentes positions du réseau d'interconnexion obtenues en effectuant le traitement des nœuds de parité dans le domaine de Fourier. Ces positions séparent les parties de l'itération à réaliser dans les PNV de celles réalisées dans les PNP.

Lorsque le réseau d'interconnexion est en position 1 (table 9.2), on retrouve la séparation classique entre processeur de variables et processeur de parités. Ce dernier peut alors soit être réalisé dans le domaine fréquentiel (comme indiqué dans la figure 9.10), soit directement dans le domaine des LRV par l'intermédiaire de l'opérateur $\oplus$ défini dans l'équation (9.3).

Position du réseau		1	2	3	4
PNV		Σ	$f \circ \Sigma$	$\Sigma \circ f$	$f \circ \Sigma \circ f$
PNP module	Fourier	$f \circ \Sigma \circ f$	$f \circ \Sigma$	$\Sigma \circ f$	Σ
	Direct	$\bigoplus$			
PNP signe		Produit des signes			

Table 9.2 – Valeur de l'opérateur générique associé aux processeurs de variables (PNV) et de parité (PNP) en fonction de la position du réseau d'interconnexion.

9.2.3 Architecture générique de propagation des messages

Présentation du modèle

Les PNP et PNG sont caractérisés par leur architecture et leur opérateur générique, dépendant de la position du réseau d'interconnexion. L'architecture présentée sur la figure 9.12 permet l'échange des messages entre ces différents processeurs.

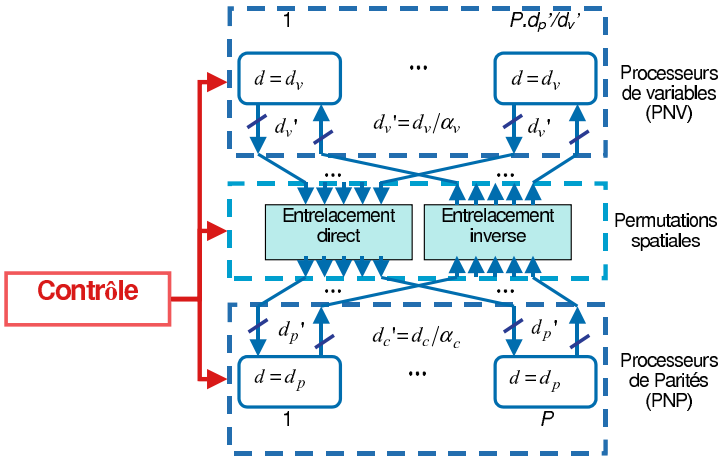


Figure 9.12 – Architecture série-parallèle générique.

Cette architecture est composée de P PNP qui génèrent chacun $d = d_p$ messages en α_p cycles d'horloge. Ces processeurs ont donc $d'_c = d_c/\alpha_p$ entrées et sorties. Ils sont connectés à un réseau d'entrelacement direct et inverse. De l'autre coté de ce réseau d'entrelacement, sont placés $\left(Pd'_p/d'_v\right)$ PNV. Ces processeurs génèrent de même $d = d_v$ messages en α_v cycles d'horloge, et ont donc $d'_v = d_v/\alpha_v$ entrées et sorties.

Le degré de parallélisme de l'architecture est défini par les trois paramètres P, α_p et α_v. Il est possible d'obtenir tous les degrés de parallélisme possibles, de l'architecture entièrement parallèle avec $P = m$, $\alpha_p = 1$ et $\alpha_v = 1$ à l'architecture entièrement série où $P = 1$, $\alpha_c = d_c$ et $\alpha_v = d_v$.

Notons qu'une telle architecture possède une puissance de calcul (équation (9.30)) $P_c = P \times d'_c$.

Les réseaux d'entrelacement directs et inverses permettent d'aiguiller les messages associés aux différents PNV vers différents PNP et inversement. Ce genre de réseaux permet généralement de réaliser plusieurs permutations dites spatiales. Un autre type de permutations, dites temporelles, permet d'accéder de façon aléatoire aux nœuds associés à un même processeur de nœuds, par adressage mémoire par exemple. La combinaison de ces deux types de permutations permet une interconnexion aléatoire telle que celle existant entre les nœuds de variable et les nœuds de parité du code *LDPC*.

Exemple de mise en œuvre

Pour bien fixer les idées sur la façon d'organiser les calculs et la propagation des messages dans le décodeur, et pour bien comprendre le lien entre l'organisation de la propagation des messages et la structure du code *LDPC*, la figure 9.13 illustre un exemple simple de décodage d'un code *LDPC* de longueur $n = 12$ et de rendement $R = 0,5$ (donc $m = 6$), avec $P = 2$, $d_c = 3$, $\alpha = 1$ et $\beta = d_v$. Il y a donc $P = 2$ processeurs de nœuds de parité et $n/P = 6$ processeurs de nœuds de variable. Une itération se fait en $m/P = 3$ étapes :

- Lors du premier cycle, une lecture des informations relatives aux bits est faite dans chacun des PNV, chacun d'eux contenant $n/P = 2$ bits du mot de code (en pratique, n/P peut être beaucoup plus élevé). Ces bits sont grisés dans chaque PNV : c'est la permutation spatiale.
- Ces informations sont ensuite envoyées aux PNP via le réseau de permutation, dont l'adresse a été générée à partir du numéro de cycle (lecture dans une mémoire par exemple) : c'est la permutation temporelle.
- La combinaison des deux décrit l'entrelacement aléatoire entre les variables et les deux premières parités du graphe bipartite représenté sur la partie droite de la figure.

En un seul cycle, les deux premières parités vont donc pouvoir être traitées. Les deux suivantes seront traitées au deuxième cycle et ainsi de suite jusqu'à ce que toutes les parités du code aient été traitées. Remarquons que cette technique où les informations du PNP arrivent simultanément interdit que deux bits contenus dans le même PNV puissent être impliqués dans la même parité. Ainsi, par exemple, les bits 1 et 2 ne peuvent pas être impliqués dans la même parité car cela entraînerait un conflit mémoire. Cette solution impose donc des contraintes sur la matrice $\mathbf{H}$, si l'on souhaite qu'elle soit décodable par cette structure. Une solution pour relâcher les contraintes consiste par exemple à entrer en série les données dans les parités.

9.2.4 Combinaisons des paramètres de l'architecture

Un certain nombre de paramètres caractérisant les architectures de décodeurs de codes *LDPC* ont été définis précédemment :

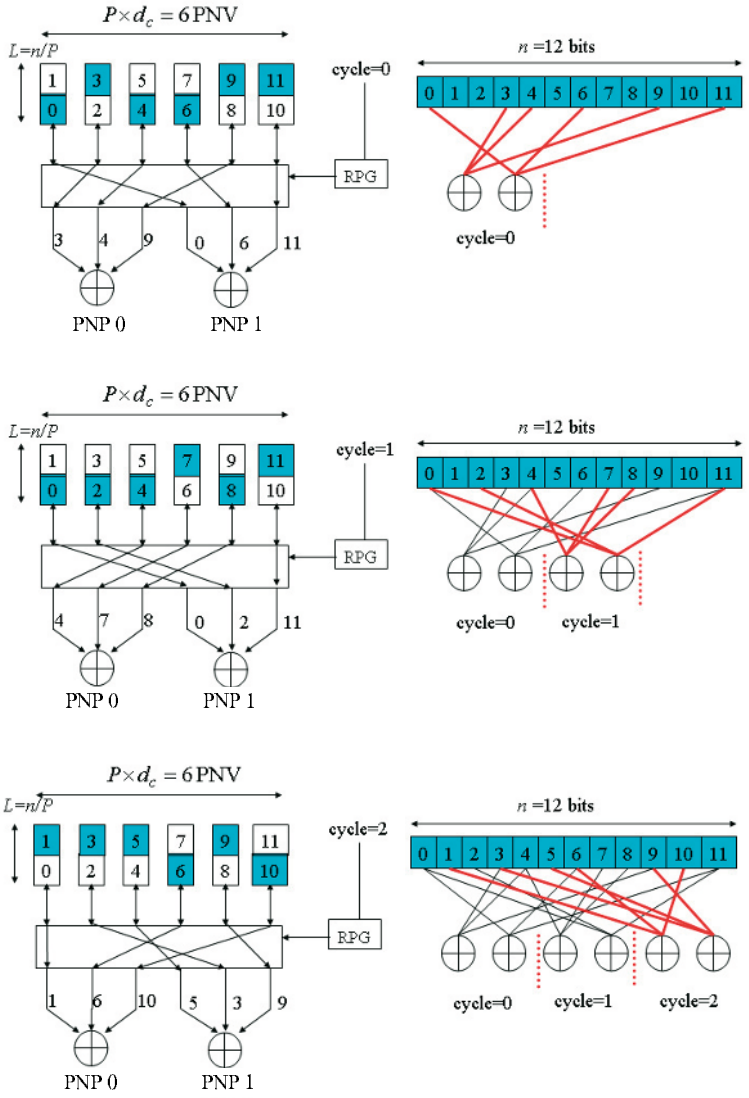


Figure 9.13 – Exemple d'architecture de propagation de message : lien entre l'adressage du décodeur et la structure du code.

- Processeurs de nœuds :
  - 3 architectures possibles (directe, treillis, somme totale)
  - 4 positions possibles du réseau d'interconnexion (voir figure 9.11)
  - 3 modes de contrôle d'entrées-sorties (compact, distribué avec mise à jour différée ou immédiate)

- Architecture de propagation de messages :
  - 3 paramètres caractérisant le niveau de parallélisme *(P, α, β)*

Toutes les combinaisons de ces différents paramètres sont possibles pour décrire ou créer une architecture de décodeur *LDPC*. Bien sûr, certaines de ces combinaisons sont plus ou moins intéressantes, selon les spécifications demandées. Par exemple, les combinaisons des modes de contrôle entre les PNV et les PNP, illustrant les différents séquencements possibles de décodage, sont données dans le tableau 9.3.

<table>
<tr><td colspan="3"></td><td colspan="3">PNV</td></tr>
<tr><td colspan="3" rowspan="2"></td><td rowspan="2">Compact</td><td colspan="2">Distribué</td></tr>
<tr><td>Mise à jour différée</td><td>Mise à jour immédiate</td></tr>
<tr><td rowspan="4">PNP</td><td colspan="2">Compact</td><td>Inondation</td><td>Inondation (parité)</td><td>Entrelacement (horizontal)</td></tr>
<tr><td rowspan="3">Distribué</td><td>Mise à jour différée</td><td>Inondation (variable)</td><td colspan="2" rowspan="2">Branches</td></tr>
<tr><td>Mise à jour immédiate</td><td>Entrelacement (vertical)</td></tr>
</table>

Table 9.3 – Séquencements associés aux différentes combinaisons des contrôles des processeurs de nœuds.

Dans le cas où les contrôles sur les deux processeurs sont de types flot compact d'entrées-sorties, le séquencement réalisé est de type inondation : tous les PNP sont traités puis tous les PNV. Ce séquencement se prête volontiers aux architectures entièrement parallèles ($P = m$). Pour les architectures mixtes ($P < m$), le traitement de toutes les parités ne peut pas se faire entièrement avant celui des variables. Le contrôle des PNV en mode distribué avec mise à jour différée permet de le faire car elle garantit que le calcul des nouvelles sorties ne se fera que lorsque toutes les parités auront été traitées. Ce mode de contrôle est noté par inondation selon les parités. De façon symétrique, on fait apparaître un séquencement par inondation selon les variables, lorsque les PNP sont en mode distribué et que les PNV sont en mode compact. Ces trois types de séquencements convergent vers les même valeurs : ils ne changent pas le déroulement de la propagation de l'information.

Lorsque l'un des deux type de processeur est contrôlé en mode compact et l'autre en mode distribué avec mise à jour immédiate, on met en œuvre un séquencement de type entrelacement (*shuffle*) horizontal ou vertical. L'ordre d'activation des processeurs est analogue au séquencement par inondation selon les variables ou les parités. Seule la mise à jour des informations change puisqu'elle se fait dès qu'une nouvelle entrée est arrivée, accélérant ainsi la convergence du code.

Le cas où les deux processeurs PNV et PNP sont contrôlés en mode distribué n'est pas très intéressant. Il correspondrait en fait au contrôle du décodage branche par branche.

La mémoire nécessaire pour implanter ces différentes combinaisons est donnée dans le tableau 9.4.

			PNV		
			Compact	Distribué	
				Mise à jour différée	Mise à jour immédiate
PNP	Compact		$B+n$	$3n+g(B,d_c)$	$2n+g(B,d_c)$
	Distribué	Mise à jour différée	$B+n+2m$	$3n+2m+B$	$2n+2m+B$
		Mise à jour immédiate	$B+n+m$	$3n+m+B$	$2n+m+B$

Table 9.4 – Quantité de mémoire nécessaire en fonction des combinaisons des différents contrôles de processeurs de nœuds.

Le paramètre B désigne comme précédemment le nombre de branches dans le graphe. Chaque information extrinsèque de branche doit être mémorisée, quel que soit le séquencement utilisé. Dans tous les cas il faut aussi mémoriser l'information intrinsèque des variables, soient n valeurs. Lorsque le mode de contrôle sur les parités est compact, il faut mémoriser dans chaque PNV l'accumulation des messages des n variables, soient n mémoires si on les met à jours immédiatement, et $2n$ dans le cas contraire. Le raisonnement est le même si les PNV sont en mode compact et que les PNP sont en mode distribué, mais dans ce cas ce sont les accumulations de messages dans les m parités qui doivent être mémorisées.

Il est parfois possible, comme nous le verrons ultérieurement, de mémoriser les messages $Z_{j,p}$ de façon comprimée. Le nombre de message à mémoriser passe alors de B à $g(B,d_c)$, avec g représentant une fonction de compression ($g(B,d_c) < B$).

9.2.5 Exemple de synthèse d'architecture de décodeurs *LDPC*

Les deux exemples décrits dans cette partie permettent d'illustrer deux architectures de décodeur *LDPC* utilisant deux séquencements différents. Pour chacun de ces exemples, on donnera les valeurs des paramètres caractérisant ces architectures.

Séquencement par inondation (selon les parités)

L'architecture décrite ici pour illustrer la séquencement par inondation est basée sur celle proposée initialement par Boutillon *et al* [9.40]. Elle est schématisée sur la figure 9.14. Dans cet exemple, $P = 3$ PNP fonctionnent simultanément en mode compact. Comme $\alpha_p = 1$ et $\alpha_v = 3$, il y a 12 PNV qui fonctionnent simultanément mais en mode distribué (seuls un PNV et un PNP

Paramètres		Valeurs
Architecture de propagation de message		($\alpha_p = 1$, $\alpha_v = d_v = 3$, $P = 3$)
Position du réseau d'interconnexion		1
PNV	Contrôle	Distribué, mise à jour différée
	Chemin de données	Somme totale, série
PNP	Contrôle	Compact
	Chemin de données	Treillis, parallèle

Table 9.5 – Paramètres caractérisant l'architecture à séquencement par inondation.

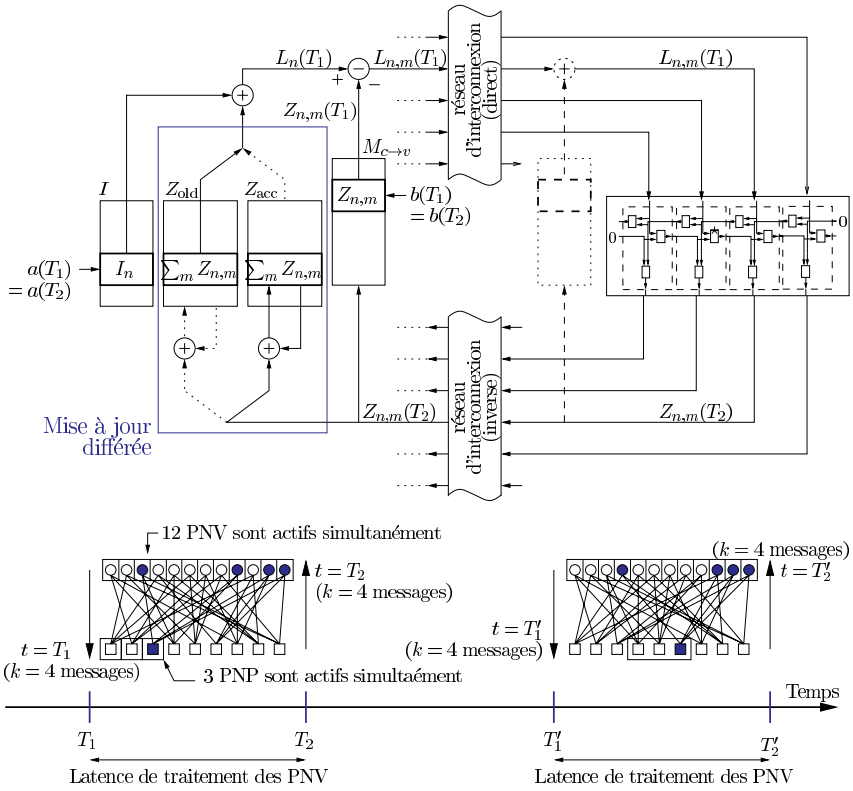


Figure 9.14 – Exemple d'architecture pour un séquencement par inondation (selon les parités).

sont représentés sur la figure). Notons que la puissance de calcul d'une telle architecture est de 12 branches par cycle.

L'architecture des PNP est de type treillis, avec une implantation parallèle. Le chronogramme en bas de la figure 9.14 indique qu'au temps T_1, $d_v = 4$ messages $L_{j,p}(T_1)$ sont délivrés à chaque PNP. Après une latence de $T_2 - T_1$ cycles d'horloge les messages sortant $Z_{j,p}(T_2)$ sont envoyés aux PNV. Cette

opération est reproduite m/P fois pour effectuer une itération complète. Le chemin de données des PNV est de type somme totale, avec une implantation série. La mise à jour différée s'illustre par l'utilisation de deux blocs de mémoire, un pour l'information extrinsèque en cours d'accumulation (Lacc), et un autre pour l'information extrinsèque totale de l'itération précédente (Lold). À la fin de chaque itération, le rôle de ces deux mémoires est échangé. Dans cette architecture, les informations extrinsèques de branches $Z_{j,p}$ peuvent être sauvegardées aussi bien du côté des PNV (en trait continu sur la figure) que du côté des PNP (en pointillé sur la figure), comme dans les architectures de Chen *et al* [9.41] et de Guilloud *et al* [9.42].

Séquencement par entrelacement horizontal

Ce deuxième exemple d'architecture illustre le séquencement par entrelacement horizontal, proposé par Mansour *et al* [9.43] dans un cas particulier de décodage turbo des codes *LDPC*. Dans cet exemple, illustré dans la figure 9.15, il y a $P = 3$ PNP qui fonctionnent simultanément en mode compact. Comme $\alpha_p = 4$ et $\alpha_v = 3$, il y a 3 PNV qui fonctionnent simultanément en mode distribué. Ce qui donne une puissance de calcul de 3 branches par cycle.

Les chemins de données des PNV et PNP sont tous les deux de type somme totale, avec une implantation série. À partir du temps T_1, $d_c = 4$ messages $L_{j,p}$ entrent en série dans le PNP. Après une latence de calcul de $T_2 - T_1$, les messages $Z_{j,p}$ calculés sont renvoyés toujours de façon série aux PNV qui sont contrôlés en mode distribué. Mais dans ce cas, la mise à jour des informations est immédiate. Cela se traduit par l'utilisation d'un seul bloc de mémoire *Lacc*. Ainsi, la somme des informations extrinsèques du bits j est mise à jour aussitôt qu'une nouvelle entrée $Z_{j,p}$ se présente.

Paramètres		Valeurs
Architecture de propagation de message		($\alpha_p = d_c = 4$, $\alpha_v = d_v = 3$, $P = 3$)
Position du réseau d'interconnexion		4
PNV	Contrôle	Compact
	Chemin de données	Somme totale, série
PNP	Contrôle	Distribué, mise à jour immédiate
	Chemin de données	Somme totale, série

Table 9.6 – Valeurs des paramètres caractérisant l'architecture à entrelacement vertical.

9.2.6 Algorithme de décodage sous optimaux

Afin de réduire la complexité du décodeur *LDPC*, de nombreux algorithmes de décodage « sous-optimaux » ont été proposés. Ces algorithmes sont basés sur un même principe : la réduction de la complexité et de la mémoire des processeurs de nœuds (de parité ou de variable), en remplaçant le calcul individuel

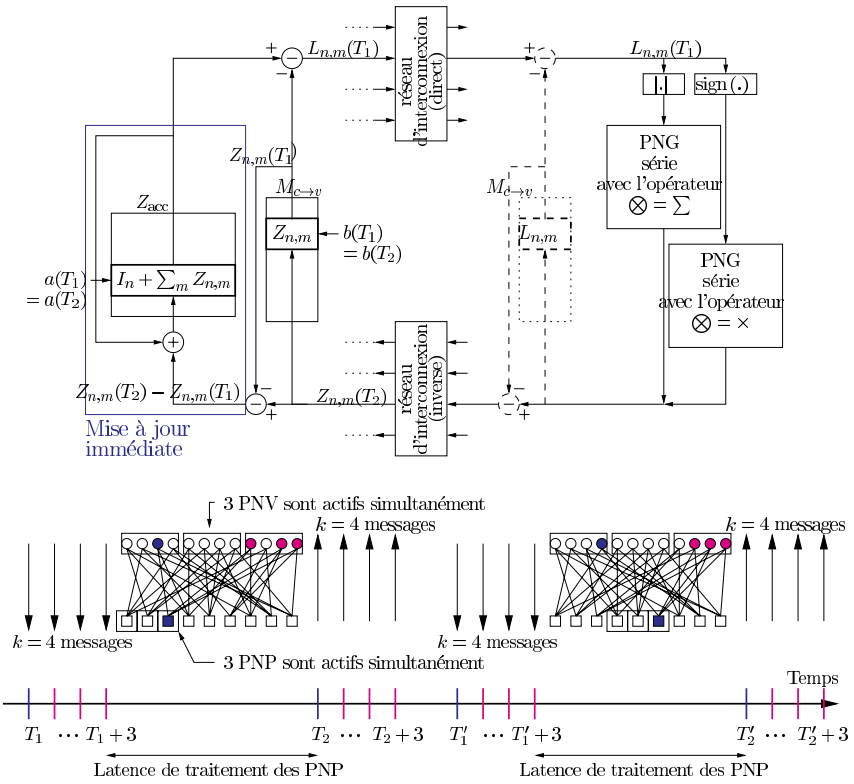


Figure 9.15 – Exemple d'architecture pour un séquencement par entrelacement vertical. L'implémentation série du PNP n'est pas détaillée dans cette figure.

des d messages de sortie (avec d le degré du nœud) par le calcul de Δ ($\Delta < d$) valeurs distinctes. Bien sûr, l'utilisation d'un algorithme sous optimal dégrade généralement les performances du code. Il s'agit alors de faire un compromis entre performance et complexité.

Algorithme de décodage à message unique ($\Delta = 1$)

Il s'agit ici de l'algorithme le plus simple puisque toutes les sorties du processeur de nœud (variable ou parité) sont affectées par une unique et même valeur à chaque étape du processus itératif.

PNV avec $\Delta = 1$

Dans cette technique, le PNV renvoie simplement L_j aux contraintes de parités auxquelles il est connecté. Ainsi, il n'est plus nécessaire de mémoriser les messages $Z_{j,p}$ puisque ceux-ci ne sont plus utilisés par le PNV. Il en découle une économie de mémoire significative. Cet algorithme a été proposé pour la

première fois par Fossorier *et al* dans [9.44] (*APP algorithm*), et repris par E. Yeo *et al* dans [9.45].

Notons que l'hypothèse d'indépendance entre les messages sortant et entrant d'un nœud de parité n'est absolument pas vérifiée. C'est pourquoi l'algorithme de décodage itératif diverge très rapidement car sujet au phénomène d'auto-confirmation : la propagation de l'information se déroule comme si des cycles de longueur 2 existaient dans le graphe.

PNP avec $\Delta = 1$

Il s'agit de l'algorithme symétrique du précédent : le PNP renvoie une unique valeur. Cette technique, très efficace en terme de complexité, permet à l'algorithme d'atteindre sa capacité de correction en très peu d'itérations, typiquement 5. Même si sa capacité de correction est très faible par rapport à l'algorithme *BP*, il est intéressant de noter que pour 5 itérations, un tel algorithme est plus performant que l'algorithme *BP* après ce même nombre d'itérations. Ainsi, de tels procédés peuvent s'appliquer avec succès pour les applications à hauts débits où seul un nombre réduit d'itération peut être effectué.

Algorithmes sous optimaux de PNP ($\Delta > 1$)

Il existe dans l'état de l'art trois algorithmes pour $\Delta > 1$ concernant le PNP.

Algorithme Min-sum ou BP-Based ($\Delta = 2$*)*

Cet algorithme proposé par Fossorier *et al* [9.44] s'affranchit de tout calcul dans le PNP. En effet, l'auteur suggère d'approximer l'équation de traitement de parité (9.22) par :

$$\begin{cases} |Z_{j,p}| = \underset{j' \in J(p)/j}{\text{Min}} (|L_{j',p}|) \\ \text{sign}\,(Z_{j,p}) = \prod\limits_{j' \in J(p)/j} \text{sign}\,(L_{j',p}) \end{cases} \tag{9.33}$$

Seul le calcul du module change : il est approximé par excès par le minimum des modules des messages entrant dans le PNP. Le traitement dans le PNP consiste donc uniquement à calculer le signe et à trier les deux plus faibles module des messages entrant. Notons que cette approximation rend le traitement itératif de décodage indépendant de la connaissance du niveau de bruit σ^2 du le canal. La perte de performance est de l'ordre de 1 dB environ par rapport à l'algorithme *BP*.

Cette approximation par excès de l'algorithme *Min-Sum* peut cependant être compensée par des méthodes simples. Il est ainsi possible de réduire la valeur de $|Z_{j,p}|$ en l'affectant d'un facteur multiplicatif A strictement inférieur à 1. Il est aussi possible de lui soustraire un offset B (B>0), en prenant toutefois la précaution de saturer le résultat à zéro si le résultat de $|Z_{j,p}| - B$ est négatif.

La valeur de $|Z_{j,p}|$ corrigée $|Z_{j,p}|^c$ est donc :

$$\begin{cases} |Z_{j,p}|^c = A \times \max(|Z_{j,p}| - B, 0) \\ \text{sign}\,(Z_{j,p}) = \prod\limits_{j' \in J(p)/j} \text{sign}\,(L_{j',p}) \end{cases} \tag{9.34}$$

Ces deux variantes de l'algorithme *Min-sum* se nomment respectivement *Offset BP-based* et *Normalized BP-Based* [9.46]. L'optimisation des coefficients A et B est permet de différencier des décodeurs. Ils peuvent être constants ou variables en fonction du rapport signal à bruit, du degré de la contrainte de parité, du numéro de l'itération traitée...

Algorithme $\lambda - \min(\Delta = \lambda + 1)$

Cet algorithme a été présenté initialement par Hu *et al* [9.47, 9.48] puis reformulé indépendamment par Guilloud *et al* [9.42]. L'allure de la fonction f définie équation (9.35) est telle que $f(x)$ est grand pour x faible, et faible lorsque x est grand. Ainsi, la somme dans (9.35) peut être approximée par ses λ plus fortes valeurs, c'est-à-dire par les λ plus faibles valeurs de $|L_{j,p}|$. Une fois que l'ensemble noté $J_\lambda(p)$ des λ minima est obtenu, le PNP va calculer $\Delta = \lambda + 1$ modules distincts :

$$|Z_{j,p}| = f\left(\sum_{j' \in J_\lambda(p)/j} f\,|L_{j',p}|\right) \quad \text{avec} \quad f\,(x) = \ln\,\tanh\left(\frac{x}{2}\right) \tag{9.35}$$

En effet, si j' est l'indice d'un bit ayant envoyé une des valeurs de l'ensemble des minimum, le module est calculé sur tous les $\lambda - 1$ autres minima (λ calculs sur $\lambda - 1$ valeurs). Par contre pour tous les bits le même module est renvoyé (un calcul sur λ valeurs). Il faut noter que les performances de l'algorithme $\lambda - \min$ peuvent être améliorées par l'ajout de facteur de correction A et B comme défini dans l'équation (9.34).

Algorithme $A - \min*(\Delta = 2)$

Le dernier algorithme sous optimal publié à ce jour est dénommé algorithme « $A - \min*$ » et a été proposé par Jones *et al* [9.51]. Ici aussi il s'agit tout d'abord de trouver l'indice j_0 du bit ayant le message de plus faible module : $j_0 = Arg\,\text{Min}_{j \in J(p)}\,(|L_{j,p}|)$. Ensuite, deux messages distincts sont calculés :

$$\text{Si} \quad j = j_0 : |Z_{j_0,p}| = f\left(\sum_{j \in J(p)/j_0} f\,|L_{j,p}|\right) \tag{9.36}$$

$$\text{Sinon} \quad j \neq j_0 : |Z_{j,p}| = f\left(\sum_{j \in J(p)} f\,|L_{j,p}|\right) \tag{9.37}$$

Une comparaison de performance en terme de taux d'erreur binaire (faisceaux de courbe inférieur) et taux d'erreur de paquets (faisceaux de courbe

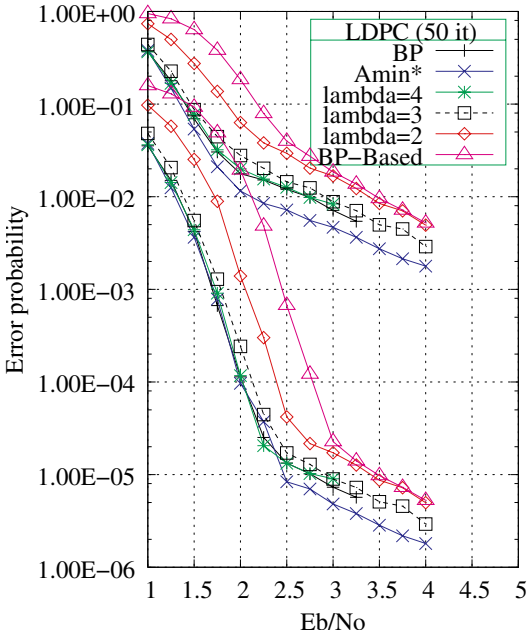


Figure 9.16 – Comparaison des performances entre les algorithmes $3-\min$ et $A-\min*$ dans le cas du décodage d'un code irrégulier C_3.

supérieur) entre les algorithmes sous-optimaux et l'algorithme *BP* est présentée sur la figure 9.16. Le code simulé est un code irrégulier de taille $n = 1008$ et de rendement $R = 0,5$, construit par Hu (*IBM Zurich Research Labs*) à l'aide de la technique *PEG* (*Progressive Edge Growth*) [9.14]. Les degrés des parités sont ainsi presque tous égaux à 8. Les degrés des variables varient de 2 à 15. On constate que l'algorithme sous-optimal n'est pas forcément moins performant que l'algorithme *BP* (typiquement pour l'algorithme $A - \min*$). Aucune compensation d'information extrinsèque n'a été utilisée pour ces simulations. Il est possible d'améliorer remarquablement les performances de ces algorithmes en leur ajoutant cette compensation, comme pour les algorithmes *offset* et *normalized BP-based* cités précédemment.

9.2.7 Influence de la quantification

La quantification est susceptible d'apporter une dégradation sur les performances en affectant essentiellement deux points : le calcul de l'information intrinsèque I_i définie par (9.21) et le calcul de l'information extrinsèque de branche, (9.22) et totale (9.24).

On distingue deux paramètres de quantification : le nombre de bits de quantification n_q et la dynamique de quantification δ. Les valeurs représentables sont ainsi comprises entre $-\delta$ et $+\delta$. Cela signifie que la position de la virgule n'est

pas forcément définie dans les n_q bits, ou bien autrement dit que la dynamique de codage n'est pas forcément une puissance de deux. Ainsi, le bit de poids le plus faible vaudra :

$$q_{\text{LSB}} = \frac{\delta}{2^{n_q-1}} \tag{9.38}$$

et l'on passe d'une donnée quantifiée a_q à une donnée non quantifiée a par les relations :

$$\begin{cases} a_q = \text{trunc}\left(a\frac{2^{n_q-1}}{\delta} + 0.5\right), \\ a = a_q\frac{\delta}{2^{n_q-1}} \end{cases} \tag{9.39}$$

où *trunc* désigne l'opération de troncature.
Ces deux paramètres peuvent influencer les performances de décodage. Une dynamique trop faible laisse apparaître un plancher d'erreur sur les courbes de taux d'erreurs. Ce plancher apparaît beaucoup plus tôt que celui lié à la distance minimale du code.

Cependant il est important de noter qu'augmenter la dynamique sans augmenter le nombre de bits de quantification augmente la valeur du bit de poids le plus faible, et par conséquent diminue la précision des calculs faits dans le PNP. Augmenter la dynamique sans augmenter le nombre de bits dégrade les performances de décodage dans la zone de convergence.

Les performances de décodage obtenues en pratique sont très proches de celles obtenues en virgule flottante pour des quantifications sur 4 à 6 bits (voir le tableau figure 9.17). L'influence des paramètres peut être étudiée par l'algorithme d'évolution de densité [9.31, 9.52].

9.2.8 État de l'art des architectures de décodeurs *LDPC* publiées

Le tableau de la figure 9.17 rassemble les principales caractéristiques des circuits de décodeurs *LDPC* publiées dans la littérature à ce jour. Les entrées du tableau sont les suivantes :

- Circuit : description du type de circuit utilisé (*ASIC* ou *FPGA*) ou, éventuellement, indication de l'arrêt du travail à la synthèse.
- Auteurs : référence aux auteurs et articles concernant la plateforme.
- Architecture :
  - Décodeur : indication du type de décodeur (série, parallèle ou mixte) avec les (des) paramètres (P, α_p, α_v) associés à l'architecture de propagation de messages.
  - Chemin de données : indication pour chaque processeur de nœud (variable et contrainte) du type d'architecture utilisée (directe, treillis ou somme totale).
  - Contrôle : indication pour chaque processeur de nœud (variable et contrainte) du type de contrôle utilisé, compact ou distribué, et le cas échéant du type de mise à jour.
  - Position du réseau d'interconnexion (entre 1 et 4).

circuit	Auteurs	Architecture								caractéristiques du code				Format de codage (bits)	Fréquence d'horloge (MHz)	débit (Mb/s)	Nombre max. d'itération
		Décodeur			chemin de données		Contrôle		position du réseau d'inter-connexion	n	m	R	degrés				
		P	α	B	PNP	PNV	PNP	PNV									
synthèse	Levine *et. al.* 2000 [9.53]	Série			parallèle (pas de LRV)				1	1200	900	0,25	dv=3 dc=4	8	50	4	10
		1	1	1													
DSP - TMS320C6201	Blaut *et. al.* 2000 [9.54]	série (?)			Série (?)					200	100		dv=3 dc=6	16	200	0,133	
ASIC 0.16um (7.5x7 mm²)	Blanksby *et. al.* 2002 [9.55]	Parallèle			Somme totale		Compacte			1024	512		dv=3,6,7,8 dc=6,7	4	64	1000	64
		M	1	1													
FPGA Xilinx Virtex-E 2600	Zhang *et. al.* 2002 [9.56]	Mixte			Treillis	direct	Compacte	Compacte	3	9216	4608	0,5	dv=3 dc=6	5	56	54	18
		3.dc	1	1													
ASIC 0.18um (3.1x4.2 mm²)	Marsour *et. al.* 09-2003 [9.57]	Mixte			Treillis	Somme totale	Compacte	Distribué, mise à jour immédiate		2048	1024			4	125	1600	1
		64	1	1													
ASIC 0.18um (9.41 mm²)	Marsour *et. al.* 12-2003 [9.58]	Mixte			Treillis	Somme totale	Compacte	Distribué, mise à jour immédiate	1	2304	1536	2/3	irrégulier	5	200	192	10
		64	1	1													
FPGA Xilinx Virtex-II 8000	Chan. Y. *et. al.* 2003 [9.41]	Mixte			Somme totale		compacte	Distribué, mise à jour différée		8088	4044	0,5	irrégulier	6	44	80	25
ASIC 0.13um (2.61 mm²)		24	1	1											212	376	
ASIC 0.13um (49.6mm²) 0.09um (15.58mm²)	Urard P. *et. al.* 2005 [9.59]	360								DVB-S2+BCH				6	200	90	
															300	135	
ASIC 0.13um (22.74mm²)	Kienle F. *et. al.* 2005 [9.60]	360								DVB-S2				6	270	255	30

Figure 9.17 – État de l'art des différentes plateformes publiées.

- Caractéristiques du code *LDPC* : taille, rendement et régularité du code *LDPC*.
- Format de codage : nombre de bits utilisés pour représenter les données dans le décodeur.
- Fréquence d'horloge de la puce en MHz.
- Débit brut (en bits par seconde). Les bits d'information et les bits de redondance sont comptés (le débit binaire utile s'obtient en multipliant par le rendement).
- Nombre maximum d'itérations.

Aucune architecture à ce jour n'a été publiée décrivant en détail un décodeur utilisant un algorithme sous-optimal. On mentionnera toutefois celle de Jones *et al* [9.51] qui contient cependant trop peu d'informations pour être classée ici.

9.3 Bibliographie

[9.1] R. G. Gallager, *Low-Density Parity-Check Codes*, Cambridge, MA, MIT Press, 1963.

[9.2] R. Tanner, « A recursive approach to low complexity codes », *IEEE Trans. Inform. Theory*, vol. 27, pp. 533-547, Septembre 1981.

[9.3] D.J.C MacKay and R.M. Neal, « Good Codes Based on Very Sparse Matrices », 5^{th} *IMA Conference on Cryprography and Coding*, Berlin, Germany, Springer, 1995.

[9.4] N. Wiberg, *Codes and Decoding on General Graphs*, Ph.D. diss., Linköping University, Sweden, 1996.

[9.5] M. Sipser and D.A. Spielman, « Expander Codes », *IEEE Trans. Inform. Theory*, vol. 42, pp. 1710-1722, November 1996.

[9.6] X.-Y.Hu , E. Eleftheriou, D.-M. Arnold and A. Dholakia, « Efficient Implementations of the Sum-Product Algorithm for Decoding LDPC Codes », *Proc. IEEE Global Telecommunications Conference*, GLOBECOM'01, pp. 1036-1036, Nov. 2001.

[9.7] F. Guilloud, *Generic Architecture for LDPC codes decoding*, Thèse de Doctorat ENST Paris, 2 juillet 2004.

[9.8] T.J. Richardson and R.L Urbanke, « Efficient Encoding of Low-Density Parity-Check Codes », *IEEE Trans. Inform. Theory*, vol. 47, pp. 638-656, February 2001.

[9.9] T.R. Oenning and Jaekyun Moon, « A low-density generator matrix interpretation of parallel concatenated single bit parity codes », *IEEE Trans. Magn.*, vol. 37, pp. 737- 741, March 2001.

[9.10] D.J.C. Mackay, « Good Error-Correcting Codes Based on Very Sparse Matrices », *IEEE Trans. Inform. Theory*, vol. 45, pp. 399-431, March 1999.

[9.11] J. Garcia-Frias and Wei Zhong, « Approaching Shannon performance by iterative decoding of linear codes with low-density generator matrix », *IEEE Commun. Lett.*, vol. 7, pp. 266-268, June 2003.

[9.12] D.J.C MacKay, S.T. Wilson and M.C. Davey, « Comparison of Constructions of Irregular Gallager Codes », *IEEE Trans. Commun.*, vol. 47, pp. 1449-1454, October 1999.

[9.13] J.W. Bond, S. Hui and H. Schmidt, « Constructing low-density parity-check codes », *EUROCOMM 2000. Information Systems for Enhanced Public Safety and Security*, IEEE AFCEA, May 2000.

[9.14] X.Y. Hu, E. Eleftheriou and D.-M. Arnold, « Progressive edge-growth Tanner graphs », *Proc. IEEE Global Telecommunications Conference*, GLOBECOM'01, Nov. 2001.

[9.15] D. Haley, A. Grant and J. Buetefuer, « Iterative encoding of low-density parity-check codes », *Proc. IEEE Global Telecommunications Conference*, GLOBECOM'02, Nov. 2002.

[9.16] T. Okamura, « Designing LDPC codes using cyclic shifts », *Proc. IEEE Int. Symp. Inform. Theory*, July 2003.

[9.17] Y. Kou, S. Lin and M.P.C. Fossorier, « Low-Density Parity-Check Codes Based on Finite Geometries : A Rediscovery and New Results », *Trans. Inform. Theory*, vol. 47, pp. 2711-2736, November 2001.

[9.18] B. Ammar, B. Honary, Y. Kou, and S. Lin, « Construction of low density parity check codes : a combinatoric design approach », *Proc. IEEE Int. Symp. Inform. Theory*, July 2002.

[9.19] B. Vasic, « Combinatorial constructions of low-density parity check codes for iterative decoding », *Proc. IEEE Int. Symp. Inform. Theory*, July 2002.

[9.20] R. J. McEliece, D. J. C. MacKay and J.-F. Cheng, « Turbo Decoding as an Instance of Pearl's Belief Propagation Algorithm », *IEEE J. Select. Areas Commun.*, Vol. 16, pp. 140-152, February 1998.

[9.21] I.B. Djordjevic, S. Sankaranarayanan and B.V. Vasic, « Projective-Plane Iteratively Decodable Block Codes for WDM High-Speed Long-Haul Transmission Systems », *J. Lightwave Technol.*, Vol. 22, March 2004.

[9.22] F.R. Kschischang and B.J. Frey, « Iterative Decoding of Compound Codes by Probability Propagation in Graphical Models », *IEEE J. Select. Areas Commun.*, Vol. 16, pp. 219-230, 1998.

[9.23] Y. Mao and A.H. Banihashemi, « Decoding Low-Density Parity-Check Codes With Probabilistic Scheduling », *IEEE Commun. Lett.*, Vol. 5, pp. 414-416, October 2001.

[9.24] T.J. Richardson, M.A. Shokrollahi and R.L. Urbanke, « Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes », *IEEE Trans. Inform. Theory*, Vol. 47, pp. 619-637, February 2001.

[9.25] S.-Y. Chung, T.J. Richardson, and R.L. Urbanke, « Analysis of Sum-Product Decoding of Low-Density Parity-Check Codes Using a Gaussian Approximation », *IEEE Trans. Inform. Theory*, vol. 47, February 2001.

[9.26] S.-Y. Chung, *On the construction of some capacity-approaching coding schemes*, Ph. D. Dissertation, MIT, Cambridge, MA, 2000.

[9.27] S. Ten Brink, « Convergence of iterative decoding », *IEE Electronics Lett.*, Vol. 35, pp. 806-808, May 1999.

[9.28] S. Ten Brink, « Iterative decoding trajectories of parallel concatenated codes », *Proc.* 3^{rd} *IEEE ITG Conference on Source and Channel Coding*, pp. 75-80, Munich, Jan. 2000.

[9.29] M. Ardakani and F.R. Kschischang, « Designing irregular LPDC codes using EXIT charts based on message error rate », *Proc. Int. Symp. Inform. Theory.*, July 2002.

[9.30] M. Ardakani, T.H. Chan, and F.R. Kschischang, « Properties of the EXIT Chart for One-Dimensional LDPC Decoding Schemes », *Proc. Canadian Workshop Inform. Theory*, May 2003.

[9.31] D. Declercq, *Optimisation et performances des codes LDPC pour des canaux non standards*, Thèse d'habilitation à diriger des recherches, université de Cergy Pontoise, décembre 2003.

[9.32] J. Campello and D.S. Modha, « Extended Bit-Filling and LDPC Code Design », *Proc. IEEE Global Telecommunications Conference*, GLOBECOM'01, pp. 985-989, Nov. 2001.

[9.33] X.-Y. Hu, , E. Eleftheriou, and D.-M. Arnold, « Progressive edge-growth Tanner graphs », *Proc. IEEE Global Telecommunications Conference*, GLOBECOM'01, Nov. 2001.

[9.34] H. Zhang and J.M.F. Moura, « The design of structured regular LDPC codes with large girth », *Proc. IEEE Global Telecommunications Conference*, GLOBECOM'03, Dec. 2003.

[9.35] T. Tian, C. Jones, J.D. Villasenor, and R.D. Wesel, « Construction of irregular LDPC codes with low error floors », *Proc. IEEE Int. Conf. Communications*, ICC'03, 2003.

[9.36] C. Berrou and S. Vaton, « Computing the minimum distance of linear codes by the error impulse method », *Proc. IEEE Int. Symp. Inform. Theory*, July 2002.

[9.37] C. Berrou, S. Vaton, M. Jézéquel, and C. Douillard, « Computing the minimum distance of linear codes by the error impulse method », *Proc. IEEE Global Telecommunications Conference*, GLOBECOM'02, Taiwan, Nov. 2002.

[9.38] X.-Y. Hu, M.P.C. Fossorier and E. Eleftheriou, « On the Computation of the Minimum Distance of Low-Density Parity-Check Codes », *Proc. IEEE Int. Conf. Communications*, ICC'04, 2004.

[9.39] R. Koetter and P.O. Vontobel, « Graph-covers and the iterative decoding of finite length codes », *Proc.* 3^{rd} *Int. Symp. Turbocodes and Related Topics*, sept. 2003

[9.40] E. Boutillon, J. Castura, and F.R. Kschischang, « Decoder-First Code Design », *Proc.* 2^{nd} *Int. Symp. Turbocodes and Related Topics*, Brest, France, pp. 459-462, 2000.

[9.41] Y. Chen and D. Hocevar, « A FPGA and ASIC Implementation of Rate 1/2, 8088-b Irregular Low Density Parity Check Decoder », *Proc. IEEE Global Telecommunications Conference*, GLOBECOM'03, 1-5 Dec. 2003.

[9.42] F. Guilloud, E. Boutillon and J.-L. Danger, « lambda-min Decoding Algorithm of Regular and Irregular LDPC Codes », *Proc.* 3^{rd} *Int. Symp. Turbocodes and Related Topics*, 1-5 Sept. 2003.

[9.43] M.M. Mansour and N.R. Shanbhag, « Turbo decoder architectures for low-density parity-check codes », *Proc IEEE Global Telecommunications Conference*, GLOBECOM'02, 17-21 Nov. 2002.

[9.44] M.P.C.Fossorier, M. Mihaljevic, and I. Imai, « Reduced Complexity Iterative Decoding of Low-Density Parity-Check Codes Based on Belief Propagation », *IEEE Trans. on Commun.*, Vol. 47, pp. 673-680, May 1999.

[9.45] E. Yeo, B. Nikolic, and V. Anantharam, « High Throughput Low-Density Parity-Check Decoder Architectures », *Proc. IEEE Global Conference on Telecommunications*, GLOBECOM'01, San Antonio, 25-29 Nov. 2001.

[9.46] J. Chen and M. Fossorier, « Near optimum universal belief propagation based decoding of low-density parity check codes », *IEEE Trans. Commun.*, vol. 50, pp. 406-414, Mar. 2002.

[9.47] X.-Y. Hu and R. Mittelholzer, « An ordered-statistics-based Approximation of the sum-product algorithm », *Proc. IEEE Int. Telecommunications Symp.*, ITS2002, Natal, Brazil, September 8-12, 2002.

[9.48] X.-Y. Hu and R. Mittelholzer, « A Sorting-based approximation of the sum-product algorithm », *Journal of the Brazilian Telecommunications Society*, Vol. 18, No. 1, June 2003, pp. 54-60.

[9.49] D.J.C. Mackay, *LDPC database*,
Available at http ://www.inference.phy.cam.ac.uk/mackay/codes/data.html.

[9.50] R. Urbanke, *programme LdpcOpt*,
Available at http ://lthcwww.epfl.ch/research/ldpcopt/.

[9.51] C. Jones, E. Vallés, M. Smith and J. Villasenor, « Approximate min* constraint node updating for LDPC code decoding », *Proc. IEEE Military Communications Conference*, MILCOM'03, 13-16 Oct. 2003.

[9.52] J. Chen and M.P.C. Fossorier, « Density Evolution for Two Improved BP-Based Decoding Algorithms of LDPC Codes », *IEEE Commun. Lett.*, Vol. 6, pp. 208-210, May 2002.

[9.53] B. Levine, R.R. Taylor and H. Schmit, « Implementation of near Shannon limit error-correcting codes using reconfigurable hardware », *Proc. IEEE Symp. Field-Programmable Custom Computing Machines*, pp. 217-226, 2000.

[9.54] T. Bhatt, K. Narayanan, and N. Kehtarnavaz, « Fixed-point DSP Implementation of Low-Density Parity Check Codes », *Proc.* 9^{th} *DSP Workshop*, DSP 2000 , Oct. 2000.

[9.55] A.J. Blanksby and C.J. Howland, « A 690-mW 1-Gb/s 1024-b, Rate-1/2 Low-Density Parity-Check Code Decoder », *IEEE J. Solid-State Circuits*, Vol. 37, pp. 404-412, March 2002.

[9.56] T. Zhang and K. K. Parhi, « A 54 Mbps (3,6)-regular FPGA LDPC Decoder », *Proc. IEEE Workshop on Signal Processing Systems*, SIPS'02, 2002.

[9.57] M.M. Mansour and N. R. Shanbhag, « A Novel Design Methodology for High-Performance Programmable Decoder Cores for AA-LDPC Codes », *Proc. IEEE Workshop on Signal Processing Systems*, SIPS'03, Seoul, Korea, pp. 29-34, Aug. 2003.

[9.58] M.M. Mansour and N.R. Shanbhag, « High-throughput LDPC decoders », *IEEE Trans. VLSI Syst.*, Vol. 11, no. 6, pp. 976-996, Dec. 2003.

[9.59] P. Urard, L. Paumier , P. Georgelin, T. Michel, V. Lebars, E. Yeo and B. Gupta, « A 135Mbps DVB-S2 compliant codec based on 64800-bit LDPC and BCH codes », (ISSCC paper 24.3) DAC 2005, pp. 547-548, 2005.

[9.60] F. Kienle , T. Brack and N. Wehn, « A Synthesizable IP Core for DVB-S2 LDPC Code Decoding », *Proc. Design, Automation and Test in Europe*, DATE'05, Vol. 3, pp. 100-105, 2005.

Chapitre 10

Turbocodes et transmissions à grande efficacité spectrale

Le transport de l'information dans les systèmes de télécommunication s'effectue à des débits toujours plus élevés et dans des bandes de fréquences de plus en plus étroites. On cherche par conséquent à maximiser le rapport débit utile sur bande, c'est-à-dire l'efficacité spectrale des transmissions. Pour ce faire, il apparaît naturel de coupler des modulations numériques à grand nombre de points avec des codes correcteurs d'erreurs puissants de haut rendement tels que les turbocodes.

Les études menées dans ce domaine sont essentiellement basées sur deux approches : les *turbo-modulations codées en treillis* et les *modulations turbocodées pragmatiques*.

10.1 Les turbo-modulations codées en treillis (TMCT)

Les turbo-modulations codées en treillis ou TMCT ont été introduites par Robertson et Wörz en 1995 [10.1, 10.2]. Elles font appel à la notion de concaténation parallèle, qui est à l'origine du turbocodage, appliquée à deux modulations codées en treillis ou MCT.

Les MCT, introduites par Ungerboeck au début des années 80 [10.3] sont basées sur l'optimisation conjointe du codage correcteur d'erreurs et de la modulation. Le codage est directement réalisé dans l'espace des signaux si bien que le code correcteur d'erreurs et le code binaire à signal de la modulation peuvent être représentés conjointement à l'aide d'un treillis unique. Le critère d'optimisation d'une MCT consiste alors à maximiser la distance euclidienne minimale entre deux séquences codées. Pour ce faire, Ungerboeck a proposé une démarche en deux temps : effectuer un partitionnement de la constellation

de la modulation en sous-constellations présentant des distances euclidiennes minimales croissantes, puis affecter à chaque branche du treillis un signal appartenant à la constellation en respectant un ensemble de règles telles que décrites dans [10.3].

Le schéma de TMCT présenté par Robertson et Wörz est décrit en figure 10.1. Chaque codeur MCT est constitué d'un codeur convolutif systématique récursif, ou codeur CSR, de rendement $q/(q+1)$ et d'une modulation sans mémoire d'ordre $Q = 2^{q+1}$. Les symboles binaires issus de la source sont regroupés par symboles de q bits. Ces symboles sont codés par la première MCT dans l'ordre où ils sont délivrés par la source et par la seconde après entrelacement.

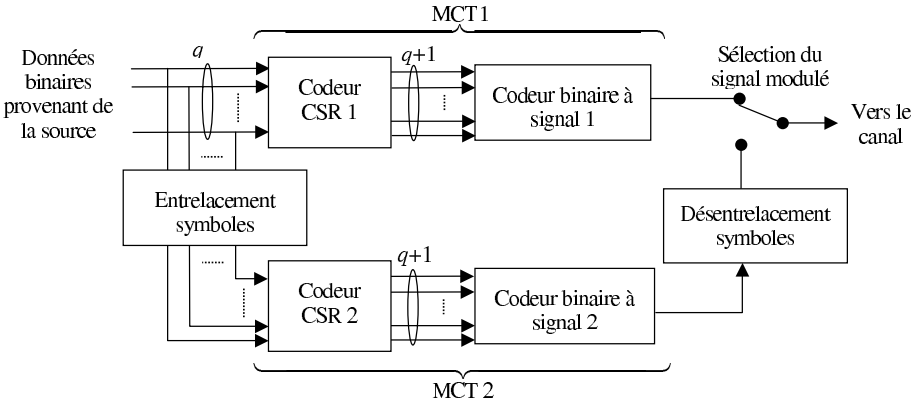


Figure 10.1 – Schéma de principe d'une turbo-modulation codée en treillis (TMCT), selon Robertson et Wörz [10.1, 10.2]. Efficacité spectrale $\eta = q$ bit/s/Hz.

Chaque q-uplet issu de la source étant codé deux fois, un opérateur de sélection transmet alternativement la sortie d'un des deux codeurs MCT, afin d'éviter la double transmission d'information, qui conduirait à une efficacité spectrale du système de $q/2$ bit/s/Hz. Ceci revient en fait à poinçonner la moitié de la séquence de redondance pour chaque code convolutif.

En réception, le décodeur de la TMCT est similaire à un turbo-décodeur, à ceci près que celui-là traite directement les symboles $(q+1)$-aires issus du démodulateur. Ainsi, le calcul des probabilités de transition à chaque étape de l'algorithme *MAP* (voir section 7.4) utilise la distance euclidienne entre le symbole reçu et le symbole porté par chaque branche du treillis. Si l'algorithme de décodage opère dans le domaine logarithmique (Log-*MAP*, Max-Log-*MAP*), ce sont les métriques de branches qui sont prises égales aux distances euclidiennes. Le calcul avant décodage d'une estimation des bits portés par chaque symbole démodulé constituerait, en effet, une mise en œuvre sous-optimale du récepteur.

De même, pour une mise en œuvre performante du turbo-décodage, les informations extrinsèques échangées par les décodeurs élémentaires doivent porter directement sur les q-uplets d'information transmis et non sur les éléments binaires dont ils sont constitués. À chaque instant de décodage, les décodeurs élémentaires s'échangent ainsi 2^q valeurs d'information extrinsèque.

La figure 10.2 fournit deux exemples de codes CSR élémentaires utilisés dans [10.1, 10.2] pour réaliser une turbo-MDP-8 en treillis à 8 états d'efficacité spectrale $\eta = 2$ bit/s/Hz et une turbo-MAQ-16 d'efficacité spectrale $\eta = 3$ bit/s/Hz.

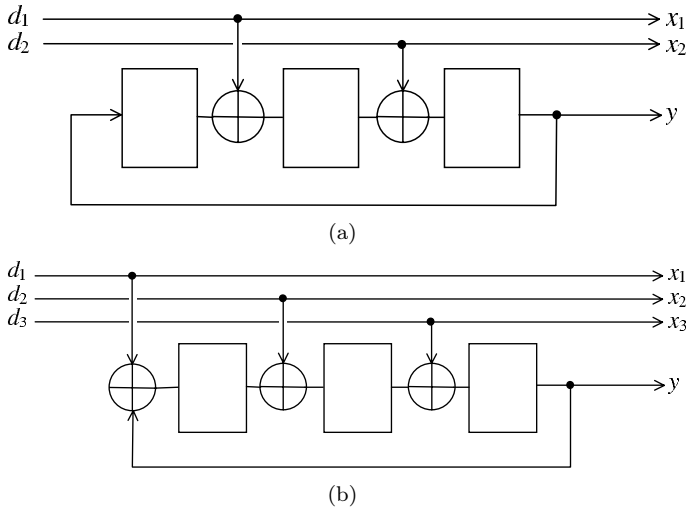


Figure 10.2 – Exemples de codes CSR élémentaires utilisés dans [10.1, 10.2] pour la construction d'une turbo-MDP-8 en treillis (a) et d'une turbo-MAQ-16 en treillis (b).

Les figures 10.3 et 10.4 montrent les performances de ces deux TMCT en terme de taux d'erreurs binaires (TEB) en fonction du rapport signal à bruit pour une transmission sur canal gaussien. Aux taux d'erreurs forts et moyens, ces schémas affichent des performances de correction proches de la capacité : un TEB de 10^{-4} est atteint à environ 0,65 dB de la limite théorique de Shannon pour la transmission de paquets de 5000 symboles codés et modulés. En revanche, la fonction d'entrelacement de la TMCT n'ayant pas fait l'objet d'une optimisation particulière dans [10.1, 10.2], les courbes de taux d'erreurs présentées font apparaître des changements de pente précoces (TEB$\sim 10^{-5}$) et prononcés.

Une variante de cette technique, proposée par Benedetto *et al* [10.4] a permis d'en améliorer les performances asymptotiques. Une méthode alternative pour construire une TMCT d'efficacité spectrale q bit/s/Hz consiste, en effet, à utiliser deux codes CSR de rendement $q/(q+1)$ et à poinçonner pour chacun d'eux $q/2$ bits d'information (q est supposé pair). On ne transmet ainsi, pour

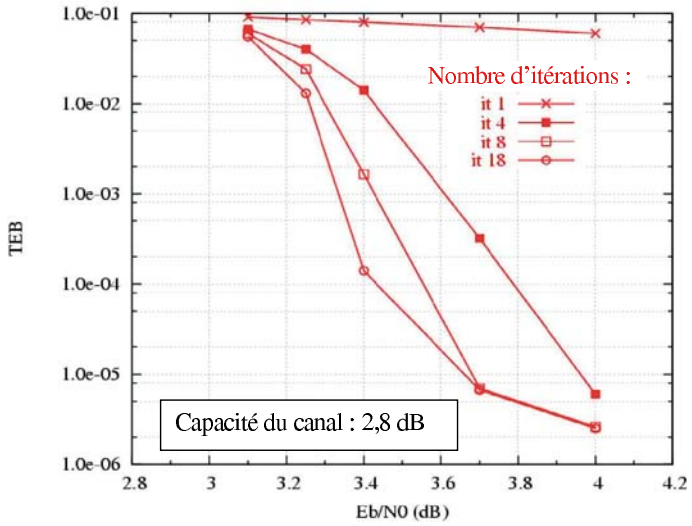


Figure 10.3 – Taux d'erreurs binaires (TEB) en fonction du rapport signal à bruit E_b/N_0 de la turbo-MDP-8 en treillis à 8 états utilisant le code CSR de la figure 10.2(a). Transmission sur canal gaussien. Efficacité spectrale $\eta = 2$ bit/s/Hz. Blocs de 10 000 bits d'information, 5000 symboles modulés. Algorithme de décodage *MAP*. Courbes extraites de [10.2].

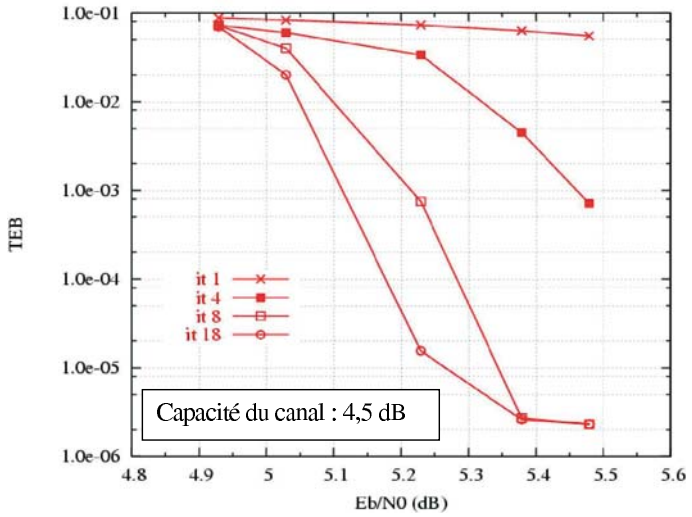
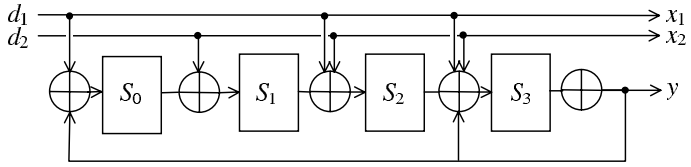


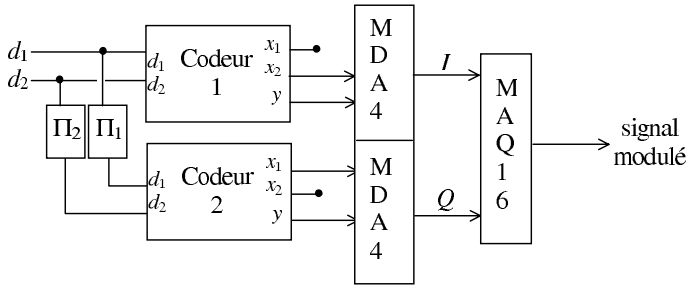
Figure 10.4 – TEB en fonction du rapport signal à bruit E_b/N_0 de la turbo-MAQ-16 en treillis à 8 états utilisant le code CSR de la figure 10.2(b). Transmission sur canal gaussien. Efficacité spectrale $\eta = 3$ bit/s/Hz. Blocs de 15 000 bits d'information, 5000 symboles modulés. Algorithme de décodage *MAP*. Courbes extraites de [10.2].

chaque code élémentaire que la moitié des bits d'information et la totalité des bits de redondance. Les bits en sortie de chaque codeur sont associés à une modulation à $2^{(q/2)+1}$ points. La même opération est réalisée pour les deux codes CSR en veillant à ce que chaque bit systématique soit transmis une fois et une seule, afin que le turbocode résultant soit systématique. Cette technique fait d'autre part appel à un entrelacement au niveau bit et non au niveau symbole comme dans l'approche précédente.

Le critère d'optimisation de la TMCT proposée dans [10.4] repose sur la maximisation de la *distance euclidienne effective*, définie comme la distance euclidienne minimale entre deux séquences codées dont les séquences d'information ont un poids de Hamming égal à 2. Les figures 10.5 et 10.6 montrent deux exemples de TMCT construites sur ce principe.



(a) Codeur élémentaire

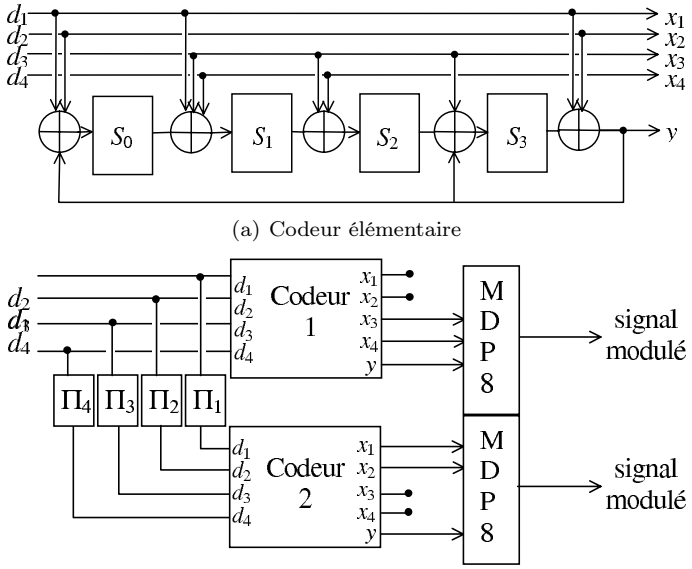


(b) Codeur TMCT

Figure 10.5 – Construction d'une turbo-MAQ-16 selon la méthode décrite dans [10.4]. Efficacité spectrale $\eta = 2$ bit/s/Hz.

Les performances de correction de ces deux TMCT sur canal gaussien sont présentées dans les figures 10.7 et 10.8. Aux taux d'erreurs forts et moyens, elles sont proches de celles affichées par le schéma de Robertson et Wörz ; en revanche, l'utilisation d'entrelaceurs opérant sur les bits plutôt que sur les symboles a permis d'améliorer de manière significative le comportement à faible taux d'erreurs.

Les TMCT conduisent à d'excellentes performances de correction sur canal gaussien, car elles constituent une approche *ad hoc* des modulations turbocodées. Elles présentent, en revanche, le désavantage majeur d'une flexibilité très limitée : un nouveau code doit être défini pour chaque rendement de codage et chaque modulation considérés. Cet inconvénient est lourd dans tout sys-



(a) Codeur élémentaire

(b) Codeur TMCT.

Figure 10.6 – Construction d'une turbo-MDP-8 selon la méthode décrite dans [10.4]. Efficacité spectrale $\eta = 2$ bit/s/Hz

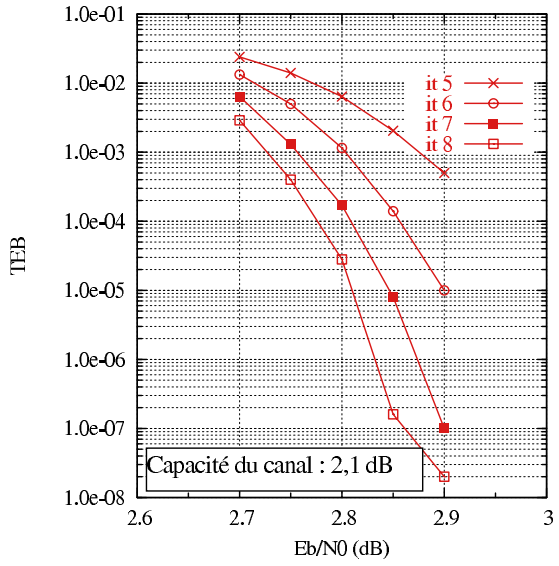


Figure 10.7 – TEB en fonction du rapport signal à bruit E_b/N_0 de la turbo-MAQ-16 en treillis à 16 états utilisant le code CSR de la figure 10.5. Transmission sur canal gaussien. Efficacité spectrale $\eta = 2$ bit/s/Hz. 2×16384 bits d'information. Algorithme de décodage *MAP*. Courbes extraites de [10.4].

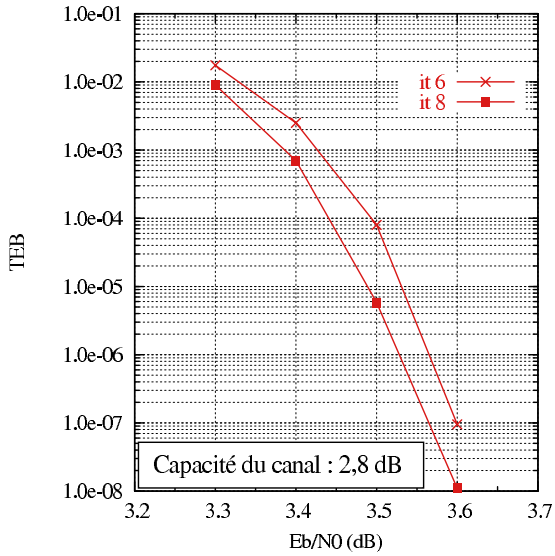


Figure 10.8 – TEB en fonction du rapport signal à bruit E_b/N_0 de la turbo-MDP-8 en treillis à 16 états utilisant le code CSR de la figure 10.6. Transmission sur canal gaussien. Efficacité spectrale $\eta = 2$ bit/s/Hz. 4×4096 bits d'information. Algorithme de décodage *MAP*. Courbes extraites de [10.5].

tème pratique requerrant un certain degré d'adaptabilité. D'autre part, si elles constituent une réponse quasi-optimale au problème des modulations codées pour le canal gaussien, leur comportement sur des canaux à évanouissements tel que les canaux de Rayleigh conduit à des performances médiocres [10.6].

10.2 Les modulations turbocodées pragmatiques

L'approche dite *pragmatique* a été chronologiquement la première mise en œuvre. Elle a été introduite par Le Goff *et al* [10.7] en 1994. Cette technique tient son nom de ses similarités avec la technique d'association code convolutif/modulation proposée par Viterbi [10.8] comme solution alternative aux MCT d'Ungerboeck. Les fonctions de codage et de modulation sont traitées de manière indépendante, sans optimisation conjointe. Elle utilise un « bon » turbocode, un codage binaire à signal de la modulation qui minimise la probabilité d'erreur binaire en sortie du canal (codage de Gray) et associe les deux fonctions par le biais d'une opération de poinçonnage et de multiplexage (ou *mapping*) pour adapter l'ensemble à l'efficacité spectrale visée. Les figures 10.9 et 10.10 présentent le schéma général de principe de l'émetteur et du récepteur pour l'association pragmatique d'un turbocode et d'une modulation à $Q = 2^q$états.

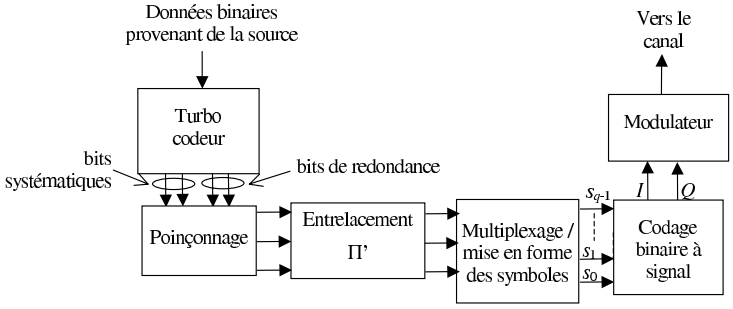


Figure 10.9 – Schéma de principe de l'émetteur dans le cas de l'association pragmatique d'un turbocode et d'une modulation à $Q = 2^q$ états.

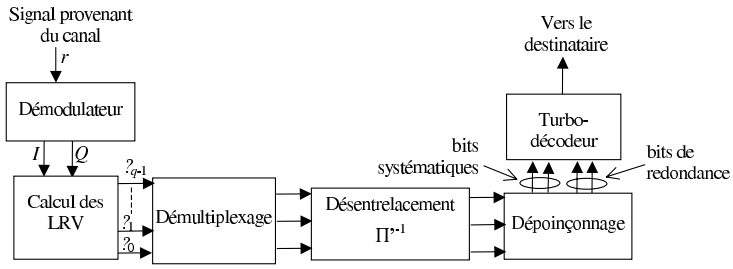


Figure 10.10 – Schéma de principe du récepteur pour le schéma de modulation turbocodée pragmatique de la figure 10.9.

Avec cette approche pragmatique des modulations turbocodées, le codeur et le décodeur utilisés sont des turbocodeur et décodeur standard, identiques pour l'ensemble des rendements de codage et des modulations considérées. Si la taille des blocs de données transmis est variable, un simple paramétrage de la fonction de permutation du code doit permettre de s'adapter aux différentes tailles.

Lorsque le rendement de codage visé est supérieur au rendement naturel du turbocode, l'opération de poinçonnage permet d'effacer, c'est-à-dire de ne pas transmettre, certains bits codés. En pratique, pour des raisons de commodité de réalisation matérielle, le motif de poinçonnage est périodique ou quasi-périodique. Si possible, seuls les bits de parité sont poinçonnés. En effet, un poinçonnage des bits systématiquesentraîne une dégradation rapide du seuil de convergence de décodage, car ces bits participent au processus de décodage des deux codes, contrairement aux bits de redondance. Lorsque le rendement de codage est élevé, un léger poinçonnage des données peut néanmoins améliorer le comportement asymptotique du système codé.

La présence des fonctions d'entrelacement Π' et Π'^{-1} se justifie par le besoin de décorréler les données à l'entrée du turbo-décodeur. En fait, il est montré dans [10.7] que l'insertion de cet entrelacement n'a pas d'effet significatif sur le

taux d'erreurs en sortie du décodeur dans le cas d'une transmission sur canal gaussien. En revanche, dans le cas de canaux à évanouissements, l'entrelacement est nécessaire car il s'agit d'éviter que les bits issus d'un même instant de codage n'appartiennent pas à un même symbole émis sur le canal, afin de ne pas être affectés simultanément par un évanouissement. Les études menées dans ce domaine [10.6-10.9] ont montré que dans le cas des canaux à évanouissements, les meilleures performances sont obtenues en utilisant des entrelaceurs indépendants au niveau bit. Cette technique est appelée *modulation codée à bits entrelacés* ou *BICM* (*Bit-Interleaved Coded Modulation*). Lorsque l'entrelaceur est placé au niveau des symboles de modulation, l'ordre de diversité de la modulation codée est égal au nombre minimal de symboles différents entre deux séquences codées et modulées. Avec des entrelaceurs indépendants placés au niveau de chaque sortie du codeur, il peut idéalement atteindre la distance de Hamming du code. En conséquence, les schémas de transmission utilisant le principe *BICM* présentent en pratique de meilleures performances sur les canaux à évanouissements que les TMCT.

Le code et la modulation n'étant pas conjointement optimisés, à la différence d'un schéma TMCT, on choisit un codage binaire à signal des points de la constellation qui minimise le taux d'erreurs binaires moyen à l'entrée du décodeur. Un codage de Gray, lorsqu'il est envisageable, satisfait cette condition. Par souci de simplicité de mise en œuvre du modulateur et du démodulateur, dans le cas des modulations d'amplitude en quadrature ou MAQ carrées (q pair), les voies en phase et en quadrature, I et Q, sont codées indépendamment.

Dans la figure 10.9, le bloc « Multiplexage / mise en forme des symboles » a pour rôle de répartir les bits codés, après entrelacement pour les canaux à évanouissements, dans les symboles de modulation. Ce bloc, charnière entre le code et la modulation, autorise un certain niveau d'ajustement de la modulation codée en fonction des performances visées. Cet ajustement est possible car le code et la modulation ne jouent pas le même rôle vis-à-vis de l'ensemble des bits transmis.

D'une part, on peut distinguer en sortie du codeur deux familles distinctes de bits codés : les bits systématiques et les bits de redondance. Ces deux familles de bits jouent un rôle différent dans le processus de décodage : les bits systématiques, directement issus de la source, sont utilisés par les deux décodeurs élémentaires en réception tandis que les bits de redondance, issus d'un des deux codeurs élémentaires, sont utilisés par l'un des deux décodeurs seulement.

D'autre part, les éléments binaires contenus dans un symbole modulé ne sont pas, en général, tous protégés de manière identique par la modulation. Par exemple, dans le cas des modulations MDP ou MAQ avec codage de Gray, seules les modulations à deux ou quatre points offrent le même niveau de protection à tous les bits d'un même symbole. Pour les modulations d'ordre supérieur, certains bits sont mieux protégés que d'autres.

À titre d'illustration, considérons une modulation MAQ-16, codée indépendamment et de manière analogue sur les voies en phase et en quadrature par un codage de Gray. La projection de cette modulation sur chacune des voies est une modulation par déplacement d'amplitude (MDA) à 4 états (voir figure 10.11).

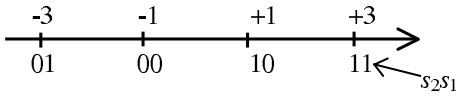


Figure 10.11 – Diagramme des signaux d'une modulation MDA-4 avec codage de Gray.

On peut montrer que, pour une transmission sur un canal gaussien, les probabilités d'erreur sur les positions binaires s_1 et s_2, connaissant le symbole transmis (± 3 ou ± 1), s'expriment sous la forme :

$$\begin{aligned}
P_{\text{eb}}(s_2 \mid \pm 3) &= \frac{1}{2}\text{erfc}\left(\frac{3}{\sigma\sqrt{2}}\right) \\
P_{\text{eb}}(s_2 \mid \pm 1) &= \frac{1}{2}\text{erfc}\left(\frac{1}{\sigma\sqrt{2}}\right) \\
P_{\text{eb}}(s_1 \mid \pm 3) &= \frac{1}{2}\text{erfc}\left(\frac{1}{\sigma\sqrt{2}}\right) - \frac{1}{2}\text{erfc}\left(\frac{5}{\sigma\sqrt{2}}\right) \approx \frac{1}{2}\text{erfc}\left(\frac{1}{\sigma\sqrt{2}}\right) \\
P_{\text{eb}}(s_1 \mid \pm 1) &= \frac{1}{2}\text{erfc}\left(\frac{3}{\sigma\sqrt{2}}\right) + \frac{1}{2}\text{erfc}\left(\frac{1}{\sigma\sqrt{2}}\right) \approx \frac{1}{2}\text{erfc}\left(\frac{1}{\sigma\sqrt{2}}\right)
\end{aligned} \tag{10.1}$$

où $erfc$ représente la fonction d'erreur complémentaire et σ^2 désigne la variance de bruit sur le canal. On observe que la position binaire s_2 est en moyenne mieux protégée par la modulation que la position s_1.

Il est par conséquent possible de définir plusieurs stratégies de construction des symboles de la modulation en associant en priorité les bits systématiques ou redondants aux positions les mieux protégées par la modulation. Deux stratégies extrêmes peuvent dans tous les cas être définies :

- schéma dit « systématique » : les bits les mieux protégés par la modulation sont associés en priorité aux bits systématiques ;
- schéma dit « redondant » : les bits les mieux protégés par la modulation sont associés en priorité aux bits de redondance.

Des modulations d'ordres plus élevés que la MAQ-16 offrent plus de deux niveaux de protection pour les différentes positions binaires. La modulation MAQ-64, par exemple, affiche trois niveaux de protections différents, si les voies en phase et en quadrature sont codées indépendamment et de manière analogue en utilisant un code de Gray. Dans ce cas, d'autres schémas intermédiaires, entre les schémas « systématique » et « redondant », peuvent être définis.

Le schéma de réception correspondant à l'émetteur de la figure 10.9 est décrit dans la figure 10.10. Un turbo-décodeur standard est utilisé, ce qui né-

cessite le calcul d'une estimation pondérée de chacun des bits contenus dans les symboles en sortie du démodulateur avant d'effectuer le décodage.

L'estimation pondérée de chaque bit est obtenue par le calcul du Logarithme du Rapport de Vraisemblance (LRV) défini par :

$$\hat{s}_i = \Lambda(s_i) = \frac{\sigma^2}{2} \ln\left(\frac{\Pr(\, s_i = 1 \mid \, r)}{\Pr(\, s_i = 0 \mid \, r)}\right) = \frac{\sigma^2}{2} \ln\left(\frac{\Pr(\, s_i = 1 \mid \, I, Q)}{\Pr(\, s_i = 0 \mid \, I, Q)}\right) \tag{10.2}$$

Le signe du LRV fournit la décision binaire sur s_i (0 si $\Lambda(s_i) \leq 0$, 1 sinon) et sa valeur absolue représente le poids, c'est-à-dire la fiabilité, associée à la décision. Dans le cas d'une transmission sur canal gaussien, $\hat{s}_i$ peut s'écrire :

$$\hat{s}_i = \frac{\sigma^2}{2} \ln\left(\frac{\displaystyle\sum_{s \in S_{i,1}} \exp\left(-\frac{d_{r,s}^2}{2\sigma^2}\right)}{\displaystyle\sum_{s \in S_{i,0}} \exp\left(-\frac{d_{r,s}^2}{2\sigma^2}\right)}\right) \tag{10.3}$$

où $S_{i,1}$ et $S_{i,0}$ représentent les ensembles de points s de la constellation tels que le i^{me} bit s_i est égal à 1 ou à 0, et $d_{r,s}$ est la distance euclidienne entre le symbole reçu r et le point de la constellation considéré s.

En pratique, l'approximation Max-Log est couramment utilisée pour simplifier le calcul des LRV :

$$\ln(\exp(a) + \exp(b)) \approx \max(a, b) \tag{10.4}$$

et les estimations pondérées sont calculées comme :

$$\hat{s}_i = \frac{1}{4}\left(\min_{s \in S_{i,0}}(d_{r,s}^2) - \min_{s \in S_{i,1}}(d_{r,s}^2)\right). \tag{10.5}$$

On notera qu'il n'est pas nécessaire de connaître la variance de bruit sur le canal, lorsque cette simplification est utilisée.

Les autres blocs du récepteur réalisent les opérations inverses des blocs de la figure 10.9. L'opération de dépoinçonnage correspond à l'insertion d'un LRV égal à 0, c'est à dire d'une décision de fiabilité nulle, à l'entrée du décodeur pour l'ensemble des bits non transmis.

L'approche pragmatique des modulations turbocodées permet d'obtenir des performances très proches des TMCT. Les figures 10.12 et 10.13 montrent les performances de deux modulations turbocodées pragmatiques utilisant le turbocode double-binaire à 16 états présenté en section 7.5.2.2 pour des conditions de transmission semblables à celles ayant conduit à l'obtention des courbes des figures 10.3 et 10.4. On observe qu'après 8 itérations de décodage, les performances des deux familles de modulations turbocodées sont équivalentes jusqu'à des TEB de 10^{-4} à 10^{-5}. Le meilleur comportement de la solution pragmatique

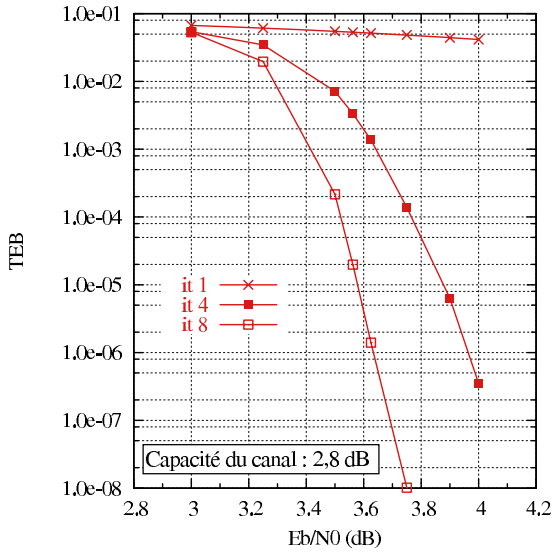


Figure 10.12 – TEB en fonction du rapport signal à bruit E_b/N_0 d'une turbo-MDP-8 pragmatique utilisant un code double-binaire à 16 états. Transmission sur canal gaussien. Efficacité spectrale $\eta = 2$ bit/s/Hz. Blocs de 10 000 bits d'information, 5000 symboles modulés. Algorithme de décodage *MAP*. Schéma « systématique ».

aux taux d'erreurs plus faibles est dû, d'une part à l'utilisation de codes élémentaires à 16 états et d'autre part à une conception soignée de l'entrelaceur du turbocode.

Les courbes de la figure 10.14 mettent en évidence l'influence de la stratégie de construction des symboles sur les performances de la modulation turbocodée. Elles montrent le comportement de l'association d'un turbocode double-binaire à seize états et d'une MAQ-16 codée indépendamment sur les voies en phase et en quadrature à l'aide du code de Gray. Les deux stratégies extrêmes de construction des symboles décrites précédemment y ont été simulées. La taille des blocs, 54 octets, et les rendements simulés, 1/2 et 3/4, sont représentatifs d'applications concrètes dans le secteur des technologies sans fil (norme IEEE 802.16, *Wireless Metropolitan Area Network*). La figure 10.14 affiche également les limites théoriques de la transmission étudiée. Ces limites prennent en compte la taille des blocs transmis ainsi que le taux d'erreurs de paquets (TEP) visé. Elles sont obtenues à partir de la valeur de la capacité du canal, à laquelle on ajoute un terme correctif obtenu à l'aide de la borne dite « d'empilement de sphères » (*sphere packing*, voir section 3.3).

On observe qu'aux taux d'erreurs forts et moyens, la convergence du processus itératif de décodage est favorisée par une meilleure protection des bits systématiques. Ce résultat peut être expliqué par le fait que, dans le processus de décodage, chaque donnée systématique est utilisée en entrée des deux décodeurs. Par conséquent, une erreur sur un bit systématique en sortie du canal

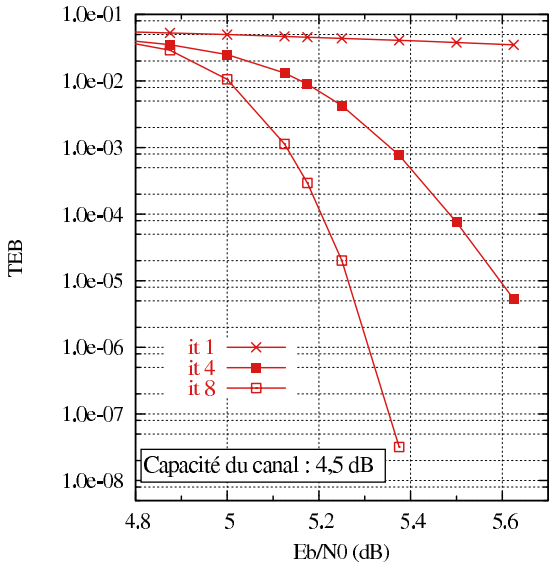


Figure 10.13 – TEB en fonction du rapport signal à bruit E_b/N_0 d'une turbo-MAQ-16 pragmatique utilisant un code double-binaire à 16 états. Transmission sur canal gaussien. Efficacité spectrale $\eta = 3$ bit/s/Hz. Blocs de 15 000 bits d'information, 5000 symboles modulés. Algorithme de décodage *MAP*. Schéma « systématique ».

provoque une erreur sur l'entrée des deux décodeurs élémentaires, alors qu'une redondance erronée n'affecte l'entrée que d'un des deux décodeurs élémentaires. En conséquence, un renforcement de la protection des bits systématiques profite simultanément aux deux décodeurs élémentaires.

L'écart de performance entre les deux schémas est d'autant plus important que la proportion de bits de redondance transmis est élevée, c'est-à-dire que le rendement de codage est faible. A titre d'exemple, à un taux d'erreurs binaires de 10^{-4}, on observe un écart de 0,9 dB pour un rendement de codage $R = 1/2$, et de 0,2 dB pour $R = 3/4$.

Pour les faibles et très faibles taux d'erreurs, le schéma favorisant la protection de la redondance procure les meilleures performances. Ce comportement est difficile à mettre en évidence par simulation pour les rendements les plus faibles, car le point supposé de croisement des courbes est situé à un taux d'erreurs difficile à obtenir par simulation ($TEP \approx 10^{-8}$ pour $R = 1/2$). L'interprétation de ce résultat fait appel à l'analyse des chemins erronés dans les treillis à fort rapport signal à bruit. Nous avons observé que, dans la majorité des cas, les séquences erronées contiennent un nombre plutôt faible de bits systématiques erronés et un nombre plutôt élevé de bits de redondance erronés. Autrement dit, les séquences erronées présentent en général un poids d'entrée faible. En particulier, les chemins erronés en question correspondent pour la plupart à des motifs d'erreurs rectangulaires (voir section 7.3.2). Il en résulte

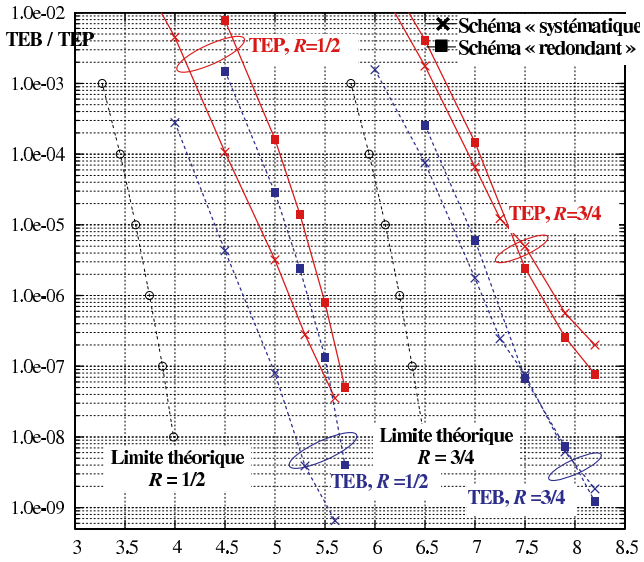


Figure 10.14 – Performance en taux d'erreurs binaires (TEB) et taux d'erreurs paquets (TEP) de l'association pragmatique d'une MAQ-16 et d'un turbocode double-binaire 16 états, pour la transmission de blocs de 54 octets sur canal gaussien. Rendements de codage 1/2 et 3/4. Décodage Max-Log-*MAP*, entrées du décodeur codées sur 6 bits, 8 itérations de décodage.

que, du point de vue du comportement asymptotique de la modulation turbocodée, il est préférable d'assurer une meilleure protection des bits de parité.

Les courbes reportées en figure 10.14 ont été obtenues à l'aide de l'algorithme de décodage simplifié Max-Log-*MAP*, en utilisant des données quantifiées sur 6 bits à l'entrée du décodeur, conditions correspondant à une réalisation matérielle du décodeur. Malgré ces contraintes, les performances obtenues montrent des écarts de seulement 1 dB à un TEP de 10^{-4} et 1,5 dB à un TEP de 10^{-6} avec les limites théoriques.

10.3 Bibliographie

[10.1] P. Robertson and T. Wörz, « Coded modulation scheme employing turbo codes », *Electronics Letters*, vol. 31, n ° 2, pp. 1546-47, Aug. 1995.

[10.2] P. Robertson and T. Wörz, « Bandwidth-efficient turbo trellis-coded modulation using punctured component codes », *IEEE Journal on Selected Areas in Communications*, vol. 16, n ° 2, pp. 206-218, Feb. 1998.

[10.3] G. Ungerboeck, « Channel coding with mutilevel/phase signals », *IEEE Transactions on Information Theory*, vol. IT-28, n ° 1, pp. 55-67, Jan. 1982.

[10.4] S. Benedetto, D. Divsalar, G. Montorsi and F. Pollara, « Parallel

concatenated trellis codes modulation », *International Conference on Communications*, pp. 974-978, ICC'96, Dallas, USA, June 1996.

[10.5] C. Fragouli and R. Wesel, « Bit vs. Symbol interleaving for parallel concatenated trellis coded modulation », *Global Telecommunications Conference*, Globecom'01, pp. 931-935, San Antonio, USA, Nov. 2001.

[10.6] E. Zehavi, « 8-PSK trellis codes for a Rayleigh channel », *IEEE Transactions on Communications*, vol. 40, n ° 5, pp. 873-884, May 1992.

[10.7] S. Le Goff, A. Glavieux, and C. Berrou, « Turbo-codes and high spectral efficiency modulation », *International Conference on Communications*, ICC'94, pp. 645-649, New Orleans, USA, May 1994.

[10.8] A. J. Viterbi, J. K. Wolf, E. Zehavi and R. Padovani, « A pragmatic approach to trellis-coded modulation », *IEEE Communications Magazine*, vol. 27, n ° 7, pp. 11-19, July 1989, .

[10.9] G. Caire, G. Taricco and E. Biglieri, « Bit-Interleaved Coded Modulation », *IEEE Trans. Info. Theory*, vol. 44, n ° 3, pp. 927-946, May 1998.

Chapitre 11

Le principe turbo appliqué à l'égalisation et à la détection

L'invention des turbocodes au début des années 90 a véritablement bouleversé le champ du codage correcteur d'erreur. Des codes relativement simples à construire et à décoder, permettant d'approcher au plus près la limite théorique de Shannon, étaient enfin disponibles. Mais l'impact de cette découverte ne s'est toutefois pas cantonné au seul domaine du codage. Plus généralement, elle a donné naissance à un nouveau paradigme de conception des systèmes de transmission numérique, communément désigné aujourd'hui sous le nom de « principe turbo ». Pour résoudre certains problèmes de traitement de signal *a priori* très complexes, on peut envisager de diviser ces problèmes en une cascade de traitements élémentaires, plus simples à mettre en œuvre. Toutefois, on sait aujourd'hui que la succession unidirectionnelle de ces traitements conduit à une perte d'information. Pour pallier cette sous-optimalité, le principe turbo préconise l'instauration d'un échange d'information probabiliste, « dans les deux sens », entre ces différents traitements. L'intégralité de l'information disponible est alors bien prise en compte dans la résolution du problème global et un consensus entre tous les traitements élémentaires peut être trouvé pour élaborer la décision finale.

L'application du principe turbo à un certain nombre de problèmes classiques en transmission numérique a démontré des gains de performances impressionnants par rapport aux systèmes traditionnels. Son utilisation s'est donc rapidement popularisée au sein de la communauté scientifique. Ce chapitre présente les deux premiers systèmes à avoir bénéficié historiquement de l'application du principe turbo à un contexte autre que le codage correcteur d'erreurs. Le premier système, baptisé turbo-égalisation, itère entre la fonction d'égalisation et une fonction de décodage pour améliorer le traitement de l'interférence entre symboles dans un contexte de transmission sur canaux multi-trajets. Le second, communément appelé turbo-CDMA, exploite le principe turbo pour

améliorer la discrimination entre utilisateurs dans le cas d'une communication radio-mobile entre plusieurs utilisateurs basée sur la technique d'accès multiple par étalement de spectre (*Code Division Multiple Access* en anglais).

11.1 La turbo-égalisation

Les canaux multi-trajets possèdent la particularité de transformer un signal transmis en un ensemble de copies de ce signal. Ainsi, à la sortie du canal, le récepteur ne voit pas uniquement le signal émis mais une superposition de ces différentes copies ou échos. La turbo-égalisation est une technique numérique de réception qui permet de traiter des données détériorées par les canaux de transmission de type multi-trajets. Elle combine le travail d'un égaliseur et d'un décodeur de canal selon le principe turbo. Schématiquement ce système numérique de réception consiste donc en une répétition de la chaîne de traitement égalisation-entrelacement-décodage. Dans un premier temps, l'égalisation réalise une première estimation des données émises. Dans un second temps, celle-ci est transmise au module de décodage qui actualise cette information. Puis l'information actualisée par le décodeur est fournie au module d'égalisation. Ainsi, au fil des itérations, les traitements d'égalisation et de décodage vont mutualiser leurs informations afin d'atteindre les performances d'une transmission sur un canal à trajet unique.

L'objectif de cette section consiste à présenter le principe de la turbo-égalisation et sa mise en œuvre suivant deux versions : la turbo-égalisation selon le critère du Maximum *A Posteriori* (*MAP*) et la turbo-égalisation selon le critère de la Minimisation de l'Erreur Quadratique Moyenne (MEQM). Nous détaillerons les algorithmes associés à ces deux techniques, ainsi que leur complexité respective. Ceci nous amènera à présenter les architectures possibles et à donner des exemples d'implémentation. Enfin, des applications potentielles et existantes pour ces techniques seront exposées.

11.1.1 Canaux multi-trajets et interférence entre symboles

Cette section est consacrée aux transmissions sur canaux multi-trajets dont la particularité est de générer un ou plusieurs échos du signal émis. Physiquement ces échos peuvent par exemple correspondre à des réflexions sur un bâtiment. Les échos ainsi produits viennent se superposer au signal initialement émis et ainsi dégrader la réception. Le modèle de canal discret équivalent permet une représentation mathématiquement simple de ces phénomènes physiques. Pour cela, il suppose que le signal émis est de type discret et que les différents échos sont séparables et sont toujours situés à un multiple du temps symbole. Soit x_i le symbole émis à l'instant discret i, et y_i le symbole reçu à

ce même instant. Le modèle est alors donné par

$$y_i = \sum_{k=0}^{L-1} h_k(i) x_{i-k} + w_i \tag{11.1}$$

où $h_k(i)$ représente l'action du canal (écho) à l'instant i sur un symbole émis à l'instant $i-k$. La réponse impulsionnelle du canal à l'instant i s'écrit alors de la manière suivante sous forme de transformée en z :

$$h(z) = \sum_{k=0}^{L-1} h_k(i) z^{i-k} \tag{11.2}$$

La réponse impulsionnelle du canal est supposée de durée finie (L coefficients), ce qui constitue une hypothèse réaliste en pratique dans la plupart des scénarios.

L'équation (11.1) montre que d'une manière générale, le symbole reçu y_i est fonction des symboles émis avant, voire après (si le canal introduit un retard de propagation) le symbole d'information x_i considéré à l'instant i. Conformément à ce qui a été introduit au chapitre 2, on dit alors que le signal reçu est entaché d'interférences entre symboles (IES). Si l'on suppose maintenant que le canal de transmissionne varie pas (ou très peu) sur la durée d'un bloc d'information transmis, le modèle (11.1) se simplifie comme suit :

$$y_i = \sum_{k=0}^{L-1} h_k x_{i-k} + w_i \tag{11.3}$$

où l'on a supprimé la dépendance temporelle sur les coefficients du canal discret équivalent. La représentation du canal discret équivalent sous la forme d'un filtre numérique à réponse impulsionnelle finie présentée en figure 11.1 découle directement de (11.3). Les coefficients du filtre sont précisément ceux de la réponse impulsionnelle du canal.

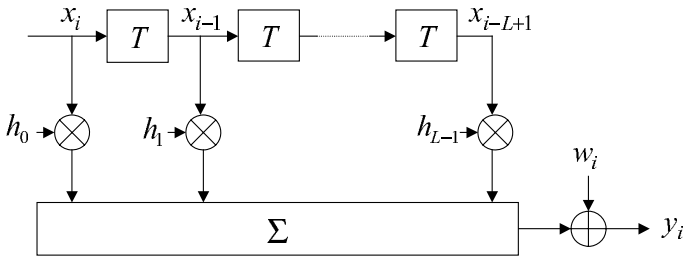
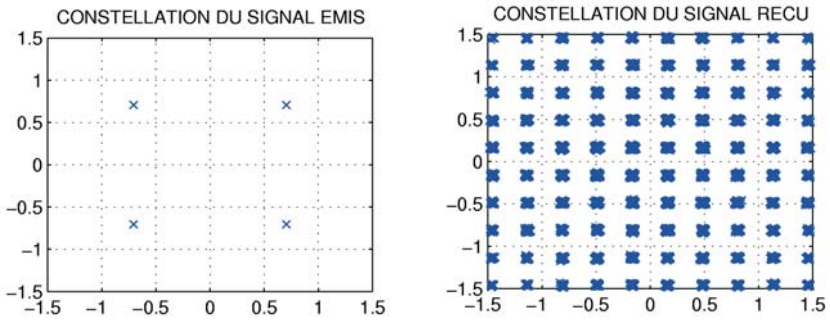


Figure 11.1 – Représentation du canal discret équivalent sous forme de filtre numérique.

L'IES peut constituer un obstacle majeur à l'établissement d'une transmission numérique de bonne qualité, et ce même en présence d'un bruit très faible.

À titre d'illustration, nous avons représenté dans la figure 11.2 la constellation des symboles reçus en sortie d'un canal fortement perturbé par de l'IES, pour un rapport signal à bruit de 20 dB[1], sachant que l'on a émis une séquence de symboles discrets à quatre états de phase (modulation MDP-4). On observe ainsi que l'IES, lorsqu'elle n'est pas traitée par un dispositif adéquat, peut entraîner une dégradation importante du taux d'erreur en réception, et donc de la qualité générale de la transmission.



(a) (b)

Figure 11.2 – Illustration du phénomène d'IES dans le cas d'un canal fortement sélectif en fréquence à 5 trajets, pour un rapport signal sur bruit de 20 dB.

Étudions maintenant les caractéristiques d'un canal multi-trajets dans le domaine fréquentiel. Nous avons représenté dans la figure 11.3 la réponse fréquentielle du canal générant la constellation présentée dans la figure 11.2. Celle-ci est fortement perturbée par l'IES. On remarque que les fréquences du signal ne seront pas atténuées et retardées de la même façon dans toute la bande de fréquences représentée. Ainsi, un signal possédant une bande W comprise entre 0 et 3 kHz sera distordu par le canal. On parle alors de canal sélectif en fréquence par opposition au canal plat non sélectif en fréquence, pour lequel toutes les fréquences subissent la même distorsion. En résumé, lorsqu'un canal multi-trajets génère de l'interférences entre symboles dans le domaine temporel, il est alors sélectif en fréquence dans le domaine fréquentiel.

On dénombre essentiellement trois techniques différentes pour lutter contre la sélectivité en fréquence des canaux de transmission : les transmissions multiporteuses, l'étalement de spectre et l'égalisation. Nous nous intéressons exclusivement dans ce chapitre à la troisième solution, appliquée ici aux transmissions sur fréquence porteuse unique (transmissions « monoporteuse »). Précisons toutefois que certains des concepts abordés ici se transposent relativement aisément aux systèmes de type multiporteuses (systèmes *Orthogonal Frequency Division Multiplexing*, ou *OFDM*).

[1] Rappelons qu'un rapport signal à bruit de 20 dB correspond à une puissance du signal émis 100 fois supérieure à la puissance du bruit additif sur la liaison.

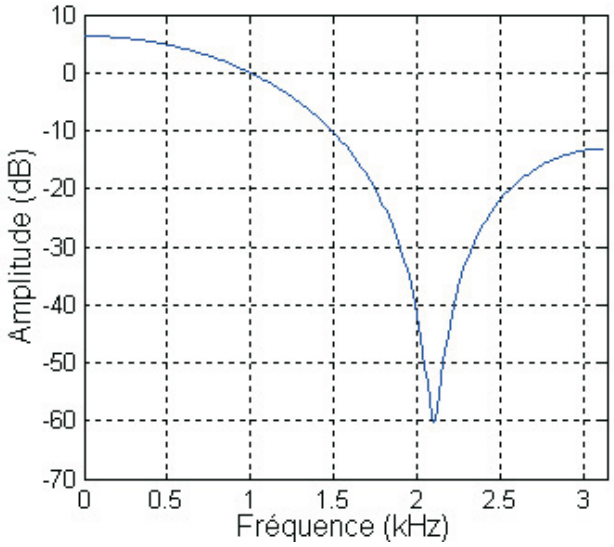


Figure 11.3 – Réponse fréquentielle en amplitude du canal à 5 trajets.

11.1.2 La fonction d'égalisation

Sous sa forme la plus générale, la fonction d'égalisation a pour objectif de restituer une estimation de la séquence de symboles émise à partir de la séquence observée à la sortie du canal, cette dernière étant perturbée à la fois par de l'interférence entre symboles et par un bruit additif, supposé gaussien. On distingue différentes stratégies d'égalisation. Nous nous limitons ici à un survol succinct des principales techniques habituellement mises en œuvre dans les systèmes. Le lecteur intéressé pourra trouver un complément d'information aux chapitres 10 et 11 de [11.1], dans les articles [11.2] et [11.3] ou dans l'ouvrage [11.4] par exemple.

Une première solution, appelée détection de la séquence la plus vraisemblable (*Maximum Likelihood Sequence Detection*, ou *MLSD*), consiste à rechercher la séquence émise la plus probable relativement à l'observation reçue en sortie du canal. On peut montrer que ce critère revient à sélectionner la séquence candidate à distance euclidienne minimale de l'observation, et qu'il minimise ainsi la probabilité d'erreur par séquence, c'est-à-dire la probabilité de retenir une séquence candidate autre que la séquence émise. Une implémentation brutale de ce critère consiste à énumérer l'ensemble des séquences admissibles de manière à calculer la distance entre chaque séquence et l'observation reçue, puis à retenir la séquence la plus proche de cette observation. Toutefois, la complexité de cette approche augmente exponentiellement avec la taille du message transmis, ce qui s'avère rédhibitoire pour une réalisation pratique.

Dans un article célèbre datant de 1972 [11.5], Forney a noté qu'un canal sélectif en fréquence présente un effet de mémoire dont le contenu caractérise son état à un instant donné. Plus précisément, l'état s du canal à l'instant i est parfaitement défini par la connaissance des $L-1$ symboles précédents, ce que l'on note $s = (x_i, \ldots, x_{i-L+2})$. Ce constat s'appuie sur la représentation du canal sous forme de registre à décalage (voir la figure 11.1). L'évolution de l'état du canal au cours du temps peut alors être représentée par un diagramme en treillis comportant M^{L-1} états, où M désigne le nombre de symboles discrets dans l'alphabet de modulation. À titre d'illustration, nous avons représenté dans la figure 11.4 le treillis associé à un canal à $L = 3$ coefficients $\mathbf{h} = (h_0, h_1, h_2)$ dans le cas d'une modulation à deux états de phase.

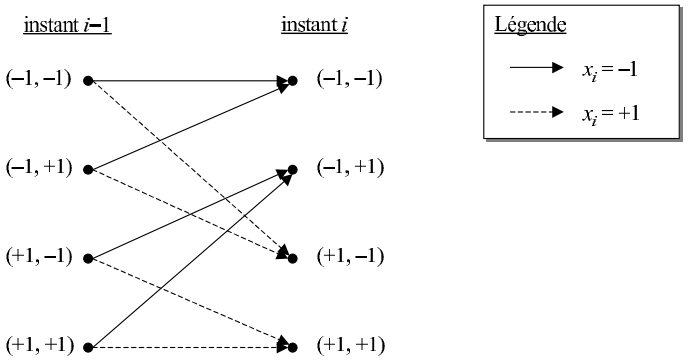


Figure 11.4 – Exemple de représentation en treillis pour une transmission MDP-2 sur un canal sélectif en fréquence à $L = 3$ trajets.

Chaque séquence candidate dessine un chemin unique dans le treillis. La recherche de la séquence à distance euclidienne minimale de l'observation peut alors être réalisée de manière récursive, avec un coût de calcul linéaire en fonction de la taille du message, en appliquant l'algorithme de Viterbi sur le treillis du canal.

L'égaliseur *MLSD* offre de très bonnes performances. En contrepartie, sa complexité de mise en œuvre augmente proportionnellement avec le nombre d'états dans le treillis, et donc exponentiellement avec la durée L de la réponse impulsionnelle du canal et la taille M de l'alphabet de modulation. Son utilisation pratique se limite par conséquent aux transmissions utilisant des modulations à petit nombre d'états (2, 4, voire 8) sur des canaux avec peu d'échos. D'autre part, il convient de noter que cet égaliseur nécessite une estimation préalable de la réponse impulsionnelle du canal afin de construire le treillis. À titre d'information, la solution *MLSD* a été adoptée par de nombreux fabricants pour réaliser l'opération d'égalisation dans les téléphones mobiles pour la norme européenne GSM (*Global System for Mobile communication*).

En présence de modulations à grand nombre d'états ou bien sur des canaux dont la longueur de la réponse impulsionnelle est importante, l'égaliseur *MLSD*

présente un temps de calcul rédhibitoire pour une application temps réel. Une stratégie alternative consiste alors à combattre l'IES à l'aide d'égaliseurs de moindre complexité, mettant en œuvre des filtres numériques.

Dans cette optique, la solution la plus simple consiste à appliquer un filtre linéaire transverse à la sortie du canal. Ce filtre est optimisé de manière à compenser (« égaliser ») les irrégularités de la réponse fréquentielle du canal, dans le but de convertir le canal sélectif en fréquence en un canal équivalent idéalement sans IES, perturbé uniquement par un bruit additif. L'estimation du message transmis est alors effectuée grâce à une simple opération de décision symbole par symbole (détecteur à seuil) à la sortie de l'égaliseur, optimale sur canal à bruit additif blanc gaussien (BABG). Cet égaliseur, décrit dans la figure 11.5, est appelé *égaliseur linéaire* (*Linear Equalizer*, ou *LE* en anglais).

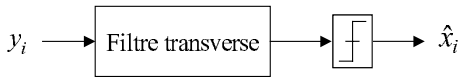


Figure 11.5 – Schéma de principe de l'égaliseur linéaire.

On distingue plusieurs critères d'optimisation pour définir les coefficients du filtre transverse. Le critère optimal consiste à minimiser la probabilité d'erreur symbole en sortie du filtre, mais son application conduit à un système d'équations difficile à résoudre. En pratique, on lui préfère des critères sous-optimaux en terme de performance, mais conduisant à des solutions facilement implémentables, à l'exemple du critère de la minimisation de l'erreur quadratique moyenne (MEQM, ou *Minimum Mean Square Error – MMSE* en anglais) [11.1]. L'égaliseur linéaire transverse MEQM constitue une solution attractive de par sa simplicité. En contrepartie, cet égaliseur souffre d'un problème d'amplification du niveau de bruit sur les canaux fortement sélectifs comportant de profondes atténuations en certains points du spectre fréquentiel.

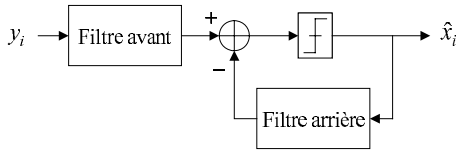


Figure 11.6 – Schéma de principe de l'égaliseur à retour de décision.

En examinant le schéma de principe de l'égaliseur linéaire, on peut remarquer que lorsque l'on prend une décision sur le symbole x_i à l'instant i, on dispose d'une estimation sur les symboles précédents $\hat{x}_{i-1}$, $\hat{x}_{i-2}$, ... On peut donc envisager de reconstruire l'interférence (causale) apportée par ces données et donc de l'évincer, ceci afin d'améliorer la prise de décision. L'égaliseur qui découle de ce raisonnement est appelé *égaliseur à retour de décision* (*Decision Feedback Equalizer*, ou *DFE*). Le schéma de principe du dispositif est décrit dans la figure 11.6. Ce dernier se compose d'un filtre avant, chargé de conver-

tir la réponse impulsionnelle du canal en une réponse purement causale, suivi d'un organe de décision et d'un filtre arrière, chargé d'estimer l'interférence résiduelle en sortie du filtre avant afin de la retrancher par le biais d'un boucle de contre-réaction.

En règle générale, l'égaliseur *DFE* présente des performances supérieures à celle de l'égaliseur linéaire, en particulier sur les canaux fortement sélectifs en fréquence. En contrepartie, cet égaliseur est non linéaire par nature, du fait de la présence de l'organe de décision dans la boucle de retour, ce qui peut donner lieu à un phénomène de propagation d'erreur (particulièrement à faible rapport signal sur bruit) lorsque certaines des données estimées sont incorrectes. En pratique, les coefficients des filtres sont généralement optimisés suivant le critère MEQM, en supposant que les données estimées sont égales aux données émises, et ce afin de simplifier les calculs (voir le chapitre 10 de [11.1] par exemple).

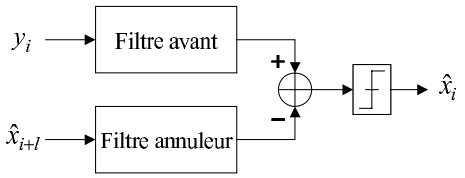


Figure 11.7 – Schéma de principe de l'annuleur d'interférence.

Si l'on suppose maintenant que l'on dispose d'une estimation $\hat{x}_{i+l}$ sur les données émises à la fois antérieures ($l < 0$) et postérieures ($l > 0$) au symbole considéré à l'instant i, on peut alors envisager de retrancher l'intégralité de l'IES à la sortie du canal. La structure d'égalisation obtenue est appelée *annuleur d'interférence* (*Interference Canceller*, ou *IC*) [11-6,11-7]. Elle est décrite dans la figure 11.7. Cette structure est constituée de deux filtres transverses numériques, de réponses impulsionnelles finies : un filtre avant (filtre adapté au canal) visant à maximiser le rapport signal à bruit avant la décision, et un filtre annuleur, en charge de reconstruire l'IES présente en sortie du filtre adapté. Signalons que par construction, le coefficient central du filtre annuleur est nécessairement nul afin d'éviter la soustraction du signal utile. Sous réserve que les données estimées $\hat{x}_{i+l}$ soient égales aux données émises, on peut montrer que cet égaliseur élimine toute l'IES, sans remontée du niveau de bruit. On atteint ainsi la borne du filtre adapté, qui représente ce que l'on peut faire de mieux avec un égaliseur sur un canal sélectif en fréquence. Bien évidemment, on ne connaît jamais *a priori* les données émises en pratique. Toute la difficulté consiste alors à construire une estimation des données qui soit suffisamment fiable pour conserver des performances proches de l'optimal.

Toutes les structures d'égaliseurs présentées jusqu'à présent ne tiennent pas compte de la présence d'un éventuel code correcteur d'erreur à l'émission. Voyons maintenant de quelle façon on peut combiner au mieux les fonctions d'égalisation et de décodage pour améliorer les performances globales du récepteur.

11.1.3 Combiner égalisation et décodage

La majorité des systèmes de transmission numérique monoporteuse opérant sur des canaux sélectifs en fréquence intègrent une fonction de codage correcteur d'erreur avant l'étape de modulation proprement dite à l'émission. Le code correcteur d'erreur est traditionnellement inséré pour combattre les erreurs occasionnées par le bruit additif sur la liaison. D'autre part, couplé à une fonction d'entrelacement judicieusement construite, le codeur offre également un degré de protection supplémentaire face aux évanouissements de puissance occasionnés par le canal, lorsque les caractéristiques de ce dernier varient au cours du temps. Il a été vu dans la section précédente qu'indépendamment de la nature de l'égaliseur employé, l'IES entraîne systématiquement une perte de performance par rapport à un canal BABG non sélectif. La présence du codeur peut alors être exploitée pour réduire cet écart de performance, en tirant profit du gain de codage en réception.

Nous allons nous intéresser dans la suite de cette section au système de transmission représenté dans la figure 11.8. Les opérations de modulation et de transmission sur le canal sont ici représentées en bande de base équivalente, de manière à faire abstraction du passage sur fréquence porteuse.

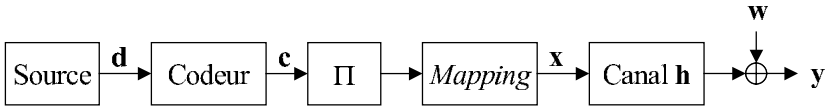


Figure 11.8 – Modélisation en bande de base du système de transmission considéré.

La source délivre une séquence de Q bits d'information, $\mathbf{d} = (d_0, \ldots, d_{Q-1})$. Ce message est protégé par un code convolutif de rendement R, pour fournir une séquence $\mathbf{c} = (c_0, \ldots, c_{P-1})$ de $P = Q/R$ bits codés. Les bits codés sont ensuite entrelacés suivant une permutation Π, puis finalement convertis par groupe de m bits successifs en symboles complexes discrets choisis dans un alphabet à $M = 2^m$ éléments, que l'on notera $\{X_1, \ldots, X_M\}$. C'est l'opération de *mapping*. On obtient ainsi une séquence de $N = P/m$ symboles complexes : $\mathbf{x} = (x_0, \ldots, x_{N-1})$. Par la suite, le vecteur de m éléments binaires codés et entrelacés associé au symbole x_i à l'instant i sera noté $(x_{i,1}, \ldots, x_{i,j}, \ldots, x_{i,m})$. Les symboles émis sont des variables aléatoires discrètes de moyenne nulle et de variance $\sigma_x^2 = E\{|x_i|^2\}$.

Ce schéma d'émission porte le nom de « modulation codée à entrelacement binaire » (ou *BICM*, pour *Bit Interleaved Coded Modulation*). On le retrouve aujourd'hui dans de nombreux systèmes : le standard de téléphonie mobile GSM, par exemple. Dans le cas du transport de la voix à 13 kbits/s (canal TCH/FS), les spécifications de l'interface radio indiquent qu'en sortie du codeur de parole, les bits les plus sensibles (bits de classe 1A et 1B) sont protégés par un code convolutif de rendement $R = 1/2$ et de polynômes générateurs (33,23) en octal [11.8]. Le message codé est ensuite entrelacé sur 8 paquets (ou *bursts*) consécu-

tifs, pour bénéficier d'un effet de diversité en présence d'évanouissements, puis finalement modulé suivant une forme d'onde du type *Gaussian Minimum Shift Keying* (*GMSK*).

Si l'on revient maintenant au scénario de la figure 11.8, la séquence $\mathbf{x}$ est transmise au sein d'un canal sélectif en fréquence à L coefficients, de réponse impulsionnelle discrète $\mathbf{h} = (h_0, \ldots, h_{L-1})$. Le signal résultant est ensuite perturbé par un vecteur $\mathbf{w}$ d'échantillons de BABG, complexes, centrés, de variance $\sigma_w^2 = E\{|w_i|^2\}$. La séquence bruitée observée en sortie du canal est finalement notée $\mathbf{y} = (y_0, \ldots, y_{N-1})$, l'expression de l'échantillon y_i à l'instant i étant donnée par l'équation (11.3).

Dans ce contexte, le problème qui se pose au concepteur du récepteur est le suivant : comment combiner au mieux égalisation et décodage, de manière à ce que chaque traitement bénéficie du résultat de l'autre traitement ?

En réponse à cette question, la théorie de l'estimation nous informe que pour minimiser la probabilité d'erreur dans ce cas de figure, les opérations d'égalisation et décodage doivent être réalisées de manière conjointe, suivant le critère du maximum de vraisemblance. Conceptuellement, réaliser le récepteur optimal revient alors à appliquer l'algorithme de Viterbi par exemple, sur un « super-treillis » prenant en compte simultanément les contraintes imposées par le code, le canal et l'entrelaceur. Le « super-treillis » présente cependant un nombre d'états qui augmente de manière exponentielle avec la taille de l'entrelaceur, ce qui exclut une réalisation pratique du récepteur optimal. Il est donc légitime de s'interroger sur la faisabilité d'un tel récepteur en l'absence d'entrelaceur. Historiquement, cette question s'est notamment posée dans le cadre des transmissions numériques dans la bande vocale (modems téléphoniques). Ces systèmes mettent en effet en œuvre un codage correcteur d'erreur dans l'espace euclidien (modulations codées en treillis), sans entrelacement, et le canal téléphonique constitue un exemple typique de canal sélectif en fréquence et invariant dans le temps. En supposant un codeur à S états, une constellation de M points et un canal discret à L coefficients, les études menées dans ce contexte ont montré que le « super-treillis » correspondant comporte exactement $S(M/2)^{L-1}$ états [11.9]. Il est alors facile de vérifier qu'en dépit de l'absence d'entrelaceur, la complexité du récepteur optimal devient là encore rapidement prohibitive, dès lors que l'on souhaite transmettre un débit important d'information (avec des modulations à grand nombre d'états) ou que l'on se trouve confronté à un canal avec des retards importants.

Pour pallier la trop grande complexité du récepteur optimal, la solution couramment adoptée en pratique consiste à réaliser les opérations d'égalisation et décodage de manière disjointe, séquentiellement dans le temps. Si l'on reprend l'exemple du GSM, les données reçues sont ainsi tout d'abord traitées par un égaliseur *MLSD*. La séquence estimée délivrée par l'égaliseur est ensuite transmise, après désentrelacement, à un décodeur de Viterbi. La fonction de permutation joue alors un double rôle dans ce contexte : d'une part combattre les évanouissement lents du canal, mais également disperser les paquets d'erreurs à l'entrée du décodeur. Cette stratégie présente l'avantage de la simplicité

de mise en œuvre, puisque la complexité totale est alors donnée par la somme des complexités individuelles de l'égaliseur et du décodeur. En contrepartie, elle conduit nécessairement à une perte de performance par rapport au récepteur optimal puisque l'opération d'égalisation n'exploite pas l'intégralité de l'information disponible. Plus précisément, l'estimation délivrée par l'égaliseur ne correspondra pas nécessairement à une séquence codée valide car l'égaliseur ne tient pas compte des contraintes imposées par le code. Les performances de la solution disjointe peuvent être améliorées lorsque l'on instaure un passage d'information pondérée (probabiliste) au lieu d'un échange de données binaires entre l'égaliseur et le décodeur. En propageant une mesure de fiabilité sur les décisions de l'égaliseur, le décodeur profite ainsi d'une information supplémentaire pour produire sa propre estimation du message, et l'on bénéficie d'un gain de correction généralement de l'ordre de plusieurs dB (voir par exemple [11.10,11.11] ou le chapitre 3 de [11.12]). Malgré cela, cette solution conserve le désavantage de n'offrir qu'une communication à sens unique entre l'égaliseur et le décodeur.

Existe t-il donc une stratégie réalisant en quelque sorte le meilleur des deux mondes, capable de concilier à la fois les bonnes performances du récepteur optimal conjoint avec la simplicité de mise en œuvre du récepteur sous-optimal disjoint ? Il est aujourd'hui possible de répondre par l'affirmative, grâce à ce que l'on a appelé la « turbo-égalisation ».

11.1.4 Principe de la turbo-égalisation

Le concept de turbo-égalisation a vu le jour au sein des laboratoires de l'ENST Bretagne au début des années 90, sous l'impulsion des résultats spectaculaires obtenus avec les turbocodes. Il est né d'un constat très simple : le schéma de transmission de la figure 11.8 peut être vu comme la concaténation en série de deux codes (chapitre 6), séparés par un entrelaceur, le second code étant formé par la mise en cascade de l'opération de *mapping* avec le canal[2]. Vu sous cet angle, il semblait alors naturel d'appliquer une stratégie de décodage de type « turbo » en réception, c'est-à-dire un échange réciproque et itératif d'information probabiliste (information extrinsèque) entre l'égaliseur et le décodeur. Le premier schéma de turbo-égalisation a été proposé en 1995 par Douillard *et al.* [11.13]. Ce schéma met en œuvre un égaliseur de Viterbi à entrée et sortie pondérées (*Soft Input Soft Output*, ou *SISO*) selon l'algorithme *SOVA* (*Soft Output Viterbi Algorithm*). Le principe a ensuite été repris en 1997 par Bauch *et al*, en substituant à l'égaliseur *SOVA* un égaliseur *SISO* optimal au sens du critère *MAP*, utilisant l'algorithme développé par Bahl *et al.* (algorithme BCJR) [11.14].

[2] Notons qu'à strictement parler, la transmission sur un canal sélectif ne constitue pas une opération de codage en soi, en dépit de son caractère convolutif, car elle n'apporte aucun gain. Bien au contraire, elle ne fait que dégrader les performances. Cette analogie prend néanmoins tout son sens du point de vue du décodage itératif.

Les résultats de simulation ont rapidement démontré la capacité du turbo-égaliseur à s'affranchir totalement de l'IES, sous certaines conditions. Rétrospectivement, ces excellentes performances s'expliquent par le fait que ce schéma de transmission réunit les deux ingrédients clés qui font la force du principe turbo :

1. La mise en œuvre d'un décodage itératif en réception, instaurant un échange d'information probabiliste entre les traitements, dont on sait aujourd'hui qu'à partir d'un rapport signal sur bruit suffisant (appelé « seuil de convergence »), il converge vers les performances du récepteur optimal conjoint après un certain nombre d'itérations.
2. La présence d'un entrelaceur à l'émission, dont le rôle consiste ici pour l'essentiel à casser les paquets d'erreurs en sortie de l'égaliseur (pour éviter un phénomène de propagation d'erreur), et à décorréler au maximum les données probabilistes échangées entre l'égaliseur et le décodeur de canal. Le turbo-égaliseur est alors capable de compenser entièrement la dégradation causée par l'IES, sous réserve que l'entrelaceur soit suffisamment grand et judicieusement construit.

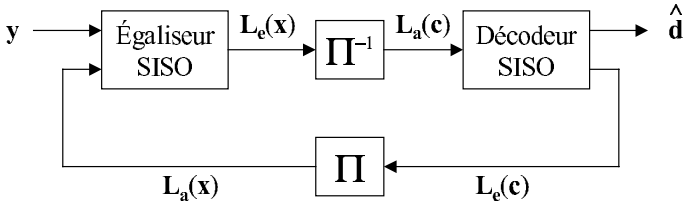


Figure 11.9 – Schéma général du turbo-égaliseur pour un émetteur de type modulation codée avec entrelacement bit (*BICM*).

D'une manière générale, le turbo-égaliseur correspondant au scénario de transmission de la figure 11.8 prend la forme représentée en figure 11.9. Il se compose en premier lieu d'un égaliseur *SISO*, qui prend en entrée à la fois le vecteur $\mathbf{y}$ des données observées à la sortie du canal, et une information probabiliste *a priori* sur l'ensemble des bits codés et entrelacés $x_{i,j}$, notée ici formellement $\mathbf{L}_a(\mathbf{x}) = \{L_a(x_{i,j})\}$. Les informations probabilistes sont propagées sous forme de logarithme de rapport de vraisemblance (LRV), dont nous rappelons ici la définition pour une variable aléatoire binaire d à valeur dans $\{0, 1\}$:

$$L(d) = \ln\left(\frac{\Pr(d=1)}{\Pr(d=0)}\right) \tag{11.4}$$

La notion de LRV est doublement porteuse d'information puisque le signe de la quantité $L(d)$ donne la décision ferme sur d, tandis que sa valeur absolue $|L(d)|$ mesure la fiabilité à accorder à cette décision.

À partir des deux informations $\mathbf{y}$ et $\mathbf{L}_a(\mathbf{x})$, l'égaliseur *SISO* délivre une information extrinsèque notée $\mathbf{L}_e(\mathbf{x}) = \{L_e(x_{i,j})\}$ sur le message binaire codé et

entrelacé. Ce vecteur $\mathbf{L}_e(\mathbf{x})$ est ensuite désentrelacé pour donner une nouvelle séquence $\mathbf{L}_a(\mathbf{c})$, qui constitue l'information *a priori* sur la séquence codée pour le décodeur *SISO*. Ce dernier en déduit alors deux informations : une décision ferme sur le message d'information transmis, notée ici $\hat{\mathbf{d}}$, ainsi qu'une nouvelle information extrinsèque sur le message codé, notée $\mathbf{L}_e(\mathbf{c})$. Cette information est ensuite ré-entrelacée puis renvoyée à l'égaliseur *SISO* où elle est exploitée comme une information *a priori* pour une nouvelle étape d'égalisation à l'itération suivante.

Le schéma de turbo-égalisation que nous venons de présenter correspond à une structure d'émission du type modulation codée à entrelacement bit. Il est toutefois important de noter que le principe de la turbo-égalisation s'applique également dans le cas d'un système mettant en œuvre une modulation codée traditionnelle, c'est-à-dire un système où les opérations de codage et de modulation sont conjointement optimisées, à condition toutefois d'entrelacer les symboles à transmettre avant de les moduler et de les envoyer sur le canal (figure 11.10). La différence principale avec le schéma précédent réside alors dans la mise en œuvre de l'égaliseur et du décodeur *SISO*. En effet, ces derniers échangent des informations probabilistes non plus au niveau binaire mais au niveau symbole, que ce soit sous forme de LRV ou bien directement sous forme de probabilité. Le lecteur intéressé trouvera davantage de détails à ce sujet dans [11.15] par exemple.

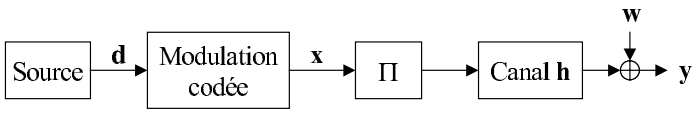


Figure 11.10 – Schéma d'émission dans le cas d'une modulation codée traditionnelle.

En règle générale, le code de canal retenu est un code convolutif et le décodeur de canal utilise un algorithme à entrée et sortie pondérées de type *MAP* (ou ses dérivées dans le domaine logarithmique : Log-*MAP* et Max-Log-*MAP*). Nous ne reviendrons pas sur la mise en œuvre du décodeur puisque ce sujet est traité dans le chapitre 7. Signalons toutefois qu'à la différence des schémas de turbodécodage classiques, le décodeur de canal délivre ici une information extrinsèque non pas sur les bits du message d'information, mais sur les bits du message codé.

En revanche, on distingue différents critères d'optimisation pour la réalisation de l'égaliseur *SISO*, conduisant à des familles distinctes de turbo-égaliseurs. La première, parfois appelée « turbo-détection » et que nous désignerons ici sous le nom de turbo-égalisation *MAP*, met en œuvre un égaliseur optimal au sens du maximum de vraisemblance *a posteriori*. L'égaliseur *SISO* est alors typiquement réalisé grâce à l'algorithme *MAP*. Comme nous le verrons dans la section suivante, cette approche conduit à d'excellentes performances, mais à l'instar de l'égaliseur *MLSD* classique, elle présente un coût de calcul important qui exclut une mise en œuvre pratique dans le cas de modulations à grand nombre

d'états et pour des transmissions sur des canaux présentant des retards temporels importants. On doit alors se tourner vers des solutions alternatives, de moindre complexité mais qui seront nécessairement sous-optimales par nature. Une stratégie envisageable dans ce contexte peut consister à réduire le nombre de branches à examiner à chaque instant dans le treillis. Cette approche est communément appelée « turbo-égalisation *MAP* à complexité réduite ». On connaît différentes méthodes pour arriver à ce résultat, qui seront brièvement présentées dans la section suivante. Une autre solution s'inspire des méthodes classiques en égalisation et met en œuvre un égaliseur *SISO* optimisé suivant le critère de la minimisation de l'erreur quadratique moyenne (MEQM). On obtient ainsi un turbo-égaliseur MEQM, schéma que nous décrirons en section 11.1.6 et qui constitue aujourd'hui une solution très prometteuse pour les transmissions haut débits sur canaux fortement sélectifs en fréquence.

11.1.5 La turbo-égalisation *MAP*

La turbo-égalisation *MAP* correspond au schéma de turbo-égalisation initialement introduit par Douillard *et al*[11.13]. Dans cette section, nous présentons tout d'abord les équations de mise en œuvre de l'égaliseur *SISO*. Les bonnes performances du turbo-égaliseur *MAP* sont illustrées par simulation. Nous introduisons également les solutions de moindre complexité dérivées du critère *MAP*. Enfin, nous examinons les problèmes rencontrés lors d'une implémentation circuit du turbo-égaliseur, ainsi que les applications potentielles de cette technique de réception.

Réalisation de l'égaliseur BCJR-*MAP*

L'égaliseur *MAP*, représenté en figure 11.11, prend en entrée le vecteur $\mathbf{y}$ des symboles discrets observés à la sortie du canal, ainsi qu'une information *a priori* notée $\mathbf{L}_a(\mathbf{x})$ sur les bits codés et entrelacés. Cette dernière provient du décodeur de canal produite à l'itération précédente. Dans le cas particulier de la première itération, on ne dispose généralement pas d'information *a priori* autre que l'hypothèse d'équiprobabilité sur les bits transmis, ce qui nous conduit à poser $L_a(x_{i,j}) = 0$.

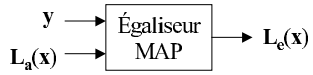


Figure 11.11 – Schéma de principe de l'égaliseur *MAP*.

L'objectif de l'égaliseur *MAP* consiste à évaluer le LRV *a posteriori* $L(x_{i,j})$ sur chaque bit codé et entrelacé $x_{i,j}$, défini comme suit :

$$L(x_{i,j}) = \ln\left(\frac{\Pr(x_{i,j} = 1\,|\mathbf{y}\,)}{\Pr(x_{i,j} = 0\,|\mathbf{y}\,)}\right) \tag{11.5}$$

On peut montrer à l'aide de résultats classiques en théorie de la détection que cet égaliseur est optimal au sens de la minimisation de la probabilité d'erreur symbole. Pour calculer le LRV *a posteriori* $L(x_{i,j})$, nous allons nous aider de la représentation en treillis associée à la transmission sur le canal sélectif en fréquence. En appliquant la relation de Bayes, la relation précédente peut encore s'écrire :

$$L(x_{i,j}) = \ln\left(\frac{\Pr(x_{i,j}=1,\mathbf{y})}{\Pr(x_{i,j}=0,\mathbf{y})}\right) \tag{11.6}$$

Parmi l'ensemble des séquences transmises possibles, chaque séquence candidate dessine un chemin unique dans le treillis. On peut alors évaluer la probabilité conjointe $\Pr(x_{i,j}=0 \text{ ou } 1,\mathbf{y})$ en sommant la probabilité $\Pr(s',s,\mathbf{y})$ de passer par une transition particulière du treillis reliant un état s' à l'instant $i-1$ à un état s à l'instant i, sur l'ensemble des transitions entre les instants $i-1$ et i pour lesquelles le j-ième bit composant le symbole associé à ces transitions vaut 0 ou 1. Ainsi :

$$L(x_{i,j}) = \ln\left(\frac{\sum\limits_{(s',s)/x_{i,j}=1}\Pr(s',s,\mathbf{y})}{\sum\limits_{(s',s)/x_{i,j}=0}\Pr(s',s,\mathbf{y})}\right) \tag{11.7}$$

En adoptant maintenant une démarche similaire à celle présentée dans l'article original de Bahl *et al* [11.16], on peut montrer que la probabilité conjointe $\Pr(s',s,\mathbf{y})$ associée à chaque transition considérée se décompose en un produit de 3 termes :

$$\Pr(s',s,\mathbf{y}) = \alpha_{i-1}(s')\gamma_{i-1}(s',s)\beta_i(s) \tag{11.8}$$

La figure 11.12 illustre les conventions de notation retenues ici. Les quantités

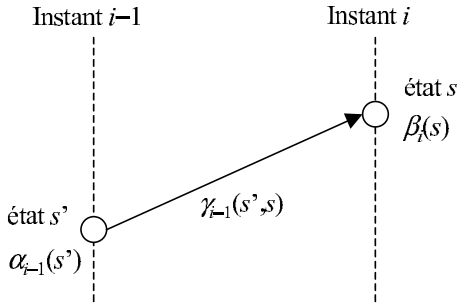


Figure 11.12 – Conventions de notation utilisées pour décrire l'égaliseur *MAP*.

$\alpha_{i-1}(s')$ et $\beta_i(s)$ sont calculables récursivement pour chaque état et à chaque instant dans le treillis, en appliquant les formules de mise à jour suivantes :

$$\alpha_i(s) = \sum_{(s',s)} \alpha_{i-1}(s')\gamma_{i-1}(s',s) \tag{11.9}$$

$$\beta_i(s') = \sum_{(s',s)} \gamma_i(s',s)\beta_{i+1}(s) \tag{11.10}$$

Ces deux étapes portent les noms respectifs de « récursion aller » (*forward recursion*) et « récursion retour » (*backward recursion*). Les sommations portent sur l'ensemble des couples d'états d'indices (s',s) pour lesquels il existe une transition valide entre deux instants consécutifs dans le treillis. La récursion aller utilise la condition initiale suivante :

$$\alpha_0(0) = 1, \quad \alpha_0(s) = 1 \text{ pour } s \neq 0 \tag{11.11}$$

Cette condition traduit le fait que l'état de départ dans le treillis (d'indice 0 par convention) est parfaitement connu. En ce qui concerne la récursion retour, on affecte habituellement le même poids à chaque état en fin de treillis car l'état d'arrivée n'est généralement pas connu *a priori* :

$$\beta_N(s) = \frac{1}{M^{L-1}}, \quad \forall s \tag{11.12}$$

En pratique, on constate que la dynamique des grandeurs $\alpha_{i-1}(s')$ et $\beta_i(s)$ croît au fur et à mesure de la progression dans le treillis. Par conséquent, ces quantités doivent être normalisées à intervalles réguliers afin d'éviter des problèmes de débordement numérique dans les calculs. Une solution naturelle consiste à diviser ces métriques à chaque instant par des constantes K_α et K_β choisies de manière à satisfaire la condition de normalisation suivante :

$$\frac{1}{K_\alpha}\sum_s \alpha_i(s) = 1 \quad \text{et} \quad \frac{1}{K_\beta}\sum_s \beta_i(s) = 1 \tag{11.13}$$

les sommes portant ici sur l'ensemble des états possibles s du treillis à l'instant i.

Pour compléter la description de l'algorithme, il nous reste à développer l'expression du terme $\gamma_{i-1}(s', s)$. Ce dernier est assimilable à une métrique de branche. On peut montrer qu'il se décompose en un produit de deux termes :

$$\gamma_{i-1}(s', s) = \Pr(s|s')P(y_i|s', s) \tag{11.14}$$

La quantité $\Pr(s|s')$ représente la probabilité *a priori* d'emprunter la transition entre l'état s et l'état s', c'est-à-dire la probabilité *a priori* $P_a(X_l) = \Pr(x_i = X_l)$ d'avoir émis à l'instant i le symbole discret X_l de la constellation associé à la branche considérée dans le treillis. Du fait de la présence de l'entrelaceur à l'émission, les bits $x_{i,j}$ composant le symbole x_i peuvent être supposés statistiquement indépendants. Par conséquent, la probabilité $P_a(X_l)$ admet la décomposition suivante :

$$P_a(X_l) = \Pr(x_i = X_l) = \prod_{j=1}^{m} P_a(X_{l,j}) \tag{11.15}$$

où l'on a posé $P_a(X_{l,j}) = \Pr(x_{i,j} = X_{l,j})$, l'élément binaire $X_{l,j}$ prenant la valeur 0 ou 1 en fonction du symbole X_l considéré et de la règle de *mapping*. Au sein du processus itératif de turbo-égalisation, les probabilités *a priori* $P_a(X_{l,j})$ sont déduites de l'information *a priori* disponible à l'entrée de l'égaliseur. À partir de la définition initiale (11.4) du LRV, on peut notamment montrer que la probabilité $P_a(X_{l,j})$ et le LRV *a priori* correspondant $L_a(x_{i,j})$ sont liés par l'expression suivante :

$$P_a(X_{l,j}) = K \exp\left(X_{l,j} L_a(x_{i,j})\right) \quad \text{avec} \quad X_{l,j} \in \{0,1\} \tag{11.16}$$

Le terme K est une constante de normalisation que l'on peut omettre dans les calculs qui suivent, sans pour autant compromettre le résultat final.

La probabilité conditionnelle $\Pr(s|s')$ est donc finalement donnée par :

$$\Pr(s\,|s'\,) = \exp\left(\sum_{j=1}^{m} X_{l,j} L_a(x_{i,j})\right) \tag{11.17}$$

Quant au second terme $P(y_i|s',s)$, il représente tout simplement la vraisemblance $P(y_i|z_i)$ de l'observation y_i relative à l'étiquette de branche z_i associée à la transition considérée. Celle-ci correspond au symbole que l'on aurait observé en sortie du canal en l'absence de bruit :

$$z_i = \sum_{k=0}^{L-1} h_k x_{i-k} \tag{11.18}$$

La séquence de symboles $(x_i, x_{i-1}, \ldots, x_{i-L+1})$ intervenant dans le calcul de z_i se déduit de la connaissance de l'état de départ s' et du symbole d'information X_l associé à la transition $s' \rightarrow s$. En présence de BABG complexe, centré, de variance totale σ_w^2, on obtient :

$$P(y_i\,|s',s\,) = P(y_i\,|z_i\,) = \frac{1}{\pi\sigma_w^2} \exp\left(-\frac{|y_i - z_i|^2}{\sigma_w^2}\right) \tag{11.19}$$

Le facteur $1/\pi\sigma_w^2$ est commun à toutes les métriques de branche et peut donc être omis dans les calculs. D'autre part, on constate ici que le calcul des métriques de branche $\gamma_{i-1}(s',s)$ nécessite à la fois la connaissance de la réponse impulsionnelle du canal discret équivalent et la connaissance de la variance du bruit. Autrement dit, dans le cadre d'une réalisation pratique du système, il faudra faire précéder l'égaliseur *MAP* d'une opération d'estimation des paramètres du canal.

En résumé, après avoir calculé les métriques de branche $\gamma_{i-1}(s',s)$ puis effectué les récursions aller et retour, le LRV *a posteriori* $L(x_{i,j})$ est finalement donné par :

$$L(x_{i,j}) = \ln \frac{\sum\limits_{(s',s)/x_{i,j}=1} \alpha_{i-1}(s')\gamma_{i-1}(s',s)\beta_i(s)}{\sum\limits_{(s',s)/x_{i,j}=0} \alpha_{i-1}(s')\gamma_{i-1}(s',s)\beta_i(s)} \tag{11.20}$$

En réalité et conformément au principe turbo, ce n'est pas cette information *a posteriori* qui est propagée au décodeur *SISO*, mais plutôt l'information extrinsèque. Cette dernière mesure ici la contribution propre de l'égaliseur dans le processus de décision global, en excluant l'information relative au bit considéré en provenance du décodeur à l'itération précédente, c'est-à-dire le LRV *a priori* $L_a(x_{i,j})$. Si l'on développe l'expression de la métrique de branche $\gamma_{i-1}(s',s)$ dans le calcul de $L(x_{i,j})$, on obtient :

$$L(x_{i,j})=\ln\left[\frac{\sum\limits_{(s',s)/x_{i,j}=1}\alpha_{i-1}(s')\exp\left(-\frac{|y_i-z_i|^2}{\sigma_w^2}+\sum\limits_{k=1}^{m}X_{l,k}L_a(x_{i,k})\right)\beta_i(s)}{\sum\limits_{(s',s)/x_{i,j}=0}\alpha_{i-1}(s')\exp\left(-\frac{|y_i-z_i|^2}{\sigma_w^2}+\sum\limits_{k=1}^{m}X_{l,k}L_a(x_{i,k})\right)\beta_i(s)}\right] \tag{11.21}$$

On peut alors factoriser le terme d'information *a priori* relatif au bit $x_{i,j}$ considéré, à la fois au numérateur ($X_{l,j}=1$) et au dénominateur ($X_{l,j}=0$), ce qui donne :

$$\begin{aligned} L(x_{i,j}) = \quad & L_a(x_{i,j}) \\ + \ \ln & \underbrace{\left[\frac{\sum\limits_{(s',s)/x_{i,j}=1}\alpha_{i-1}(s')\exp\left(-\frac{|y_i-z_i|^2}{\sigma_w^2}+\sum\limits_{k\neq j}X_{l,k}L_a(x_{i,k})\right)\beta_i(s)}{\sum\limits_{(s',s)/x_{i,j}=0}\alpha_{i-1}(s')\exp\left(-\frac{|y_i-z_i|^2}{\sigma_w^2}+\sum\limits_{k\neq j}X_{l,k}L_a(x_{i,k})\right)\beta_i(s)}\right]}_{L_e(x_{i,j})} \end{aligned} \tag{11.22}$$

On constate finalement que l'information extrinsèque s'obtient tout simplement en soustrayant l'information *a priori* du LRV *a posteriori* calculé par l'égaliseur :

$$L_e(x_{i,j}) = L(x_{i,j}) - L_a(x_{i,j}) \tag{11.23}$$

Cette remarque conclut la description de l'égaliseur *MAP*. Tel que nous l'avons présenté, cet algorithme se révèle délicat à implémenter sur circuit du fait de la présence de nombreuses opérations de multiplications. Afin de simplifier les calculs, on peut alors envisager de transposer l'ensemble de l'algorithme dans le domaine logarithmique (algorithme Log-*MAP*), l'avantage étant que les multiplications sont alors converties en additions, plus simples à réaliser. Si l'on souhaite réduire davantage la complexité de traitement, on peut également utiliser une version simplifiée (mais sous-optimale), l'algorithme Max-Log-*MAP* (ou Sub-*MAP*). Ces deux variantes ont été présentées dans le contexte des turbocodes au chapitre 7. La dérivation est tout à fait similaire dans le cas de l'égaliseur *MAP*. La référence [11.17] présente une comparaison de performance entre ces différents algorithmes dans un scénario de turbo-égalisation *MAP*. Il en ressort notamment que l'égaliseur Max-Log-*MAP* offre le meilleur compromis performance/complexité lorsque l'estimation du canal est imparfaite.

Exemple de performances

Afin d'illustrer les bonnes performances offertes par la turbo-égalisation *MAP*, nous avons choisi de simuler le scénario de transmission suivant : une source binaire génère des messages de 16382 bits d'information, qui sont ensuite protégés par un code convolutif non récursif non systématique de rendement $R = 1/2$ à 4 états, de polynômes générateurs (5,7) en octal. Deux bits nuls sont insérés à la fin du message afin de forcer la fermeture du treillis dans l'état 0. On obtient ainsi une séquence de 32768 bits codés, qui sont ensuite entrelacés par bloc suivant une permutation générée de manière pseudo-aléatoire, puis convertis en symboles MDP-2. Ces symboles sont transmis sur un canal discret équivalent à 5 trajets, invariant dans le temps, de réponse impulsionnelle :

$$\mathbf{h} = (0.227,\ 0.460,\ 0.688,\ 0.460,\ 0.227)$$

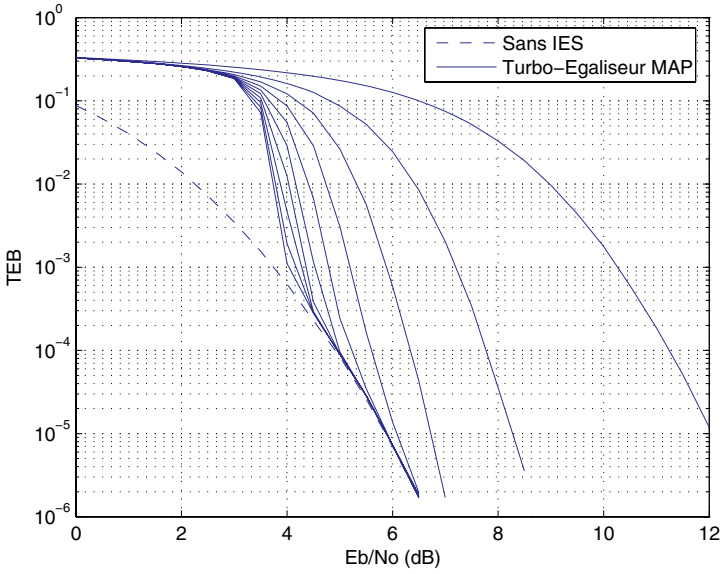


Figure 11.13 – Performance du turbo-égaliseur *MAP* sur le canal Proakis C, avec un code convolutif non récursif non systématique de rendement $R = 1/2$, à 4 états, un entrelaceur pseudo-aléatoire de taille 32768 bits et une modulation MDP-2.

Ce modèle de canal, dit Proakis C, extrait du chapitre 10 de [11.1], est relativement difficile à égaliser. En réception, on met en œuvre 10 itérations du turbo-égaliseur *MAP* précédemment décrit. Le décodeur *SISO* est réalisé à l'aide de l'algorithme BCJR-*MAP*. La figure 11.13 présente le taux d'erreur binaire après décodage, mesuré à chaque itération, en fonction du rapport signal à bruit normalisé E_b/N_0 sur le canal. Nous avons également représenté

les performances obtenues après décodage sur un canal BABG non sélectif en fréquence. Cette courbe représente les performances optimales du système. On constate qu'au delà d'un rapport signal à bruit de 5 dB, le turbo-égaliseur supprime toute l'IES après 5 itérations, et l'on atteint les performances idéales du canal BABG. D'autre part, pour un taux d'erreur binaire cible de 10^{-5}, le processus itératif apporte un gain de l'ordre de 6.2 dB par rapport aux performances du récepteur conventionnel réalisant l'égalisation et le décodage de manière disjointe, données par la courbe à la $1^{\text{ère}}$ itération. Ces performances sont très semblables à celles présentées dans la référence [11.14].

Ces résultats appellent un certain nombre de remarques car l'exemple considéré ici présente un comportement caractéristique des systèmes Turbo. On constate ainsi que le gain apporté par le processus itératif n'apparaît qu'à partir d'un certain rapport signal à bruit (seuil de convergence, égal à 3 dB ici). Au delà de ce seuil, il se produit une convergence rapide du turbo-égaliseur vers les performances asymptotiques du système, données par la probabilité d'erreur après décodage sur canal BABG non sélectif. Pour améliorer les performances globales du système, on peut envisager l'utilisation d'un code correcteur d'erreur plus puissant. L'expérience montre que l'on se heurte alors à la nécessité d'un compromis, à trouver dans le choix du code, entre convergence rapide du processus itératif et bonnes performances asymptotiques du système (à fort rapport signal à bruit). Plus la capacité de correction du code est grande, plus le seuil de convergence est élevé. Signalons à ce sujet qu'il existe aujourd'hui des outils semi-analytiques tels que les diagrammes *EXIT* (*EXtrinsic Information Transfer*) [11.18], permettant de prédire avec précision la valeur du seuil de convergence ainsi que le taux d'erreur après décodage pour un scénario de transmission donné, sous l'hypothèse d'un entrelacement idéal (de taille infinie). Une seconde solution consiste à introduire un effet de contre-réaction au niveau du canal discret équivalent par l'insertion d'un schéma de précodage adéquat à l'émission. La mise en cascade du précodeur avec le canal produit un nouveau modèle de canal, récursif par nature, conduisant à un gain de performance d'autant plus important que l'entrelaceur considéré est de grande dimension. Ce phénomène est connu sous le nom de « gain d'entrelacement » dans la littérature consacrée aux turbocodes en série. Sous réserve de choisir judicieusement le précodeur, on peut alors surpasser les performances du schéma de turbo-égalisation non récursif classique comme cela a été montré dans [11.19] et [11.20].

Complexité du turbo-égaliseur *MAP* et solutions alternatives

La complexité du turbo-égaliseur *MAP* est essentiellement dictée par la complexité de l'égaliseur *MAP*. Or celle-ci augmente proportionnellement avec le nombre de branches à examiner à chaque instant dans le treillis. En considérant une modulation à M états et un canal discret à L coefficients, le nombre total de branches par section du treillis s'élève à $M \times M^{L-1} = M^L$. On voit donc que le coût de traitement associé à l'égaliseur *MAP* augmente exponen-

tiellement avec le nombre d'états de la modulation et la longueur de la réponse impulsionnelle du canal. À titre d'illustration, l'évolution EDGE (*Enhanced Data Rate for GSM Evolution*) du GSM prévoit l'utilisation d'une modulation MDP-8 sur des canaux avec 6 coefficients au maximum, soit un peu plus de 262000 branches à examiner à chaque instant ! La turbo-égalisation *MAP* constitue donc une solution attractive pour les modulations à faible nombre d'états (typiquement MDP-2 et MDP-4) sur des canaux présentant une IES limitée à quelques périodes symboles. Au delà, on doit se tourner vers des solutions de moins complexe mais aussi moins performantes.

Il existe plusieurs façons d'aborder ce problème. Si on se limite à l'utilisation d'égaliseurs dérivés du critère *MAP*, une idée consiste à réduire le nombre de chemins examinés par l'algorithme dans le treillis. Une première approche réalise une troncature de la réponse impulsionnelle du canal pour n'en garder que les $J < L$ premiers coefficients. Le nombre d'états dans le treillis s'en trouve alors diminué. Les termes d'IES ignorés dans la définition des états sont alors pris en compte dans le calcul des métriques de branche, par retour de décision à partir de la connaissance du chemin survivant en chaque état. Cette stratégie porte le nom de *Delayed Decision Feedback Sequence Estimator* (*DDFSE*). Elle offre de bonnes performances sous réserve que la majorité de l'énergie du canal soit concentrée dans ses premiers coefficients, ce qui nécessite en pratique la mise en œuvre d'une opération de préfiltrage à phase minimale. L'application de cette technique à la turbo-égalisation a par exemple été étudiée dans [11.21]. Un raffinement de cet algorithme consiste à regrouper certains états du treillis entre eux, en accord avec les règles de *set-partitioning* définies par Ungerboeck [10.3] pour la conception des modulations codées en treillis. Cette amélioration, baptisée *Reduced State Sequence Estimation* (*RSSE*), englobe l'égaliseur *DDFSE* comme un cas particulier [11.22]. Dans le même état d'esprit, on peut également envisager de retenir plus d'un chemin survivant en chaque état pour améliorer la robustesse de l'égaliseur et éventuellement supprimer l'utilisation du préfiltrage [11.23]. Plutôt que de réduire par troncature le nombre d'états du treillis, il est également envisageable de n'examiner qu'une liste non exhaustive des chemins les plus vraisemblables à chaque instant. L'algorithme résultant est appelé « algorithme M », et son extension à l'égalisation *SISO* a été étudiée dans [11.24]. En tout état de cause, la recherche d'égaliseurs performants à complexité réduite continue à susciter régulièrement de nouvelles contributions.

Toutes les stratégies que nous venons de citer entrent dans la catégorie des « turbo-égaliseurs *MAP* à complexité réduite ». D'une manière générale, ces solutions sont intéressantes lorsque le nombre d'états de la modulation n'est pas trop élevé. En revanche, dans le cas de transmissions haut débit sur des canaux fortement sélectifs en fréquence, il est préférable d'envisager des solutions de type turbo-égaliseur MEQM.

Architectures et applications

Lorsque les systèmes à base de turbo-égalisation *MAP* requièrent un traitement temps réel avec des débits relativement élevés (de l'ordre de plusieurs Mbits/s), une mise en œuvre logicielle n'est pas envisageable. Dans ce cas, on doit faire appel à une réalisation sous forme de circuit spécifique. L'implémentation circuit d'un turbo-égaliseur *MAP* pose des problèmes semblables à ceux rencontrés dans le cadre de la mise en œuvre matérielle d'un turbo-décodeur. Deux solutions architecturales peuvent être envisagées :

- La première fait appel à une réalisation du turbo-décodeur sous forme de *pipeline* par la mise en cascade de plusieurs modules élémentaires, chaque module réalisant une itération de détection et de décodage.
- La seconde fait appel à un module matériel unique, réalisant les itérations successives de manière séquentielle, en re-bouclant sur lui même.

La première architecture présente une latence moins grande, elle est donc mieux adaptée aux applications nécessitant des débits élevés. En revanche, la seconde solution permet d'économiser le nombre de transistors et donc la surface silicium. Afin de réduire davantage la surface utilisée, certains auteurs ont proposé des architectures sophistiquées permettant un partage d'une partie de l'algorithme *SISO* entre l'égaliseur et le décodeur, en dépit de la structure différente des treillis concernés [11.25]. Cette approche permet également de réduire la longueur du chemin critique, et donc la latence globale du système. Ce dernier facteur représente un obstacle majeur à la mise en œuvre pratique de la turbo-égalisation (et plus généralement des systèmes Turbo) car toutes les applications ne tolèrent pas une augmentation du retard de restitution en réception. Le recours à l'électronique analogique permettra peut-être de lever cet obstacle prochainement. Une implémentation analogique d'un turbo-égaliseur *MAP* simplifié a ainsi été reportée en [11.26] avec des résultats prometteurs.

Du point de vue algorithmique, l'application de la turbo-égalisation *MAP* au système GSM a fait l'objet de plusieurs études [11.12, 11.27-11.29]. Le schéma traditionnel de turbo-égalisation doit alors être revu afin de prendre en compte les spécificités du standard (entrelacement inter-trame, différents niveaux de protection appliqués aux bits en sortie du codeur parole, modulation *GMSK*, ...). Les résultats de simulation montrent des gains de performance généralement mitigés, en contrepartie d'une augmentation importante de la complexité du récepteur. Ceci peut s'expliquer en partie par le fait que le système GSM classique ne présente qu'un niveau limité d'IES sur la majorité des canaux de test définis dans le standard. En revanche, l'introduction de la modulation MDP-8dans le cadre de l'évolution *Enhanced Data for GSM Evolution* (EDGE) du GSM augmente sensiblement le niveau d'interférence. Ce scénario semble donc plus propice à l'emploi de la turbo-égalisation, ce qui a suscité plusieurs contributions. En particulier, les auteurs de [11.30][3] ont étudié la mise en œuvre d'un système complet de turbo-égalisation reposant sur un égaliseur *SISO* de type *DDFSE* avec préfiltrage, couplé à un estimateur

[3] Consulter également les références citées dans cet article.

de canal. Ils ont ainsi obtenu des gains de l'ordre de plusieurs dB, en fonction du schéma de modulation et codage considéré, en comparaison avec les performances du récepteur classique. D'autre part, ils ont également mis en évidence le fait que le principe d'égalisation et décodage itératif pouvait être judicieusement exploité dans le cadre du protocole de retransmissionARQ défini dans EDGE (schéma *Incremental Redundancy*) pour améliorer la qualité globale de service en réception.

11.1.6 La turbo-égalisation MEQM

La croissance des débits, en réponse aux besoins actuels en matière de services multimédias, conjuguée à l'engouement pour la mobilité et les infrastructures sans fils, confrontent les récepteurs à de sévères conditions de propagation. Ainsi, si l'on prend l'exemple de l'interface radio du standard WirelessMAN (*Metropolitan Area Network*) 802.16a normalisé par l'IEEE courant 2003 et opérant dans la bande 2-11 GHz, l'IES rencontrée est susceptible de recouvrir jusqu'à 50 durées symboles, voire davantage. L'application de la turbo-égalisation à de tels scénarios implique l'utilisation d'égaliseurs *SISO* de faible complexité. La turbo-égalisation MEQM réalise une solution attractive dans ce contexte.

Par opposition avec les approches décrites dans la section précédente, la turbo-égalisation MEQM consiste pour l'essentiel à substituer à l'égaliseur *MAP* une structure d'égaliseur à base de filtres numériques, optimisée suivant le critère de la minimisation de l'erreur quadratique moyenne[4]. Cette solution présente un certain nombre d'avantages. Tout d'abord, les simulations montrent que le turbo-égaliseur MEQM présente de très bonnes performances en moyenne, parfois très proches des performances offertes par la turbo-égalisation *MAP*. D'autre part, la complexité de l'égaliseur MEQM augmente linéairement (et non exponentiellement) avec la longueur de la réponse impulsionnelle du canal, et ce indépendamment de l'ordre de la modulation. Enfin, comme nous le verrons par la suite, cette approche se prête naturellement bien à une mise en œuvre sous forme adaptative, propice à la poursuite des variations du canal de transmission.

Historiquement, le premier schéma de turbo-égalisation intégrant un égaliseur MEQM a été proposé par Glavieux *et al* en 1997 [11.31-11.33]. Cette contribution originale a véritablement posé les bases de la turbo-égalisation MEQM, notamment en ce qui concerne la gestion des entrées et sorties pondérées au niveau de l'égaliseur. En effet, les égaliseurs classiques à base de filtres numériques ne se prêtent pas naturellement à la manipulation d'information probabiliste. Cette difficulté a été levée par l'insertion d'une opération de conversion binaire M-aire à l'entrée de l'égaliseur, chargée de reconstruire

[4] En toute rigueur, l'utilisation d'égaliseurs optimisés suivant le critère du Forçage à Zéro est également envisageable. Néanmoins, ces égaliseurs introduisent une remontée significative du niveau de bruit sur les canaux sévères et s'avèrent ainsi généralement moins performants que les égaliseurs MEQM.

une estimation souple des symboles émis à partir de l'information *a priori* délivrée par le décodeur. En complément, un module de *demapping SISO* placé à la sortie de l'égaliseur convertit les données égalisées (symboles complexes) en LRV extrinsèques sur les bits codés, qui sont ensuite passés au décodeur. Ce schéma initial reposait sur la mise en œuvre d'une structure d'égalisation de type annuleur d'interférence, dont les coefficients étaient mis à jour de manière adaptative grâce à l'algorithme *Least Mean Square* (*LMS*). Une avancée notable a ensuite été réalisée sous l'impulsion des travaux de Wang *et al* [11.34], repris ensuite par Reynolds *et al* [11.35] puis par Tüchler *et al* [11.36, 11.37]. Ces contributions ont permis d'établir une expression théorique pour les coefficients de l'égaliseur, prenant en compte explicitement la présence d'une information *a priori* sur les données émises. Cette avancée se révèle particulièrement intéressante pour les transmissions en mode paquet, où les coefficients de l'égaliseur sont précalculés une fois pour toute à partir d'une estimation de la réponse impulsionnelle du canal, puis appliqués ensuite à l'ensemble du bloc reçu.

La turbo-égalisation MEQM repose sur un schéma d'égalisation linéaire à entrées et sorties pondérées, optimisé suivant le critère MEQM. Ce type d'égaliseur est également parfois désigné sous le nom « d'égaliseur linéaire MEQM avec information *a priori* » dans la littérature. Cette section décrit le principe de cet égaliseur, en supposant que l'on connaît les paramètres du canal, ce qui permet de calculer directement les coefficients des filtres, ainsi que la mise en œuvre sous forme adaptative. Nous présentons ensuite quelques exemples de performance du turbo-égaliseur MEQM, puis nous décrivons l'implémentation sur processeur de signal (*Digital Signal Processor*, ou *DSP*) de cette solution. Cette partie se termine par une réflexion sur les applications potentielles de la turbo-égalisation MEQM.

Principe de l'égalisation linéaire MEQM à entrée et sortie pondérées

D'une manière générale, l'égaliseur linéaire MEQM à entrée et sortie pondérées peut se décomposer formellement en trois fonctions principales (figure 11.14).

1. La première opération, le *mapping SISO*, calcule une estimation souple (pondérée) sur les symboles émis, notée $\bar{\mathbf{x}} = (\bar{x}_0, \ldots, \bar{x}_{N-1})$, à partir de l'information *a priori* $\mathbf{L}_a(\mathbf{x})$ en provenance du décodeur à l'itération précédente.
2. L'égaliseur linéaire utilise ensuite les données estimées $\bar{x}_i$ pour reconstruire puis annuler l'IES au niveau du signal reçu, et le signal résultant est filtré afin d'éliminer l'interférence résiduelle. Les coefficients du filtre sont optimisés de manière à minimiser l'erreur quadratique moyenne entre la donnée égalisée et la donnée émise correspondante. Toutefois, et à la différence de l'égaliseur linéaire MEQM classique, les informations de fiabilité en provenance du décodeur sont ici prises en compte explicitement dans le calcul des coefficients.

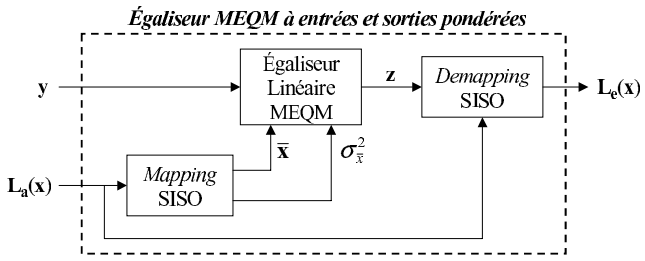


Figure 11.14 – Schéma de principe d'un égaliseur linéaire à entrées et sorties pondérées, optimisé suivant le critère MEQM.

3. L'égaliseur est finalement suivi d'un module de *demapping* à entrée et sortie pondérées dont le rôle consiste à convertir les symboles égalisés en LRV extrinsèques sur les bits codés et entrelacés.

Examinons maintenant plus en détails la mise en œuvre de chacune de ces trois fonctions.

•*Mapping SISO*

Cette opération consiste à calculer la donnée estimée $\bar{x}_i$, définie comme l'espérance mathématique de la donnée x_i émise à l'instant i :

$$\bar{x}_i = E_a\{x_i\} = \sum_{l=1}^{M} X_l \times P_a(X_l) \tag{11.24}$$

La somme porte ici sur l'ensemble des symboles discrets de la constellation. Le terme $P_a(X_l)$ désigne la probabilité *a priori* $\Pr(x_i = X_l)$ d'avoir émis le symbole X_l à l'instant i. Nous avons fait apparaître l'indice a au niveau du terme d'espérance pour souligner le fait que ces probabilités sont déduites de l'information *a prio*ri à l'entrée de l'égaliseur. En effet, sous réserve que les m bits composant le symbole x_i soient statistiquement indépendants, il est possible d'écrire :

$$P_a(X_l) = \prod_{j=1}^{m} P_a(X_{l,j}) \tag{11.25}$$

où l'élément binaire $X_{l,j}$ prend la valeur 0 ou 1 en fonction du symbole X_l considéré et de la règle de *mapping*. D'autre part, partant de la définition générale (11.4) du LRV, on peut montrer que probabilité *a priori* et LRV *a priori* sont liés par la relation suivante :

$$P_a(X_{l,j}) = \frac{1}{2}\left(1 + (2X_{l,j} - 1)\tanh\left(\frac{L_a(x_{i,j})}{2}\right)\right) \text{ avec } X_{l,j} \in \{0,1\} \tag{11.26}$$

Dans le cas particulier d'une modulation MDP-2, les calculs précédents se simplifient notablement. On obtient alors l'expression suivante pour le symbole

estimé $\bar{x}_i$:

$$\bar{x}_i = \tanh\left(\frac{L_a(x_i)}{2}\right) \tag{11.27}$$

Dans la situation classique où l'on fait l'hypothèse d'équiprobabilité sur les symboles émis, nous avons $L_a(x_{i,j}) = 0$ et $\bar{x}_i$ tend vers 0. En revanche, dans le cas idéal d'une information *a priori* parfaite, $L_a(x_{i,j}) \to \pm\infty$ et la donnée estimée $\bar{x}_i$ est alors strictement égale à la donnée émise. En résumé, la valeur de la donnée estimée $\bar{x}_i$ évolue en fonction de la qualité de l'information *a priori* délivrée par le décodeur. Ceci justifie l'appellation d'estimation « pondérée » (ou probabiliste) pour $\bar{x}_i$. Par construction, les données estimées $\bar{x}_i$ sont des variables aléatoires. On peut notamment montrer (voir [11.38] par exemple) qu'elles vérifient les propriétés statistiques suivantes :

$$E\{\bar{x}_i\} = 0 \tag{11.28}$$

$$E\left\{\bar{x}_i x_j^*\right\} = E\left\{\bar{x}_i \bar{x}_j^*\right\} = \sigma_{\bar{x}}^2 \delta_{i-j} \tag{11.29}$$

Le paramètre $\sigma_{\bar{x}}^2$ désigne ici la variance des données estimées $\bar{x}_i$. En pratique, cette quantité peut être estimée sur l'horizon d'une trame de N symboles comme suit :

$$\sigma_{\bar{x}}^2 = \frac{1}{N}\sum_{i=0}^{N-1} |\bar{x}_i|^2 \tag{11.30}$$

On vérifie aisément que sous l'hypothèse de symboles équiprobables *a priori*, $\sigma_{\bar{x}}^2 = 0$. À l'opposé, on obtient $\sigma_{\bar{x}}^2 = \sigma_x^2$ dans le cas d'une information *a priori* parfaite sur les symboles émis. En résumé, la variance des données estimées offre une mesure de la fiabilité des données estimées. Ce paramètre joue un rôle capital quant au comportement de l'égaliseur.

•*Calcul des coefficients de l'égaliseur linéaire*

Comme nous l'avons expliqué précédemment, l'étape d'égalisation peut être vue comme la mise en cascade d'une opération d'annulation d'interférence suivie d'une opération de filtrage. Les coefficients du filtre sont optimisés de manière à minimiser l'erreur quadratique moyenne $E\{|z_i - x_{i-\Delta}|^2\}$ entre la donnée égalisée z_i restituée à l'instant i et la donnée $x_{i-\Delta}$ émise à l'instant $i - \Delta$. L'introduction du retard de restitution Δ permet de prendre en compte l'anticausalité de la solution. Nous allons faire appel ici à un formalisme matriciel pour dériver la forme optimale des coefficients de l'égaliseur. En effet, les filtres numériques comportent toujours un nombre fini de coefficients en pratique. Le formalisme matriciel prend précisément en compte cet aspect et nous permet ainsi d'établir les coefficients optimaux sous la contrainte d'une réalisation de longueur finie.

On considère ici un filtre à F coefficients : $\mathbf{f} = (\mathbf{f}_0, \ldots, \mathbf{f}_{F-1})$. La réponse impulsionnelle du canal ainsi que la variance du bruit sont supposées connues, ce qui nécessite une estimation préalable de ces paramètres en pratique. Partant de l'expression (11.3) et en regroupant les F derniers échantillons observés à

la sortie du canal jusqu'à l'instant i sous forme d'un vecteur colonne $\mathbf{y}_i$, nous pouvons écrire :

$$\mathbf{y}_i = \mathbf{H}\mathbf{x}_i + \mathbf{w}_i \tag{11.31}$$

avec $\mathbf{y}_i = (y_i, \ldots, y_{i-F+1})^{\mathbf{T}}$, $\mathbf{x}_i = (x_i, \ldots, x_{i-F-L+2})^{\mathbf{T}}$ et $\mathbf{w}_i = (w_i, \ldots, w_{i-F+1})^{\mathbf{T}}$. La matrice $\mathbf{H}$, de dimensions $F \times (F + L - 1)$, est une matrice de Toeplitz[5] décrivant la convolution par le canal :

$$\mathbf{H} = \begin{pmatrix} h_0 & \cdots & h_{L-1} & 0 & \cdots & 0 \\ 0 & h_0 & & h_{L-1} & & \vdots \\ \vdots & & \ddots & & \ddots & 0 \\ 0 & \cdots & 0 & h_0 & \cdots & h_{L-1} \end{pmatrix} \tag{11.32}$$

Avec ces notations, l'étape d'annulation d'interférence à partir du signal estimé $\bar{\mathbf{x}}$ peut alors s'écrire formellement :

$$\tilde{\mathbf{y}}_i = \mathbf{y}_i - \mathbf{H}\tilde{\mathbf{x}}_i \tag{11.33}$$

où l'on a introduit le vecteur $\tilde{\mathbf{x}}_i = (\bar{x}_i, \ldots, \bar{x}_{i-\Delta+1}, 0, \bar{x}_{i-\Delta-1}, \ldots, \bar{x}_{i-F-L+2})^{\mathbf{T}}$, de dimension $F + L - 1$. La composante relative à la donnée estimée $x_{i-\Delta}$ est mise à zéro afin d'annuler uniquement l'IES et non le signal utile. En sortie du filtre transverse, l'expression de l'échantillon égalisé à l'instant i est donnée par :

$$z_i = \mathbf{f}^{\mathbf{T}}\tilde{\mathbf{y}}_i = \mathbf{f}^{\mathbf{T}}\left[\mathbf{y}_i - \mathbf{H}\tilde{\mathbf{x}}_i\right] \tag{11.34}$$

Il reste à déterminer l'expression théorique des coefficients du filtre $\mathbf{f}$ minimisant l'erreur quadratique moyenne $E\{|z_i - x_{i-\Delta}|^2\}$. Dans le cas le plus général, ces coefficients varient dans le temps. La solution correspondante, développée en détails par Tüchler *et al* [11.36,11.37], conduit à un égaliseur dont les coefficients doivent être recalculés à chaque nouveau symbole traité. Cet égaliseur représente ce que l'on peut faire de mieux à l'heure actuelle en matière d'égalisation MEQM en présence d'information *a priori*. En contrepartie, la charge de calcul associée à l'actualisation des coefficients symbole par symbole augmente de manière quadratique avec le nombre F de coefficients, ce qui s'avère encore trop complexe pour une implémentation temps réel. L'égaliseur que nous présentons ici peut être vu comme une version simplifiée, et donc sous-optimale, de la solution précédemment citée. Les coefficients du filtre $\mathbf{f}$ ne sont calculés qu'une seule fois par trame (à chaque itération) puis appliqués ensuite à l'ensemble du bloc, ce qui diminue considérablement le coût de mise en œuvre. D'autre part et en dépit de cette réduction de complexité, cet égaliseur conserve des performances proches de l'optimal[6], ce qui en fait un

[5] Les coefficients de la matrice sont constants le long de chacune des diagonales.

[6] La dégradation mesurée expérimentalement en comparaison avec la solution optimale variant dans le temps est généralement de l'ordre de 1 dB au maximum suivant le modèle de canal considéré.

excellent candidat pour une réalisation pratique. Cette solution a été dérivée indépendamment par plusieurs auteurs, dont [11.36] et [11.38].

Avec ces hypothèses, la forme optimale du jeu de coefficients $\mathbf{f}$ s'obtient à l'aide du théorème de projection, qui stipule que l'erreur d'estimation doit être orthogonale aux observations[7] :

$$E\left\{(z_i - x_{i-\Delta})\tilde{\mathbf{y}}_i^{\mathbf{H}}\right\} = \mathbf{0} \tag{11.35}$$

On obtient alors la solution suivante :

$$\mathbf{f}^* = E\{\tilde{\mathbf{y}}_i\tilde{\mathbf{y}}_i^{\mathbf{H}}\}^{-1}E\{x_{i-\Delta}^*\tilde{\mathbf{y}}_i\} \tag{11.36}$$

En utilisant les propriétés statistiques des données estimées $\bar{x}_i$, on remarque que :

$$E\{x_{i-\Delta}^*\tilde{\mathbf{y}}_i\} = E\{x_{i-\Delta}^*\mathbf{H}(\mathbf{x}_i - \tilde{\mathbf{x}}_i)\} = \mathbf{H}\mathbf{e}_{\boldsymbol{\Delta}}\sigma_x^2 \tag{11.37}$$

où nous avons introduit le vecteur unitaire $\mathbf{e}_\Delta$ de dimension $F+L-1$ qui vaut 1 en position Δ et 0 partout ailleurs. En désignant par $\mathbf{h}_\Delta$ la Δ-ième colonne Δ de la matrice $\mathbf{H}$, l'expression précédente s'écrit encore :

$$E\left\{x_{i-\Delta}^*\tilde{\mathbf{y}}_i\right\} = \mathbf{h}_{\boldsymbol{\Delta}}\sigma_x^2 \tag{11.38}$$

D'autre part,

$$\begin{aligned} E\left\{\tilde{\mathbf{y}}_i\tilde{\mathbf{y}}_i^{\mathbf{H}}\right\} &= \mathbf{H}E\left\{(\mathbf{x}_i - \tilde{\mathbf{x}}_i)(\mathbf{x}_i - \tilde{\mathbf{x}}_i)^{\mathbf{H}}\right\}\mathbf{H}^{\mathbf{H}} + \sigma_w^2\mathbf{I} \\ &= (\sigma_x^2 - \sigma_{\bar{x}}^2)\mathbf{H}\mathbf{H}^{\mathbf{H}} + \sigma_{\bar{x}}^2\mathbf{h}_{\boldsymbol{\Delta}}\mathbf{h}_{\boldsymbol{\Delta}}^{\mathbf{H}} + \sigma_w^2\mathbf{I} \end{aligned} \tag{11.39}$$

En résumé, la forme optimale des coefficients de l'égaliseur s'écrit finalement :

$$\mathbf{f}^* = \left[(\sigma_x^2 - \sigma_{\bar{x}}^2)\mathbf{H}\mathbf{H}^{\mathbf{H}} + \sigma_{\bar{x}}^2\mathbf{h}_{\boldsymbol{\Delta}}\mathbf{h}_{\boldsymbol{\Delta}}^{\mathbf{H}} + \sigma_w^2\mathbf{I}\right]^{-1}\mathbf{h}_{\boldsymbol{\Delta}}\sigma_x^2 \tag{11.40}$$

Par un jeu de calcul faisant intervenir une forme simplifiée du lemme d'inversion matricielle[8], la solution précédente peut encore s'écrire :

$$\mathbf{f}^* = \frac{\sigma_x^2}{1+\beta\sigma_{\bar{x}}^2}\,\tilde{\mathbf{f}}^* \tag{11.41}$$

où l'on a introduit le vecteur $\tilde{\mathbf{f}}$ et la quantité scalaire β définis comme suit :

$$\tilde{\mathbf{f}}^* = \left[(\sigma_x^2 - \sigma_{\bar{x}}^2)\mathbf{H}\mathbf{H}^{\mathbf{H}} + \sigma_w^2\mathbf{I}\right]^{-1}\mathbf{h}_{\boldsymbol{\Delta}} \quad \text{et} \quad \beta = \tilde{\mathbf{f}}^{\mathbf{T}}\mathbf{h}_{\boldsymbol{\Delta}} \tag{11.42}$$

Par l'intermédiaire de cette nouvelle écriture, on remarque que le calcul des coefficients de l'égaliseur repose principalement sur l'inversion de la matrice $(\sigma_x^2 - \sigma_{\bar{x}}^2)\mathbf{H}\mathbf{H}^{\mathbf{H}} + \sigma_w^2\mathbf{I}$, de dimensions $F \times F$. Or cette matrice possède

[7] Rappelons ici que la notation $\mathbf{A}^{\mathbf{H}}$ désigne la transposée *hermitienne* (transposée conjugée) de la matrice A.

[8] $\left[\mathbf{A} + \mathbf{u}\mathbf{u}^{\mathbf{H}}\right]^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{u}\mathbf{u}^{\mathbf{H}}\mathbf{A}^{-1}}{1+\mathbf{u}^{\mathbf{H}}\mathbf{A}^{-1}\mathbf{u}}$.

une structure riche puisqu'il s'agit d'une matrice de Toeplitz à symétrie hermitienne. Par conséquent, l'opération d'inversion peut être réalisée efficacement à l'aide d'algorithmes spécialement optimisés, avec un coût de calcul en $O(F^2)$ (voir au chapitre 4 de [11.39] par exemple). Afin de réduire encore davantage la complexité de la détermination des coefficients, les auteurs de [11.38] ont proposé une méthode sous-optimale mais néanmoins performante, utilisant la transformée de Fourier rapide (*Fast Fourier Transform*, ou *FFT*), présentant un coût en $O(F\log_2(F))$. En contrepartie, le nombre de coefficients F doit être une puissance de 2.

Il est particulièrement instructif d'étudier la forme limite prise par l'égaliseur dans le cas classique où les symboles émis sont supposés équiprobables (ce qui correspond à la 1ère itération du turbo-égaliseur). Dans ce cas, $\sigma_{\bar{x}}^2 = 0$ et les coefficients de l'égaliseur s'écrivent :

$$\mathbf{f}^* = \left[\sigma_x^2 \mathbf{H}\mathbf{H}^{\mathbf{H}} + \sigma_w^2 \mathbf{I}\right]^{-1} \mathbf{h}_{\mathbf{\Delta}} \sigma_x^2 \tag{11.43}$$

On reconnaît ici la forme d'un égaliseur linéaire MEQM classique, de longueur finie. À l'inverse, sous l'hypothèse d'une information *a priori* parfaite sur les symboles transmis, on a $\sigma_{\bar{x}}^2 = \sigma_x^2$. L'égaliseur prend alors la forme suivante :

$$\mathbf{f} = \frac{\sigma_x^2}{\sigma_x^2 \|\mathbf{h}\|^2 + \sigma_w^2} \mathbf{h}_{\mathbf{\Delta}}^* \quad \text{avec} \quad \|\mathbf{h}\|^2 = \mathbf{h}_{\mathbf{\Delta}}^{\mathbf{H}} \mathbf{h}_{\mathbf{\Delta}} = \sum_{k=0}^{L-1} |h_k|^2 \tag{11.44}$$

et le signal égalisé z_i s'écrit :

$$z_i = \frac{\sigma_x^2 \|\mathbf{h}\|^2}{\sigma_x^2 \|\mathbf{h}\|^2 + \sigma_w^2} \left(x_{i-\Delta} + \mathbf{h}_{\mathbf{\Delta}}^{\mathbf{H}} \mathbf{w}_i\right) \tag{11.45}$$

C'est l'expression de la sortie d'un annuleur d'interférence MEQM classique, alimenté par une estimation parfaite des données émises. Le signal égalisé se décompose comme la somme du signal utile $x_{i-\Delta}$, affecté d'un coefficient de biais caractéristique du critère MEQM, et d'un terme de bruit coloré. Autrement dit, l'égaliseur supprime toute l'IES sans remonter le niveau de bruit et atteint ainsi la borne théorique du filtre adapté.

En résumé, on constate que l'égaliseur linéaire adapte la stratégie d'égalisation en fonction de la fiabilité des données estimées, mesurée ici par le paramètre $\sigma_{\bar{x}}^2$.

Signalons pour conclure la description de l'égaliseur que l'opération d'annulation d'interférence définie formellement par l'équation (11.33) n'a pas de réalité physique au sens où elle ne peut être réalisée directement de cette manière à l'aide de filtres linéaires transverses. En pratique, on utilisera plutôt l'une des deux architectures présentées dans la figure 11.15, strictement équivalentes du point de vue théorique. La réalisation (1) fait intervenir le coefficient central $g_\Delta = \mathbf{f}^T \mathbf{h}_\Delta$ de la mise en cascade du canal avec le filtre $\mathbf{f}$. Dans le cas de la réalisation (2), on retrouve la structure classique d'un égaliseur de type

annuleur d'interférence opérant ici sur le signal estimé $\bar{\mathbf{x}}$. Le filtre $\mathbf{g} = \mathbf{f}^T\mathbf{H}$ est donné par la convolution du filtre $\mathbf{f}$ avec la réponse impulsionnelle du canal, le coefficient central g_Δ étant alors forcé à zéro.

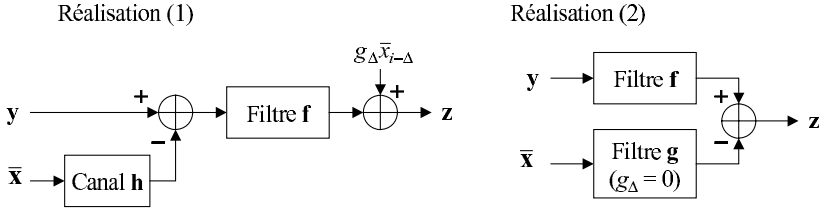


Figure 11.15 – Réalisation pratique de l'égaliseur à l'aide de filtres transverses.

•*Demapping SISO*

Le rôle de ce module consiste à convertir les données égalisées z_i en LRV extrinsèques sur les bits codés et entrelacés, qui seront ensuite transmis au décodeur de canal. D'une manière générale, on peut toujours décomposer l'expression de z_i comme la somme de deux quantités :

$$z_i = g_\Delta x_{i-\Delta} + \nu_i \tag{11.46}$$

Le terme $g_\Delta x_{i-\Delta}$ représente le signal utile, affecté d'un biais d'estimation g_Δ. Rappelons que ce biais correspond tout simplement au coefficient central de la mise en cascade du canal avec l'égaliseur. Quant au terme ν_i, il tient compte à la fois de l'interférence et du bruit résiduels en sortie de l'égaliseur. Afin de réaliser l'opération de *demapping*, on fait l'hypothèse que le terme d'interférence ν_i suit une distribution gaussienne[9], complexe, de moyenne nulle et de variance totale σ_ν^2. Les paramètres g_Δ et σ_ν^2 se déduisent facilement de la connaissance du jeu de coefficients de l'égaliseur. On peut ainsi montrer ([11.36], [11.38]) que l'on a :

$$g_\Delta = \mathbf{f^T h_\Delta} \text{ et } \sigma_\nu^2 = E\left\{|z_i - g_\Delta x_{i-\Delta}|^2\right\} = \sigma_x^2 g_\Delta(1 - g_\Delta) \tag{11.47}$$

Partant de ces hypothèses, le module de *demapping* calcule le LRV *a posteriori* sur les bits codés et entrelacés, noté $L(x_{i,j})$ et défini comme suit :

$$L(x_{i,j}) = \ln\left(\frac{\Pr(x_{i,j} = 1\,|z_i)}{\Pr(x_{i,j} = 0\,|z_i)}\right) \tag{11.48}$$

Les quantités présentes au numérateur et dénominateur peuvent être évaluées en sommant la probabilité *a posteriori* $\Pr(x_i = X_l|z_i)$ d'avoir émis un symbole particulier X_l de la constellation sur l'ensemble des symboles pour lesquels le

[9] En toute rigueur, cette hypothèse n'est strictement valable qu'à condition que l'égaliseur supprime toute l'IES, ce qui suppose une connaissance parfaite des données émises. Néanmoins, elle constitue une bonne approximation en pratique, particulièrement dans un contexte de turbo-égalisation où la fiabilité des décisions en sortie du décodeur augmente au fur et à mesure des itérations, ce qui améliore l'opération d'égalisation en retour.

j-ième bit composant ce symbole prend respectivement la valeur 0 ou 1. Ainsi, on peut écrire :

$$\begin{aligned} L(x_{i,j}) &= \ln\left(\frac{\sum\limits_{X_l/X_{l,j}=1}\Pr(x_i=X_l|z_i)}{\sum\limits_{X_l/X_{l,j}=0}\Pr(x_i=X_l|z_i)}\right) \\ &= \ln\left(\frac{\sum\limits_{X_l/X_{l,j}=1}P(z_i|x_i=X_l)P_a(X_l)}{\sum\limits_{X_l/X_{l,j}=0}P(z_n|x_i=X_l)P_a(X_l)}\right) \end{aligned} \tag{11.49}$$

La seconde égalité résulte de l'application de la relation de Bayes. Elle fait apparaître la probabilité *a priori* $P_a(X_l) = \Pr(x_i = X_l)$ d'avoir émis un symbole donné X_l de l'alphabet de modulation. Cette probabilité est calculée à partir de l'information *a priori* disponible à l'entrée de l'égaliseur (relations (11.25) et (11.26)). D'autre part, en exploitant les hypothèses précédentes, la vraisemblance de l'observation z_i conditionnellement à l'hypothèse d'avoir émis le symbole X_l à l'instant i s'écrit :

$$P(z_i\,|x_i\,=X_l) = \frac{1}{\pi\sigma_\nu^2}\exp\left(-\frac{|z_i - g_\Delta X_l|^2}{\sigma_\nu^2}\right) \tag{11.50}$$

Après simplification, le LRV *a posteriori* calculé par l'opération de *demapping* devient :

$$L(x_{i,j}) = \ln\left(\frac{\sum\limits_{X_l/X_{l,j}=1}\exp\left(-\frac{|z_i-g_\Delta X_l|^2}{\sigma_\nu^2}+\sum\limits_{k=1}^{m}X_{l,k}L_a(x_{i,k})\right)}{\sum\limits_{X_l/X_{l,j}=0}\exp\left(-\frac{|z_i-g_\Delta X_l|^2}{\sigma_\nu^2}+\sum\limits_{k=1}^{m}X_{l,k}L_a(x_{i,k})\right)}\right) \tag{11.51}$$

Comme dans le cas de l'égaliseur BCJR-*MAP*, on peut factoriser au numérateur et au dénominateur le terme d'information *a priori* relatif au bit considéré, afin de faire apparaître l'information extrinsèque à communiquer ensuite au décodeur :

$$L(x_{i,j}) = L_a(x_{i,j}) + \underbrace{\ln\left(\frac{\sum\limits_{X_l/X_{l,j}=1}\exp\left(-\frac{|z_i-g_\Delta X_l|^2}{\sigma_\nu^2}+\sum\limits_{k\neq j}X_{l,k}L_a(x_{i,k})\right)}{\sum\limits_{X_j/X_{j,i}=0}\exp\left(-\frac{|z_i-g_\Delta X_l|^2}{\sigma_\nu^2}+\sum\limits_{k\neq j}X_{l,k}L_a(x_{i,k})\right)}\right)}_{L_e(x_{i,j})} \tag{11.52}$$

Au final, l'information extrinsèque s'obtient donc tout simplement en soustrayant l'information *a priori* du LRV *a posteriori* calculé par l'égaliseur :

$$L_e(x_{i,j}) = L(x_{i,j}) - L_a(x_{i,j}) \tag{11.53}$$

Dans le cas particulier d'une modulation MDP-2, les équations de *demapping SISO* se simplifient pour donner l'expression suivante du LRV extrinsèque :

$$L(x_i) = \frac{4}{1 - g_\Delta} Re\{z_i\} \tag{11.54}$$

Lorsqu'on utilise une conversion M-aire binaire reposant sur un *mapping* de Gray, l'expérience montre que l'on peut réduire la complexité du *demapping* en ignorant l'information *a priori* en provenance du décodeur dans les équations précédentes[10], sans que cela affecte véritablement les performances du dispositif. En revanche, cette simplification ne s'applique plus lorsque l'on considère des règles de *mapping* plus évoluées, à l'exemple du principe de *Set Partitioning* utilisé dans la conception des modulations codées. Ce point a été particulièrement bien mis en évidence dans [11.40] et [11.41].

Ceci complète la description de l'égaliseur linéaire MEQM à entrées et sorties pondérées. Au final, on peut remarquer que contrairement à l'égaliseur BCJR-*MAP*, la complexité des opérations de *mapping* et *demapping SISO* augmente linéairement (et non exponentiellement) en fonction de la taille M de la constellation et du nombre L de coefficients intervenant dans le modèle discret équivalent du canal.

Réalisation adaptative de l'égaliseur

Le premier turbo-égaliseur MEQM a été historiquement proposé en 1997, directement sous forme adaptative [11.31][11.32]. Dans cette contribution, les relations permettant d'obtenir directement les filtres de l'égaliseur MEQM (11.40) à partir de la connaissance du canal n'étaient alors pas connues. La solution retenue consistait alors à faire converger de manière adaptative les coefficients des filtres de l'égaliseur en utilisant, par exemple, un algorithme de type gradient stochastique visant à minimiser l'erreur quadratique moyenne. Comme on le verra par la suite lors de l'évaluation des performances, le turbo-égaliseur MEQM adaptatif reste une solution tout à fait intéressante pour les canaux invariants ou faiblement variants dans le temps. L'objectif de ce paragraphe est de montrer que l'égaliseur MEQM adaptatif et l'égaliseur MEQM proposé en (11.40) présentent des performances et des caractéristiques proches.

La structure de l'égaliseur considérée est celle de la figure 11.15 (réalisation (2)). L'obtention des coefficients des filtres de l'égaliseur est assurée par un algorithme adaptatif composé de deux phases distinctes : la phase d'apprentissage et le régime de poursuite. La phase d'apprentissage nécessite l'utilisation de séquences constituées de symboles connus du récepteur permettant à l'égaliseur de converger vers une solution proche de l'égaliseur recherché. Ensuite, l'algorithme est piloté par les décisions sur les symboles en sortie de l'égaliseur ou par les décisions sur les symboles estimés.

[10] Ceci revient tout simplement à supposer les symboles émis équiprobables, c'est-à-dire à poser $P_a(X_l) = 1/M$ quels que soient le symbole et l'itération considérés.

Les techniques adaptatives consistent à déterminer, pour chaque symbole entrant dans l'égaliseur, la sortie z_i à partir de la relation suivante :

$$z_i = \mathbf{f_i^T} \mathbf{y}_i - \mathbf{g_i^T} \tilde{\mathbf{x}}_i \tag{11.55}$$

où $\mathbf{y}_i = (y_{i+F}, \ldots, y_{i-F})^T$ et $\tilde{\mathbf{x}}_i = (\bar{x}_{i+G}, \ldots, \bar{x}_{i-\Delta+1}, 0, \bar{x}_{i-\Delta-1}, \ldots, \bar{x}_{i-G})^T$ sont les vecteurs des échantillons provenant du canal et des symboles estimés de longueur respective $2F+1$ et $2G+1$. Notons que la composante relative à la donnée estimée $\bar{x}_{i-\Delta}$ est mise à zéro afin de ne pas annuler le signal utile. Les vecteurs $\mathbf{f}_i = (f_{i,F}, \ldots, f_{i,-F})^T$ et $\mathbf{g}_i = (g_{i,G}, \ldots, g_{i,-G})^T$ représentent les coefficients des filtres $\mathbf{f}$ et $\mathbf{g}$ lesquels sont fonction du temps car ils évoluent à chaque nouveau symbole reçu de manière adaptative.

Les relations utilisées pour actualiser les vecteurs des coefficients peuvent être obtenues à partir d'un algorithme du gradient stochastique de type *LMS* :

$$\begin{array}{rcl} \mathbf{f}_{i+1} & = & \mathbf{f}_i - \mu\,(z_i - x_{i-\Delta})\,\mathbf{y}_i^* \\ \mathbf{g}_{i+1} & = & \mathbf{g}_i - \mu\,(z_i - x_{i-\Delta})\,\tilde{\mathbf{x}}_i^* \end{array} \tag{11.56}$$

où μ est un facteur réel positif de faible valeur.

Lors de la première itération du turbo-égaliseur, $\tilde{\mathbf{x}}_i$ est un vecteur dont toutes les composantes sont nulles ; il en résulte que le vecteur des coefficients $\mathbf{g}_i$ est lui aussi nul. L'égaliseur MEQM converge alors de manière adaptative vers un égaliseur transverse MEQM. Lorsque les données estimées sont très fiables et proches des données émises l'égaliseur MEQM converge vers un annuleur d'interférences idéal (génie), présentant alors les performances d'une transmission sans interférences entre symboles. Les formes limites de l'égaliseur adaptatif sont donc totalement identiques à celles obtenues en (11.43) et (11.44), à condition bien sûr de permettre à l'algorithme adaptatif de converger vers un minimum local proche de la solution optimale.

Notons cependant que pour les itérations intermédiaires où les symboles d'informations $\bar{x}_i$ estimés ne sont ni nuls ni parfaits, il ne faut pas alimenter le filtre $\mathbf{g}_i$ directement avec les symboles émis sinon l'égaliseur va converger vers la solution de l'annuleur d'interférence génie ce qui n'est pas le but recherché. Pour permettre à l'égaliseur de converger vers la solution visée, l'idée consiste ici à fournir au filtre $\mathbf{g}_i$ une information pondérée construite à partir des symboles émis, pendant les périodes d'apprentissage :

$$(\bar{x}_i)_{SA} = \sigma_{\bar{x}} x_i + \sqrt{1 - \sigma_{\bar{x}}^2}\,\eta_i \tag{11.57}$$

où $\sigma_{\bar{x}}^2$ correspond à la variance estimée à partir des symboles d'informations estimés $\bar{x}_i$ de l'itération précédente (11.24) et η_i un BABG de moyenne nulle et de variance unitaire.

En période de poursuite et afin de permettre à l'égaliseur de suivre des variations du canal, il est possible de remplacer dans les relations (11.56) les symboles émis x_i par les décisions en sortie de l'égaliseur $\hat{x}_i$ ou encore par les décisions sur les symboles estimés.

Lorsqu'on détermine l'égaliseur MEQM de manière adaptative, on ne connaît pas explicitement l'expression de la réponse impulsionnelle du canal et la relation d'actualisation de $\mathbf{g}_i$ ne permet pas d'obtenir g_Δ car la composante $g_{i,\Delta}$ est nulle. Pour réaliser l'opération de *demapping SISO*, on doit cependant estimer à la fois le biais g_Δ sur les données z_n délivrées par l'égaliseur et la variance σ_ν^2 de l'interférence résiduelle en sortie de l'égaliseur. Ces deux paramètres peuvent être estimés comme nous allons le voir à partir de la sortie de l'égaliseur. En reprenant la relation (11.46), on peut montrer le résultat général suivant :

$$E\left(|z_i|^2\right) = g_\Delta \sigma_x^2 \tag{11.58}$$

En supposant que la variance des données émises est normalisée à l'unité, l'estimation de g_Δ s'obtient par :

$$\hat{g}_\Delta = \frac{1}{N} \sum_{i=0}^{N-1} |z_i|^2 \tag{11.59}$$

Une fois que l'on a estimé g_Δ, on en déduit immédiatement la valeur de σ_ν^2 grâce à la relation (11.47) :

$$\sigma_\nu^2 = \sigma_x^2 \hat{g}_\Delta (1 - \hat{g}_\Delta) \tag{11.60}$$

Une particularité de la turbo-égalisation MEQM adaptative concerne la détermination des symboles estimés. En effet, afin d'améliorer les performances du turbo-égaliseur et conformément aux remarques de [11.31] et [11.42], on utilise dans (11.27) l'information *a posteriori* en sortie du décodeur de canal au lieu de l'information extrinsèque.

Nous avons donc défini un turbo-égaliseur MEQM adaptatif dont les coefficients des filtres de l'égaliseur avec information *a priori* sont obtenus à partir d'un algorithme du gradient stochastique de faible complexité permettant entre autres un suivi des variations temporelles lentes du milieu de transmission. Un inconvénient de cette technique réside dans la nécessité d'émettre des séquences d'apprentissage (SA), lesquelles diminuent l'efficacité spectrale. Toutefois l'utilisation d'algorithmes autodidactes ou aveugles peuvent permettre de diminuer voire de s'affranchir de séquences d'apprentissage. Ainsi l'égaliseur de la première itération de type transverse peut être remplacé par un égaliseur autodidacte appelé *SADFE* (*Self Adaptive Decision-Feedback Equalizer*) [11.32] ne nécessitant qu'une petite part de séquence d'apprentissage. Les travaux de Hélard *et al*[11.43] ont montré qu'un turbo-égaliseur ainsi constitué d'un *SADFE* en première itération permet d'atteindre des performances identiques au turbo-égaliseur MEQM adaptatif avec séquence d'apprentissage tout en obtenant une meilleure efficacité spectrale, grâce à la suppression de la séquence d'apprentissage. Pour cela, il est cependant nécessaire d'augmenter le nombre d'itérations.

Exemples de performances

À des fins de comparaison, les performances du turbo-égaliseur MEQM ont été simulées en considérant le même scénario de transmission que pour le turbo-égaliseur *MAP*.

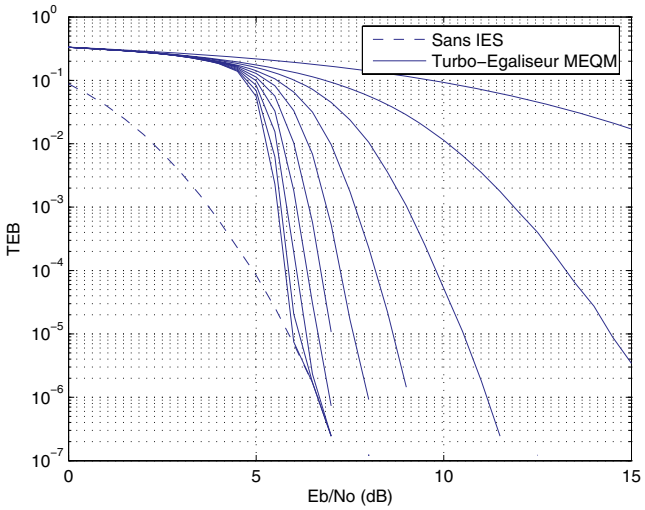


Figure 11.16 – Performances du turbo-égaliseur MEQM sur le canal Proakis C, avec un code convolutif non récursif non systématique de rendement $R = 1/2$, à 4 états, un entrelaceur pseudo-aléatoire de taille 16384 bits et une modulation MDP-2.

Dans un premier temps, les paramètres du canal ont été supposés parfaitement estimés. Les coefficients sont calculés une fois par trame, par inversion matricielle, en considérant un filtre numérique avec $F = 15$ coefficients et un retard de restitution $\Delta = 9$. Les résultats de simulation, obtenus après 10 itérations, sont présentés sur la figure 11.16. Le déclenchement du processus itératif s'effectue ici à partir d'un rapport signal à bruit seuil d'environ 4 dB, et le turbo-égaliseur supprime l'intégralité de l'IES après 10 itérations au-delà d'un rapport signal à bruit de 6 dB. Comparativement aux résultats obtenus avec le turbo-égaliseur *MAP* (figure 11.13), on retiendra donc les remarques suivantes, conséquences naturelles de la sous-optimalité du critère employé pour réaliser l'égaliseur :

1. La convergence est plus tardive en turbo-égalisation MEQM (de l'ordre de 1 dB ici par rapport au *MAP*).
2. Le turbo-égaliseur MEQM nécessite plus d'itérations que le turbo-égaliseur *MAP* pour atteindre des taux d'erreurs comparables.

Cependant le turbo-égaliseur MEQM démontre ici sa capacité à supprimer toute l'IES lorsque le rapport signal à bruit est suffisamment élevé, et ce même

sur un canal pourtant réputé difficile à égaliser. Il constitue donc une solution alternative sérieuse au turbo-égaliseur MAP lorsque ce dernier ne peut être utilisé pour raisons de complexité.

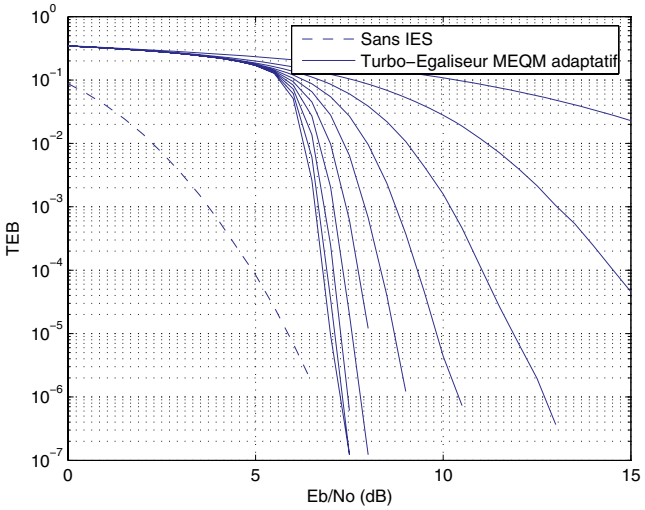


Figure 11.17 – Performances du turbo-égaliseur MEQM en régime adaptatif, sur le canal Proakis C, avec un code convolutif non récursif non systématique de rendement $R = 1/2$, à 4 états, un entrelaceur pseudo-aléatoire de taille 16384 bits et une modulation MDP-2.

Dans un deuxième temps, l'hypothèse de la connaissance parfaite des paramètres du canal a été levée et le turbo-égaliseur a été cette fois-ci simulé sous forme adaptative, en conservant les mêmes paramètres de transmission. La communication débute par l'émission d'une séquence d'apprentissage initiale de 16384 symboles supposée parfaitement connue du récepteur, puis des séquences d'apprentissage périodiques de 1000 symboles entrecoupées de 16384 symboles d'informations, périodes durant lesquelles le turbo-égaliseur est entièrement piloté par décision. Les filtres de l'égaliseur comportent chacun 21 coefficients avec $F = G = 10$. Les coefficients sont actualisés à l'aide de l'algorithme LMS, avec un pas d'adaptation $\mu = 0,0005$ durant les phases d'apprentissage, puis à $0,000005$ en régime de poursuite. Les résultats de simulation sont donnés dans la figure 11.17, en considérant 10 itérations en réception. On observe une dégradation de l'ordre de 1 dB seulement par rapport à la situation idéale où le canal est supposé parfaitement connu. On notera que, lorsque le canal est estimé et utilisé pour le calcul direct des coefficients de l'égaliseur MEQM, des pertes de performances apparaissent ce qui réduit la dégradation par rapport à la situation idéale de la figure 11.16. Pour établir la figure 11.17, nous n'avons pas pris en compte la perte engendrée par l'utilisation de séquences d'apprentissage sur le rapport signal à bruit.

Au vu de ces résultats, nous remarquons que la différence majeure entre la turbo-égalisation MEQM adaptative et celle qui utilise le calcul direct des coefficients à partir de l'estimé du canal réside dans la manière de déterminer les coefficients des filtres, puisque la structure et le critère d'optimisation des égaliseurs sont identiques.

Signalons pour terminer que, de la même façon que pour le turbo-égaliseur *MAP*, on peut utiliser les diagrammes *EXIT* afin de prédire le seuil de convergence théorique du turbo-égaliseur MEQM, sous l'hypothèse d'un entrelacement idéal. Le lecteur trouvera un complément d'information à ce sujet dans [11.15] ou [11.37] par exemple.

Exemple d'implémentation et applications

L'implémentation d'un turbo-égaliseur MEQM sur processeur de signal a été rapportée dans [11.44]. La cible retenue était le processeur TMS320VC5509 proposé par Texas Instruments. Ce *DSP* opérant en virgule fixe sur des données 16 bits se caractérise par sa faible consommation, ce qui en fait un candidat idéal pour les récepteurs radio-mobiles. Le schéma de transmission considéré comprenait un codeur convolutif de rendement 1/2 à 4 états et un entrelaceur de taille 1024 bits suivi d'une modulation MDP-4. L'ensemble du turbo-égaliseur a été implanté en langage C sur le *DSP*, à l'exception de certains traitements optimisés en assembleur (filtrage et *FFT*) offerts par une librairie spécialisée. L'égaliseur comportait 32 coefficients, de manière à pouvoir traiter de l'IES recouvrant jusqu'à 16 périodes symboles. Le décodage était réalisé suivant l'algorithme Max-Log-*MAP*. Les résultats de simulation ont montré que sous réserve de choisir judicieusement la représentation en virgule fixe des données manipulées (dans la limite des 16 bits maximum impartis par le *DSP*), l'étape de quantification n'occasionnait aucune perte de performance en comparaison avec le récepteur correspondant en virgule flottante. Le débit final obtenu était de l'ordre de 42 kbits/s après 5 itérations, ce qui démontre la faisabilité de tels récepteurs avec les moyens technologiques actuels. Le challenge consiste maintenant à définir des architectures circuit adéquates, capables d'opérer à plusieurs Mbits/s, afin de répondre aux exigences émergentes en matière de services hauts débits.

La turbo-égalisation MEQM constitue une technologie relativement récente. À ce titre, on dénombre encore peu d'études, au moment de la rédaction de cet ouvrage, quant aux applications potentielles de cette technique de réception. D'une manière générale, le recours à la turbo-égalisation MEQM prend tout son sens dans le cadre de transmissions haut débit sur des canaux fortement sélectifs en fréquence. Ce système a ainsi démontré d'excellentes performances sur le canal ionosphérique typiquement utilisé dans le cadre de communications militaires HF. En effet, les échos longs produit par ce canal empêchent toute utilisation de récepteur à base d'égaliseur *MAP*. D'autre part, les schémas d'égalisation linéaire conventionnelle ne permettent pas d'atteindre une qualité de transmission acceptable pour des modulations du type MDP-8, voire MAQ-

16. La turbo-égalisation MEQM est alors une solution attractive au problème de la croissance du débit des transmissions militaires. Dans le contexte des communications HF, l'intérêt de la turbo-égalisation MEQM pour des modulations à forte efficacité spectrale a ainsi été validée par les travaux de Langlais [11.45] et ceux de Otnes [11.46] qui montrent que cette technique offre des gains jusqu'à 5 dB comparée à un récepteur conventionnel. À notre connaissance, la turbo-égalisation MEQM n'a pas encore été implémentée dans des modems standardisés. Cependant, il est important de noter que cette technique de réception permet d'améliorer notablement les performances des transmissions tout en conservant les émetteurs standardisés.

11.2 La turbo-détection multi-utilisateurs et son application aux systèmes CDMA

11.2.1 Introduction et quelques notations

Dans un système Accès Multiple à Répartition par Code (AMRC) (*Code Division Multiple Access, CDMA* en anglais), tel que représenté en figure 11.18, l'utilisateur k ($1 \leqslant k \leqslant K$) transmet une suite d'éléments binaires $\{d_k\} = \pm 1$ avec une amplitude A_k. Pour chacun des utilisateurs, un codeur de canal (CC_k) est utilisé suivi d'un entrelaceur externe (π_k) avant l'opération d'étalement (multiplication par le code d'étalement $\mathbf{s}_k$, de taille N, normalisée) qui fournit des symboles binaires appelés *chips*. Ce code peut varier à chaque temps symbole.

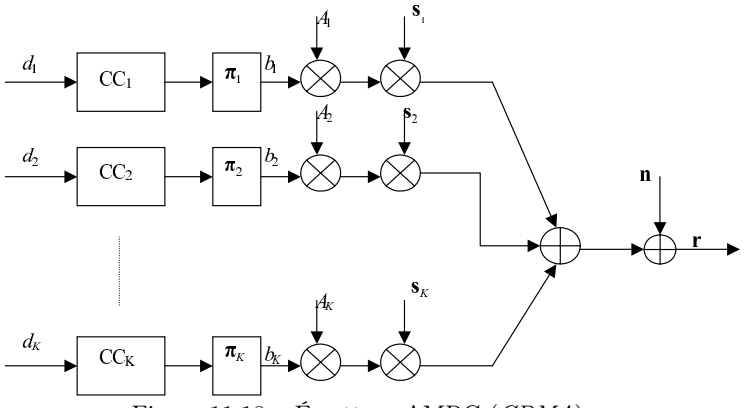


Figure 11.18 – Émetteur AMRC (*CDMA*).

Le signal reçu, $\mathbf{r}$, peut s'écrire sous forme matricielle par :

$$\mathbf{r} = \mathbf{SAb} + \mathbf{n} \tag{11.61}$$

où :

- **S** est la matrice $N \times K$ formée par les codes normalisés de chaque utilisateur (la k-ième colonne représente le k-ième code $\mathbf{s}_k$ dont la norme est égale à l'unité),
- **A** est une matrice $K \times K$ diagonale formée par les amplitudes A_k de chaque utilisateur.
- **b** est le vecteur de dimension K formé par les éléments transmis après codage canal par les K utilisateurs.
- **n** est le vecteur de dimension N Gaussien centré de matrice de covariance est $\sigma^2\mathbf{I}_N$.

Les débits source des différents utilisateurs peuvent être différents. La taille du code d'étalement est tel que le débit *chip* (après étalement) soit le même pour tous les utilisateurs. Le signal reçu **r** est donné par la contribution de l'ensemble des K utilisateurs plus un BABG centré de variance σ^2. Nous cherchons, à partir de l'observation **r**, à retrouver les bits d'information d_k de chaque utilisateur. La figure 11.19 donne le schéma du récepteur utilisant une technique de type turbo-*CDMA* pour traiter conjointement la multi-détection et le décodage de canal :

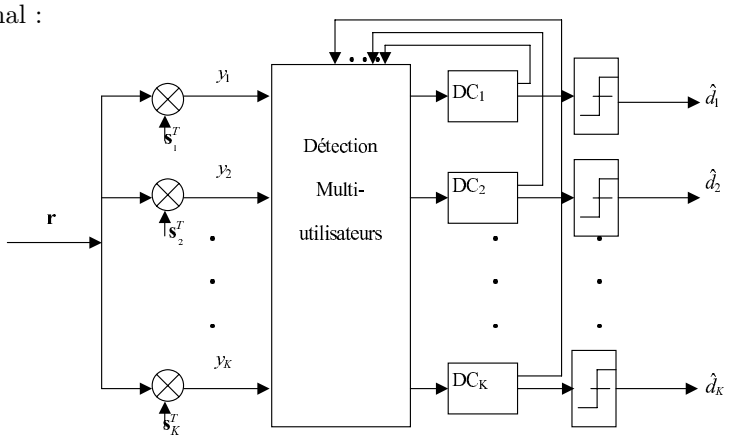


Figure 11.19 – Récepteur Turbo-*CDMA*

11.2.2 Détection multi-utilisateurs

Dans cette section, les principales méthodes de détection multi-utilisateurs sont présentées. Afin de simplifier la description de ces méthodes, seul le cas de transmissions synchrones à travers des canaux gaussiens est considéré.

Récepteur standard

Le détecteur le plus simple (détecteur conventionnel ou standard) est celui qui fonctionne comme si chaque utilisateur était seul sur le canal. Le récepteur est tout simplement formé du filtre adapté à la signature de l'uti-

lisateur concerné (cette opération est également appelée désétalement), voir figure 11.20.

À la sortie du banc de filtres adaptés, le signal peut s'écrire sous la forme :

$$\mathbf{y} = \mathbf{S}^T\mathbf{r} = \mathbf{RAb} + \mathbf{S}^T\mathbf{n} \tag{11.62}$$

Remarquons que le vecteur du bruit additif en sortie du banc de filtres adaptés, est formé de composantes corrélées. Sa matrice de covariance dépend directement de la matrice d'intercorrélation des séquences d'étalement utilisées, $\mathbf{R} = \mathbf{S}^T\mathbf{S}$. On a $\mathbf{S}^T\mathbf{n} \sim N(0, \sigma^2\mathbf{R})$.

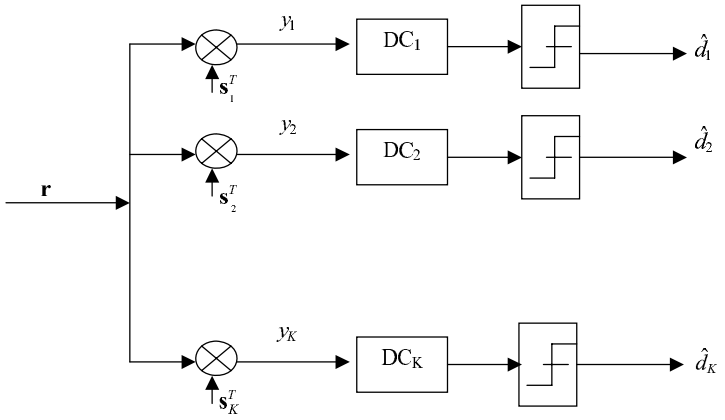


Figure 11.20 – Détecteur standard.

On peut montrer que la probabilité d'erreur (avant décodage de canal) pour le k-ième utilisateur peut s'écrire sous la forme :

$$P_{\mathrm{e},k} = P\left(\hat{b}_k \neq b_k\right) = \frac{1}{2^{K-1}} \sum_{\mathbf{b}_{-k}\in\{-1,+1\}^{K-1}} Q\left(\frac{A_k}{\sigma} + \sum_{j\neq k} b_j \frac{A_j}{\sigma}\rho_{jk}\right) \tag{11.63}$$

où $\rho_{jk} = \mathbf{s}_j^T\mathbf{s}_k$ mesure l'intercorrélation entre les codes des utilisateurs j et k, avec $\mathbf{b}_{-k} = (b_1, b_2, \cdots, b_{k-1}, b_{k+1}, \cdots, b_K)$.

En supposant que les codes d'étalement utilisés sont tels que les coefficients d'intercorrélation soient constants et égaux à $0,2$, la figure 11.21 donne les performances du récepteur standard, en terme de probabilité d'erreur du premier utilisateur en fonction du Rapport Signal à Bruit, pour un nombre d'utilisateurs variant de 1 à 6. Les messages de tous les utilisateurs sont supposés être reçus avec la même puissance. On remarque bien sûr que plus le nombre d'utilisateurs augmente plus les performances se dégradent. Cette probabilité d'erreur peut même tendre vers $1/2$, alors que le rapport signal à bruit augmente, si la condition suivante (effet d'éblouissement) n'est pas vérifiée : $A_k > \sum_{j\neq k} A_j\,|\rho_{jk}|$.

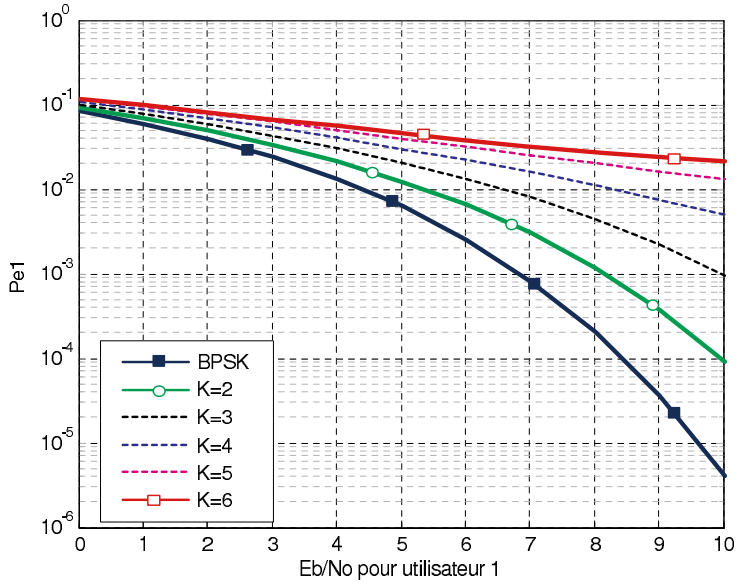


Figure 11.21 – Probabilité d'erreur du premier utilisateur en fonction du rapport signal à bruit $Eb/N0$ pour des coefficients d'intercorrélation constants $\rho = 0,2$, pour K=1 à 6 utilisateurs partageant la ressource.

Détection conjointe optimale

La détection conjointe optimale consiste à maximiser la probabilité *a posteriori* (probabilité du vecteur $\mathbf{b}$ conditionnellement à l'observation $\mathbf{y}$). Si on suppose que les éléments binaires transmis sont équiprobables et sachant que $\mathbf{y} = \mathbf{S}^T\mathbf{r} = \mathbf{RAb} + \mathbf{S}^T\mathbf{n}$ avec $\mathbf{S}^T\mathbf{n} \sim N(0, \sigma^2\mathbf{R})$, on peut déduire que la détection conjointe optimale se traduit par les équivalences :

$$\mathrm{Max}_{\mathbf{b}}\left(f_{\mathbf{y}}^{\mathbf{b}}(\mathbf{y})\right) \Leftrightarrow \mathrm{Min}_{\mathbf{b}}\left(\|\mathbf{y}-\mathbf{RAb}\|^2_{\mathbf{R}^{-1}}\right) \Leftrightarrow \mathrm{Max}_{\mathbf{b}}\left(2\mathbf{b}^T\mathbf{Ay}-\mathbf{b}^T\mathbf{ARAb}\right) \qquad (11.64)$$

La recherche exhaustive de la solution optimale est relativement complexe.

Détecteur décorrélateur

Le détecteur décorrélateur multiplie l'observation $\mathbf{y}$ par l'inverse de la matrice d'intercorrélation des codes : $\mathbf{R}^{-1}\mathbf{y} = \mathbf{Ab} + \mathbf{R}^{-1}\mathbf{S}^T\mathbf{n}$. Cette équation montre que le décorrélateur permet d'annuler complétement les interférences d'accès multiple, ce qui le rend robuste vis-à-vis des effets d'éblouissements (*Near Far Effect*). En revanche, le bruit additif gaussie résultant a une variance plus importante. On a en effet : $\mathbf{R}^{-1}\mathbf{S}^T\mathbf{n} \sim N(0, \sigma^2\mathbf{R}^{-1})$. La probabilité

d'erreur du k-ième utilisateur peut alors s'écrire sous la forme :

$$P_{\mathrm{e},k} = P\left(\hat{b}_k \neq b_k\right) = Q\left(\frac{A_k}{\sigma\sqrt{\left(\mathbf{R}^{-1}\right)_{kk}}}\right) \tag{11.65}$$

Détecteur MEQM linéaire

Le détecteur MEQM consiste à trouver la transformation $\mathbf{M}$ qui minimise l'Erreur Quadratique Moyenne : $\mathrm{Min}_{\mathbf{M}\in R^{K\times K}} E\left(\|\mathbf{b}-\mathbf{My}\|^2\right)$. Cette transformation n'est autre que :

$$\mathbf{M} = \mathbf{A}^{-1}\left(\mathbf{R}+\sigma^2\mathbf{A}^{-2}\right)^{-1} \tag{11.66}$$

Par conséquent, la probabilité d'erreur du k-ième utilisateur s'écrit sous la forme :

$$P_{\mathrm{e},k} = \frac{1}{2^{K-1}} \sum_{\mathbf{b}_{-k}\in\{-1,+1\}^{K-1}} Q\left(\frac{A_k}{\sigma}\frac{(\mathbf{MR}_{k,k})}{\sqrt{(\mathbf{MRM})_{k,k}}}\left(1+\sum_{j\neq k}\frac{(\mathbf{MR}_{k,j})A_j b_j}{(\mathbf{MR}_{k,k})A_k}\right)\right) \tag{11.67}$$

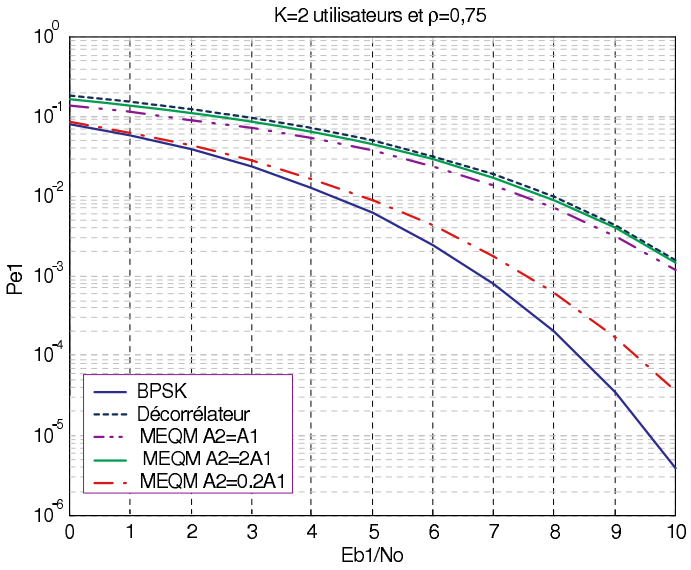


Figure 11.22 – Comparaison entre méthode du décorrélateur et MEQM.

Afin de comparer les deux techniques, détecteur décorrélateur et MEQM, les deux récepteurs ont été simulés avec deux utilisateurs dont les codes d'étalements sont fortement corrélés (coefficients d'intercorrélation égaux à 0,75). La figure 11.22 montre les courbes du TEB pour le premier utilisateur et paramétré

par la puissance du deuxième utilisateur (ou son amplitude). Les performances du récepteur MEQM sont toujours meilleures que celles du décorrélateur. Pour une faible puissance du deuxième utilisateur, les performances sont proches de la référence mono-utilisateur. Par contre pour une forte puissance du deuxième utilisateur, les performances MEQM vont tendre vers celles du décorrélateur.

Détecteur Itératif

Les récepteurs décorrélateur ou MEQM peuvent être implémentés avec des méthodes itératives d'inversion matricielle (Jacobi, Gauss-Siedel, ou relaxation). La méthode de Jacobi conduit à la méthode *PIC* (*Parallel Interference Cancellation*). Celle de Gauss-Siedel conduit à la méthode *SIC* (*Successive Interference Cancellation*). La figure 11.23 représente le schéma d'implémentation du *SIC* (avec $K = 4$ utilisateurs et $M = 3$ itérations) où $ICU_{m,k}$ représente l'unité d'annulation d'interférences (en anglais *interference cancellation unit* ou ICU) pour le k-ième utilisateur à l'itération m (voir figure 11.24). Les éléments binaires sont initialisés à zéro : à l'itération $m = 0$, $b_{0,k} = 0$ pour $k = 1, \ldots, K$.

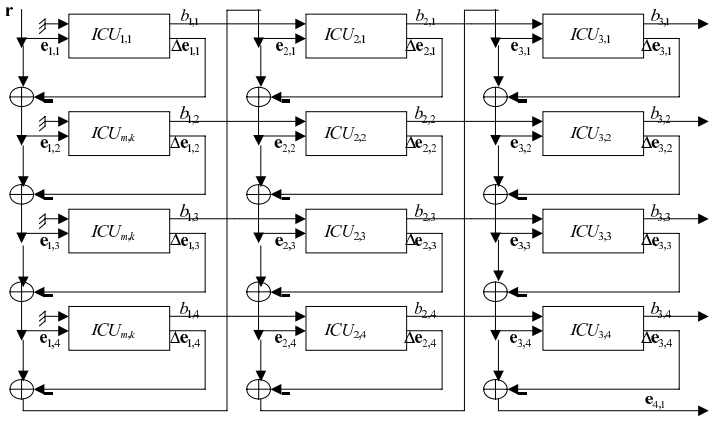


Figure 11.23 – Détecteur itératif *SIC* (*Successive Interference Cancellation*), avec $K = 4$ utilisateurs et $M = 3$ itérations.

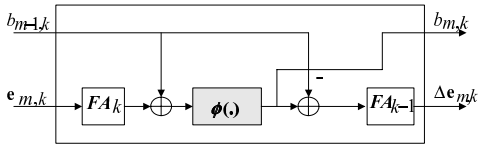


Figure 11.24 – Unité d'annulation d'interférence $ICU_{m,k}$ pour l'utilisateur k, à l'itération m.

la fonction FA_k (respectivement FA_k^{-1}) représente le désétalement (respectivement l'étalement) du k-ième utilisateur. La fonction $\phi(.)$ peut être choisie comme une fonction non linéaire ou tout simplement comme étant égale à la fonction identité (voir aussi le choix de $\phi(.)$ dans le cas du turbo-CDMA). Si l'on choisit la fonction identité, l'unité *ICU* de la figure 11.24 peut bien sûr être simplifiée et est aisée à définir. Dans ce dernier cas, on peut vérifier que, pour l'utilisateur k et à l'itération m, la sortie du récepteur peut s'écrire comme le résultat d'un filtrage linéaire :

$$b_{m,k} = \mathbf{s}_k^T \prod_{j=1}^{k-1} \left(\mathbf{I} - \mathbf{s}_j \mathbf{s}_j^T\right) \sum_{p=0}^{m-1} \mathbf{\Phi}_K^p \mathbf{r} = \mathbf{g}_{m,k}^T \mathbf{r} \tag{11.68}$$

avec :

$$\mathbf{\Phi}_K = \prod_{j=1}^{K} \left(I - \mathbf{s}_j \mathbf{s}_j^T\right) \tag{11.69}$$

On peut montrer que la probabilité d'erreur pour le k-ième utilisateur à l'itération m, peut s'écrire sous la forme suivante, où $\mathbf{S}$ est la matrice des codes et $\mathbf{A}$ est la matrice diagonale des amplitudes :

$$P_e(m,k) = \frac{1}{2^{K-1}} \sum_{\mathbf{b}/b_k=+1} Q\left(\frac{\mathbf{g}_{m,k}^T \mathbf{SAb}}{\sigma\sqrt{\mathbf{g}_{m,k}^T \mathbf{g}_{m,k}}}\right) \tag{11.70}$$

La figure 11.25 donne un exemple de simulations de la méthode *SIC* avec 5 utilisateurs, un facteur d'étalement de 20 et une matrice d'intercorrélation donnée par :

$$\mathbf{R} = \begin{pmatrix} 1 & 0,3 & 0 & 0 & 0 \\ 0,3 & 1 & 0,3 & 0,3 & 0,1 \\ 0 & 0,3 & 1 & 0 & -0,2 \\ 0 & 0,3 & 0 & 1 & 0 \\ 0 & 0,1 & -0,2 & 0 & 1 \end{pmatrix}$$

On constate qu'au bout de 3 itérations, le *SIC* converge vers le résultat obtenu avec le décorrélateur (on peut en effet prouver mathématiquement que le *SIC* converge vers ce résultat quand le nombre d'itérations M tend vers l'infini).

11.2.3 Turbo-CDMA

Plusieurs techniques de type turbo-CDMA ont été proposées pour traiter conjointement la multi-détection et le décodage de canal :

- Varanasi *et al* [11.50] ont proposé de décoder (estimation dure) et de recoder immédiatement chaque utilisateur avant de soustraire cette contribution au signal reçu. La même opération est effectuée sur le signal résiduel pour décoder les informations du deuxième utilisateur et ainsi de suite jusqu'au dernier utilisateur.

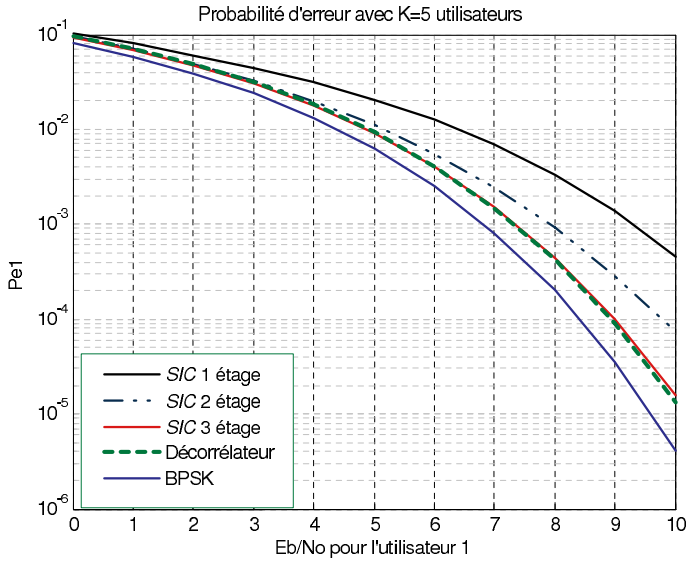


Figure 11.25 – Simulation d'un récepteur *SIC*.

- Reed *et al* [11.52] ont proposé d'utiliser un banc de filtres adaptés suivi (en parallèle) des différents décodeurs avant de soustraire pour chaque utilisateur l'interférence d'accès multiple liée aux $K-1$ autres utilisateurs.
- Wang *et al* [11.34] ont proposé un détecteur multi-utilisateur qui consiste en une implémentation en parallèle des filtres MEQM associés à chaque utilisateur suivis des décodeurs de canal correspondants. Ces deux éléments échangent itérativement leurs informations extrinsèques.
- Tarable *et al*[11.51] ont proposé une simplification de la méthode présentée dans [11.34]. Pour les premières itérations, un détecteur multi-utilisateur du type MEQM est utilisé, suivi des décodeurs de canal disposés en parallèle. Pour les dernières itérations, le filtre MEQM est remplacé par un banc de filtres adaptés.

Détecteur Turbo-*SIC*

Dans ce paragraphe, le décodage de canal est introduit dans une nouvelle structure d'annulation d'interférences successives (*SIC*). La figure 11.23 reste valable, seule les unités $ICU_{m,k}$ changent. Chaque unité d'annulation d'interférence $ICU_{m,k}$, relative au k-ième utilisateur et à l'itération m, est donnée par la figure 11.26. L'originalité réside dans la façon dont cette unité est conçue : le signal d'erreur résiduel $\mathbf{e}_{m,k}$ est désétalé (par $\mathbf{s}_k$) puis désentrelacé (π_k^{-1}) avant addition de l'estimation pondérée des données $b_{m-1,k}$ du même utilisateur calculées à l'itération précédente. Le signal ainsi obtenu, $y_{m,k}$, passe par le décodeur de canal qui fournit le logarithme du rapport de vraisemblance a

posteriori, conditionnellement à toute l'observation, de l'ensemble des éléments binaires (à la fois pour les bits d'information et les bits de parité) :

$$\mathrm{LLR}(b_k/y_{m,k}) = \log\left(\frac{P\left[b_k = +1/y_{m,k}\right]}{P\left[b_k = -1/y_{m,k}\right]}\right) \tag{11.71}$$

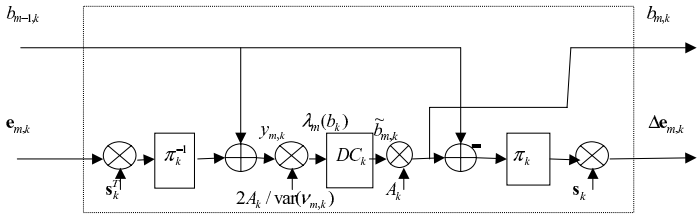


Figure 11.26 – Unité d'annulation d'interférence pour le décodeur Turbo-*SIC* en CDMA pour le k-ième utilisateur et à l'itération m.

Ce rapport est ensuite transformé en une estimation pondérée des éléments binaires :

$$\tilde{b}_{m,k} = E\left[b_k/y_{m,k}\right] = \tanh\left(\frac{1}{2}\mathrm{LLR}\left(b_k/y_{m,k}\right)\right) \tag{11.72}$$

L'estimation soft de à l'itération m est donnée par $b_{m,k} = A_k\tilde{b}_{m,k}$. La différence $(b_{m,k} - b_{m-1,k})$ est entrelacée par π_k avant l'étalement par $\mathbf{s}_k$. Le résultat ainsi obtenu $\Delta\mathbf{e}_{m,k}$ est soustrait au signal résiduel $\mathbf{e}_{m,k}$ pour obtenir le nouveau signal résiduel $\mathbf{e}_{m,k+1}$ de l'utilisateur suivant (si $k < K$) ou pour obtenir le nouveau signal résiduel $\mathbf{e}_{m+1,1}$ pour le premier utilisateur à l'itération suivante ($\mathbf{e}_{m,K+1} = \mathbf{e}_{m+1,1}$). Ici, $y_{m,k}$ est écrit sous la forme $y_{m,k} = A_k b_k + \nu_{m,k}$ où $\nu_{m,k}$ (interférence d'accès multiple résiduel plus le bruit additif) est approché par une variable aléatoire gaussienne centrée dont la variance est donnée par :

$$\mathrm{var}\left(\nu_{m,k}\right) = \sum_{i<k} A_i^2\rho_{i,k}^2\left(1 - \tilde{b}_{m,i}^2\right) + \sum_{i>k} A_i^2\rho_{i,k}^2\left(1 - \tilde{b}_{m-1,i}^2\right) + \sigma^2 \tag{11.73}$$

On montre que l'information extrinsèque de à l'itération m est donnée par :

$$\lambda_m(b_k) = \log\left(\frac{P\left[y_{m,k}/b_k = +1\right]}{P\left[y_{m,k}/b_k = -1\right]}\right) = \frac{2y_{m,k}A_k}{\mathrm{var}(\nu_{m,k})} \tag{11.74}$$

Cette information extrinsèque sert d'entrée au décodeur associé au k-ième utilisateur.

Quelques simulations

Pour donner une idée des performances du décodeur turbo-*SIC*, des séquences de Gold de taille 31 sont générées. Le turbocodeur de canal (de rendement $R = 1/3$) normalisé pour l'UMTS [11.8] est utilisé. On considère des

trames de 640 bits par utilisateur. Les entrelaceurs externes des différents utilisateurs sont produits de façon aléatoire. Les TEB et TEP sont moyennés sur l'ensemble des utilisateurs. Pour le turbo-décodeur de canal, l'algorithme Max-Log-*MAP* est utilisé avec un nombre d'itérations internes au turbo-décodeur de 8. La figure 11.27(a) donne les performances du décodeur turbo-*SIC* pour une, deux et trois itérations avec $K = 31$ utilisateurs (soit 100% de taux de charge) de même puissance. Les performances du détecteur mono-utilisateur et du détecteur conventionnel y sont également indiquées. La figure 11.27(b) fournit les performances en terme de TEP.

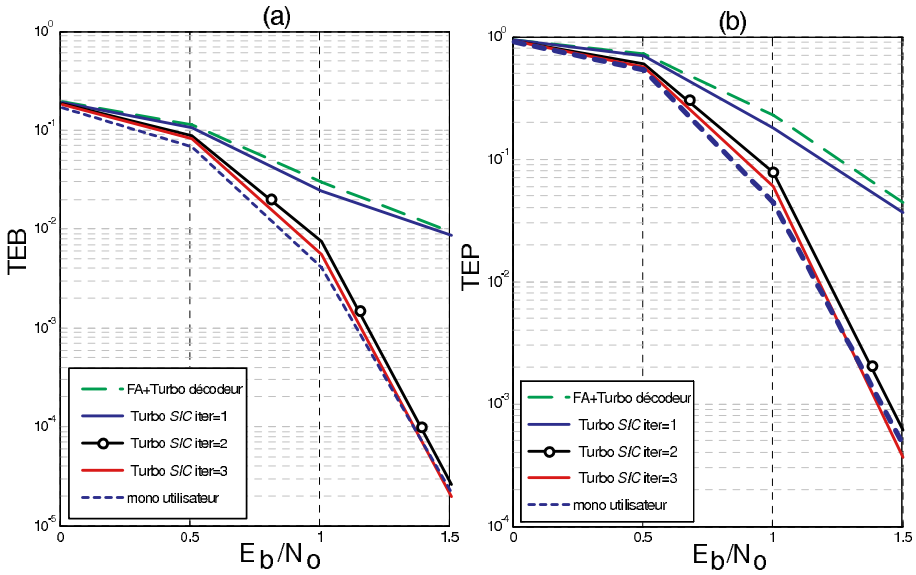


Figure 11.27 – Performance du décodeur turbo-*SIC* : (a) Taux d'Erreurs Binaires (TEB) moyen (b) Taux d'Erreurs de Paquets (TEP) moyen. $K = 31$ utilisateurs, facteur d'étalement de 31, taille de trame de 640 bits.

Détecteur turbo-*SIC*/*RAKE*

Dans le cas où le canal de propagation du k-ième utilisateur possède une réponse impulsionnelle à trajets multiples $c_k(t)$, il suffit de remplacer la fonction de désétalement par un filtre *RAKE* (filtre adapté à la séquence d'étalement convolué avec la fonction de transfert $c_k(t)$ dans l'unité $ICU_{m,k}$), et de remplacer la fonction d'étalement par la fonction d'étalement convoluée par $c_k(t)$. Cette nouvelle structure est baptisée : décodeur turbo-*SIC*/*RAKE*.

Le décodeur turbo-*SIC*/*RAKE* s'utilise en particulier dans le contexte de la liaison montante du système UMTS-FDD.

11.3 Conclusions

Dans ce chapitre, nous avons présenté les deux premiers systèmes à avoir bénéficié de l'application du principe turbo à un contexte autre que le codage correcteur d'erreurs. Dans la première partie, nous nous sommes attachés à décrire le principe de la turbo-égalisation. Il s'agit d'un échange réciproque et itératif d'information probabiliste entre l'égaliseur *SISO* et le décodeur *SISO*. L'égaliseur *SISO* peut prendre différentes formes selon le critère d'optimisation choisi. Nous avons présenté deux types d'égaliseurs *SISO* : l'égaliseur BCJR-*MAP*, basé sur la représentation en treillis du canal, et l'égaliseur MEQM, qui utilise le filtrage linéaire. Sous sa forme BCJR-*MAP*, l'égaliseur conduit à d'excellentes performances du turbo-égaliseur comparativement au récepteur conventionnel. Cependant, cette approche est souvent écartée en pratique car elle conduit à un coût de calcul très important. Nous avons cité diverses techniques de réduction de la complexité de l'égaliseur BCJR-*MAP*. Le turbo-égaliseur MEQM quant à lui offre un bon compromis performance-complexité. Pour de nombreuses configurations de transmission il conduit à des performances proches de celles offertes par le turbo-égaliseur BCJR-*MAP*, et cela pour une complexité raisonnable. De plus, contrairement au turbo-égaliseur BCJR-*MAP*, il est possible de mettre en œuvre une version adaptative du turbo-égaliseur MEQM qui réalise conjointement l'égalisation et la poursuite des variations de canal.

Dans la deuxième partie, nous nous sommes intéressés à l'application du principe turbo au domaine des communications multi-utilisateurs. Nous avons présenté un état de l'art des techniques conventionnelles de détection multi-utilisateurs. Notamment, les méthodes *PIC* et *SIC* d'annulation successives des interférences multi-utilisateurs on été décrites. Leurs structures particulières conduisent une exploitation relativement aisée du principe turbo dans le contexte d'une transmission multi-utilisateurs. Comme pour la turbo-égalisation, différents détecteurs peuvent être mis en œuvre basés sur des filtres MEQM ou des bancs de filtres adaptés, par exemple.

Dans ce chapitre, nous nous sommes volontairement restreints à la présentation de deux systèmes particuliers exploitant le principe Turbo. Cependant, de manière plus générale, tout problème de détection de symboles ou d'estimation des paramètres de transmission peut tirer parti du principe turbo. Ainsi, l'éventail de solutions permettant de traiter les interférences apportées par un système multi-antennes à l'émission et à la réception (*MIMO*) a été enrichi de techniques itératives telles que le turbo-*BLAST* (*Bell Labs layered space time*) [11.54]. Le challenge consiste à proposer des détecteurs *SISO* de complexité raisonnable sans sacrifier le débit et/ou les performances importantes potentielles de tels systèmes.

Nous pouvons également citer les efforts dédiés à la synchronisation des récepteurs. En effet, les gains de puissance apportés par le principe turbo conduisent à décaler le point de fonctionnement des systèmes vers de faibles rapports signal à bruit. Or, les moyens conventionnels de synchronisation n'étaient pas initialement prévus pour fonctionner dans ces conditions difficiles [11.55].

Une solution possible est alors d'intégrer la synchronisation au processus turbo. Dans [11.56], un état de l'art des méthodes turbo pour la synchronisation du rythme a récemment été présenté. De façon plus générale, lorsque le choix d'un traitement turbo à la réception est réalisé, il semble intéressant, voire nécessaire, d'adjoindre au récepteur un système d'estimation itérative des paramètres de transmission telle qu'une turbo-estimation de canal ou une turbo-synchronisation.

Parmi d'autres applications, la voie montante des futurs systèmes de communications radio-mobiles requerra toujours plus de débit, avec un nombre d'utilisateurs toujours croissant. C'est l'une des applications de prédilection du principe turbo, dont la généralisation sera indispensable pour répondre au perpétuel défi technologique posé par l'évolution des télécommunications.

11.4 Bibliographie

[11.1] J. G. Proakis, *Digital Communications*, 4$^{\text{th}}$ edition. McGraw-Hill, New-York, 2000.

[11.2] S. U. H. Qureshi, « Adaptive Equalization », *Proc. IEEE*, vol. 73, n ° 9, pp. 1349-1387, Sept. 1985.

[11.3] G. M. Vitetta, B. D. Hart, A. Mämmelä and D. P. Taylor, « Equalization Techniques for Single-Carrier Unspread Digital Modulations », *Wideband Wireless Digital Communications*, A. F. Molish, Ed. Prentice-Hall, Upper Saddle River, NJ, 2001, pp. 155-308.

[11.4] J.-M. Brossier, *Signal et Communication Numérique – Egalisation et Synchronisation*, Collection Traitement du Signal, Hermès, Paris, 1997.

[11.5] G. D. Forney Jr, « Maximum-Likelihood Sequence Estimation of Digital Sequences in the Presence of Intersymbol Interference », *IEEE Trans. Inform. Theory*, vol. IT-18, n ° 3, pp. 363-378, May 1972.

[11.6] M. S. Mueller and J. Salz, « A Unified Theory of Data-Aided Equalization », *Bell Sys. Tech. J.*, vol. 60, n ° 11, pp. 2023-2038, Nov. 1981.

[11.7] A. Gersho and T. J. Lim, « Adaptive Cancellation of Intersymbol Interference for Data Transmission », *Bell Sys. Tech. J.*, vol. 60, n ° 11, pp. 1997-2021, Nov. 1981.

[11.8] ETSI Digital Cellular Telecommunication System (Phase 2+). *GSM 05 Series*, Rel. 1999.

[11.9] P. R. Chevillat and E. Eleftheriou, « Decoding of Trellis-Encoded Signals in the Presence of Intersymbol Interference and Noise », *IEEE Trans. Commun.*, vol. 37, n ° 7, pp. 669-676, July 1989.

[11.10] W. Koch and A. Baier, « Optimum and Sub-Optimum Detection of Coded Data Disturbed by Time-Varying Intersymbol Interference », *Proc. IEEE. Global Telecommun. Conf. GLOBECOM'90*, San Diego, CA, 2-5 Dec. 1990, vol. 3, pp. 1679-1684.

[11.11] J. Hagenauer, « Soft-In / Soft-Out – The Benefits of Using Soft Decisions in all Stages of Digital Receivers », *Proc. 3rd Int. Symposium on DSP*

Techniques applied to Space Communications, Noordwijk, The Netherlands, Sept. 1992.

[11.12] P. Didier, *La Turbo-Egalisation et son Application aux Communications Radiomobiles*, Thèse de l'université de Bretagne Occidentale, Brest, France, Décembre 1996.

[11.13] C. Douillard, M. Jézéquel, C. Berrou, A. Picart, P. Didier and A. Glavieux, « Iterative Correction of Intersymbol Interference : Turbo-Equalization », *European Trans. Telecommun.*, vol. 6, n ° 5, pp. 507-511, Sept.-Oct. 1995.

[11.14] G. Bauch, H. Khorram and J. Hagenauer, « Iterative Equalization and Decoding in Mobile Communication Systems », *Proc.* 2^{nd} *European Personal Mobile Commun. Conf. EPMCC'97*, Bonn, Germany, Sept.-Oct. 1997, pp. 307-312.

[11.15] R. Le Bidan, *Turbo-Equalization for Bandwidth-Efficient Digital Communication over Frequency-Selective Channels*, Thèse de l'INSA de Rennes, Rennes, France, Nov. 2003.

[11.16] L. Bahl, J. Cocke, F. Jelinek and J. Raviv, « Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate », *IEEE Trans. Inform. Theory,* vol. IT-20, n ° 2, pp. 284-287, Mar. 1974.

[11.17] G. Bauch and V. Franz, « A Comparison of Soft-In/Soft-Out Algorithms for Turbo-Detection », *Proc. Int. Conf. Telecommun.*, ICT'98, Porto Carras, Greece, June 1998, pp. 259-263.

[11.18] S. ten Brink, « Designing Iterative Decoding Schemes with the Extrinsic Information Transfer Chart », *AEÜ Int. J. Electron. Commun.*, vol. 54, n ° 6, pp. 389-398, Nov. 2000.

[11.19] I. Lee, « The Effects of a Precoder on Serially Concatenated Coding Systems with an ISI Channel », *IEEE Trans. Commun.*, vol. 49, n ° 7, pp. 1168-1175, July 2001.

[11.20] K. Narayanan, « Effects of Precoding on the Convergence of Turbo Equalization for Partial Response Channels », *IEEE J. Select. Areas Commun.*, vol. 19, n ° 4, pp. 686-698, April 2001.

[11.21] A. O. Berthet, B. S. Ünal and R. Visoz, « Iterative Decoding of Convolutionally Encoded Signals over Multipath Rayleigh Fading Channels », *IEEE J. Select. Areas Commun.*, vol. 19, n ° 9, pp. 1729-1743, Sept. 2001.

[11.22] G. Colavolpe, G. Ferrari and R. Raheli, « Reduced-State BCJR-Type Algorithms », *IEEE J. Select. Areas Commun.*, vol. 19, n ° 5, pp. 849-859, May 2001.

[11.23] B. Penther, D. Castelain and H. Kubo, « A Modified Turbo-Detector for Long Delay Spread Channels », *Proc.* 2^{nd} *Int. Symp. on Turbo-Codes & Related Topics*, Brest, France, Sept. 2000, pp. 295-298.

[11.24] C. Fragouli, N. Al-Dhahir, S. N. Diggavi and W. Turin, « Prefiltered Space-Time M-BCJR Equalizer for Frequency-Selective Channels », *IEEE Trans. Commun.*, vol. 50, n ° 5, pp. 742-753, May 2002.

[11.25] S.-J. Lee, N. R. Shanbhag and A. C. Singer, « Area-Efficient High-Throughput VLSI Architecture for MAP-Based Turbo-Equalizer », *Proc. IEEE*

Workshop on Signal Processing Systems, SIPS 2003, pp. 87-92, Seoul, Korea, Aug. 2003.

[11.26] J. Hagenauer, E. Offer, C. Measson and M. Mörz, « Decoding and Equalization with Analog Non-Linear Networks », *European Trans. Telecommun.*, pp. 107-128, Oct. 1999.

[11.27] A. Picart, P. Didier and A. Glavieux, « Turbo-Detection : A New Approach to Combat Channel Frequency Selectivity », *Proc. IEEE Int. Conf. Commun.* ICC'97, Montréal, pp. 1498-1502, Canada, June 1997.

[11.28] G. Bauch and V. Franz, « Iterative Equalization and Decoding for the GSM System », *Proc. IEEE Veh. Technol. Conf.*, VTC'98, pp. 2262-2266, Ottawa, Canada, May 1998.

[11.29] V. Franz, *Turbo-Detection for GSM Systems – Channel Estimation, Equalization and Decoding*, Ph.D. Thesis, Lehrstuhl für Nachrichten Technik, Münich, Germany, Nov. 2000.

[11.30] N. Nefedov, M. Pukkila, R. Visoz and A. O. Berthet, « Iterative Receiver Concept for TDMA Packet Data Systems », *European Trans. Telecommun.*, vol. 14, n ° 5, pp. 457-469, Sept.-Oct. 2003.

[11.31] A. Glavieux, C. Laot and J. Labat, « Turbo Equalization over a Frequency Selective Channel », *Proc. Int. Symposium on Turbo Codes & Related Topics*, pp. 96-102, Brest, France, Sept. 1997.

[11.32] C. Laot, *Egalisation Auto-didacte et Turbo-Egalisation – Application aux Canaux Sélectifs en Fréquence*, Thèse de l'Université de Rennes I, Rennes, France, July 1997.

[11.33] C. Laot, A. Glavieux and J. Labat, « Turbo-Equalization : Adaptive Equalization and Channel Decoding Jointly Optimized », *IEEE J. Select. Areas Commun.*, vol. 19, n ° 9, pp. 1744-1752, Sept. 2001.

[11.34] X. Wang and H. V. Poor, « Iterative (Turbo) Soft Interference Cancellation and Decoding for Coded CDMA », *IEEE Trans. Commun.*, vol. 47, n ° 7, pp. 1046-1061, July 1999.

[11.35] D. Reynolds and X. Wang, « Low-Complexity Turbo-Equalization for Diversity Channels », *Signal Proc.*, vol. 81, n ° 5, pp. 989-995, May 2001.

[11.36] M. Tüchler, A. C. Singer and R. Kötter, « Minimum Mean-Squared Error Equalization Using A Priori Information », *IEEE Trans. Signal Processing*, vol. 50, n ° 3, pp. 673-683, Mar. 2002.

[11.37] M. Tüchler, R. Kötter and A. C. Singer, « Turbo-Equalization : Principles and New Results », *IEEE Trans. Commun.*, vol. 50, n ° 5, pp. 754-767, May 2002.

[11.38] C. Laot, R. Le Bidan and D. Leroux, « Low-Complexity MMSE Turbo Equalization : A Possible Solution for EDGE », *IEEE Trans. Wireless Commun.*, VOL. 4, NO.3, May 2005.

[11.39] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd edition. The Johns Hopkins University Press, Baltimore, 1996.

[11.40] A. Dejonghe and L. Vandendorpe, « Turbo-Equalization for Multilevel Modulation : An Efficient Low-Complexity Scheme », *Proc. IEEE Int.*

Conf. on Commun., ICC 2002, vol. 3, pp. 1863-1867, New-York City, NY, 28 Apr.-2 May 2002.

[11.41] C. Langlais and M. Hélard, « Mapping optimization for turbo-equalization improved by iterative demapping », *Electronics Letters*, vol. 38, n ° 2, oct. 2002.

[11.42] F. Volgelbruch and S. Haar, « Improved Soft ISI Cancellation for Turbo Equalization using Full Soft Output Channel Decoder's Information », *Proc. Globecom 2003*, pp. 1736-1740, San Francisco, USA, December 2003.

[11.43] M. Hélard, P.J. Bouvet, C. Langlais, Y.M. Morgan, I. Siaud, « On the performance of a turbo equalizer including Blind Equalizer over Time and Frequency Selective Channel. Comparison with an OFDM system », *Proc. Int. Symposium on Turbo Codes & Related Topics*, pp. 419-422, Brest, France, Sept. 2003.

[11.44] R. Le Bidan, C. Laot and D. Leroux, « Real-Time MMSE Turbo-Equalization on the TMS320C5509 Fixed-Point DSP », *Proc. IEEE Int. Conf. on Accoustics, Speech and Signal Processing*, ICCASP 2004, vol. 5, pp. 325-328, Montreal, CA, 17-21 May 2004.

[11.45] C. Langlais, *Etude et amélioration d'une technique de réception numérique itérative : Turbo-Egalisation*, Thèse de l'INSA de Rennes, Rennes, France, Nov. 2002.

[11.46] R. Otnes, *Improved receivers for digital High Frequency communications : iterative channel estimation, equalization, and decoding (adaptive turbo equalization)*, Doctor Engineer Thesis of the Norwegian university of science and technology, 2002.

[11.47] S. Verdu, *Multiuser detection*. Cambridge University Press, 1998.

[11.48] L.K. Rasmussen, T.J. Lim and A.L. Johanson, « A matrix-algebraic approach to successive interference cancellation in CDMA », *IEEE Transactions on Communications*, vol. 48, n ° 1, pp. 145-151, January 2000.

[11.49] G. Dongning, L.K. Rasmussen and T.J. Lim, « Linear parallel interference cancellation in long-code CDMA multiuser detection », *IEEE Journal on Selected Areas In Communications*, vol. 17, n ° 12, pp. 2074-2081, December 1999.

[11.50] M.K. Varanasi and T. Guess, « Optimum decision feedback multiuser equalization with successive decoding achieves the total capacity of the gaussian multiple-access channel », *Conference Record of the Thirty-First Asilomar Conference on Signals, Systems & Computers*, 2 :1405-1409, 2-5 Nov.1997.

[11.51] A. Tarable, G. Montorsi and S. Benedetto, « A linear Front End for Iterative Soft Interference Cancellation and Decoding in Coded CDMA », *Proc. ICC'01*, International Conference on Communications, June 11-14 2001.

[11.52] M.C. Reed and P.D. Alexander, « Iterative Multiuser detection using antenna arrays and FEC on Multipath channels », *IEEE Journal on Selected Areas in Communications*, vol. 17, n ° 12, pp. 2082-2089, December 1999.

[11.53] Third Generation Partnership Project (3GPP) Technical Specification Group (TSG) Radio Access Network Working Group. *Multiplexing and channel coding (FDD)*, Technical report 3GPP, 2002.

[11.54] S. Haykin, M. Sellathurai,, Y. de Jong and T. Willink, « Turbo-MIMO for wireless communications », *IEEE Communications Mag.*, pp. 48-53, October 2004.

[11.55] C. Langlais, M. Hélard and M. Lanoiselée, « Synchronisation in the carrier recovery of a satellite link using turbo-codes with the help of tentative decisions », *Proc. IEE Colloqium on Turbo Codes i n Digital Broadcasting*, November 1999.

[11.56] J. R. Barry, A. Kavcic, S. W. McLaughlin, A. Nayak and W. Zeng, « Iterative Timing Recovery », *IEEE Signal Processing Mag.*, vol. 21, n ° 1, pp. 89-102, Jan. 2004.

Index

AMRC(Accès Multiple à Répartition par Code), 378
APP(*a posteriori probability*), 166, 230, 314
arbre d'un code, 171
ARP (*almost regular permutation*), 223
ARQ (*Automatic Repeat reQuest*), 87, 245, 363

BABG, voir bruit additif blanc gaussien
bande de cohérence, 70
bande de fréquences, 2, 31, 325, 344
BCJR (Bahl-Cocke-Jelinek-Raviv), 166, 351
belief propagation, voir propagation de croyance
Berlekamp-Massey (algorithme de), 144, 280
Bernoulli (distribution de), 85
BICM (*Bit-Interleaved Coded Modulation*), 333, 349
bipartite (graphe), 283
bits de fermeture, 215
bits de redondance, 180, 209, 289, 329
bits systématiques, 332
bruit additif blanc gaussien, 9, 37, 189, 230, 295, 347

canal à bande limitée, 59, 93
canal à évanouissements, 70, 331
canal binaire symétrique, 84, 132, 185
canal de Rayleigh, 73, 95, 228
canal de transmission, 7, 83, 343
canal discret équivalent, 342
canal gaussien, 9, 37, 93, 228, 300
canal radio-mobile, 184, 342
canal sélectif en fréquence, 78, 344
capacité, 16, 85, 86, 176, 314, 327, 360
canal binaire symétrique, 8
CDMA (*Code Division Multiple Access*), 342
Chase (algorithme de), 138, 265
Chase-Pyndiah (algorithme de), 268
chemin survivant, 56, 186, 361
Chien (algorithme de), 149
circulaire(codage,terminaison), 182, 191, 215
codage aléatoire, 88, 210
codage aléatoire systématique, 209
code à faible densité, 288
code à répétition, 78, 117
code BCH, 125, 280
code catastrophique, 175
code circulaire, 231
code convolutif, 165, 167, 207, 293
code convolutif systématique récursif, 209, 239, 326
code convolutif systématique récursif circulaire, 191, 203, 209
code cyclique, 122
code de Golay, 130
code de Hadamard, 118
code de Hamming, 117, 200, 260
code de parité, 116, 128, 284
code de Reed-Muller, 119
code de Reed-Solomon, 130, 143, 259
code double-binaire, 167, 213, 243, 335
code dual, 113, 273
code étendu, 115
code linéaire, 90, 115, 287
code m-binaire, 167, 239
code produit, 18, 115, 199, 246, 259, 261, 265
code raccourci, 115
code séparable, 5
code systématique, 194, 200, 212
code systématique récursif, 170
cohérent (récepteur), 38, 42

comportement asymptotique, 14, 245, 338
concaténation multiple, 202, 209
concaténation parallèle, 199, 202, 209, 212
concaténation série, 199, 212
constellation, 3, 20, 105, 325, 344
convergence, 16, 199, 211, 249, 297, 317, 336, 360
CRC (*Cycle Redundancy Code*), 128
CSR, voir code convolutif systématique récursif
CSRC, voir code convolutif systématique récursif circulaire

décodage ligne-colonne, 261
décodage séquentiel, 166, 172
décorrélateur, 381
demapping à entrée et sortie pondérées (SISO), 370
désordre, 219, 221, 223
détecteur d'erreur, 7
détecteur-SIC/RAKE, 387
détection d'erreur, 132
détection multi-utilisateurs, 379
diagramme de l'œil, 61
diagramme en treillis, 172, 346
distance euclidienne, 9, 10, 138, 326, 345
distance euclidienne effective, 329
distance libre, 176, 216, 240
distance minimale de Hamming, 6, 101, 114, 176, 197, 207, 245, 259, 300
distance spatiale cumulée, 217
diversité, 2, 74, 75, 216, 350
DMH, voir (istance minimale de Hamming)6
DRP (*dithered relatively prime*), 223
dsp (densité spectrale de puissance), 24

EDGE (*Enhanced Data Rate for GSM Evolution*), 361
égalisation, 81, 345, 349
égalistation adaptative, 372
égaliseur à entrée et sortie pondérées, 351
égaliseur à retour de décision, 347
égaliseur de Viterbi, 351
égaliseur linéaire, 347
entrée ferme, 8, 134, 143, 261, 294
entrée pondérée, 134
entrée souple, 8, 265, 276, 294, 295
entrelacement, 197, 203, 216, 306, 312, 327, 333, 349, 360
entropie, 86
équation de parité, 284
étalement de spectre, 342, 344
Euclide (algorithme de), 144, 280
évolution de densité, 298
EXIT (*EXtrinsic Information Transfer*), 249, 299, 360
extrinsèque (information), 208, 225, 265, 294, 327, 351, 358

Fang-Battail (algorithme de), 272
fermeture de treillis, 165, 167, 190
filtre adapté, 40
fonction d'erreur complémentaire, 13, 44, 334
fonction de transfert, 177, 249

gain asymptotique, 15, 142, 200, 245
Galois (corps de), 109
Gilbert-Varshamov (borne de), 89, 259
GMSK (*Gaussian Minimum Shift Keying*), 35, 350
Gray (codage de), 23, 47, 333, 372
GSM (*Global System for Mobile communication*), 37, 346

Hadamard (transformée), 273, 274
Hartmann-Nazarov (algorithme de), 272

ICU (*interference cancellation unit*), 383
IES, voir interférence entre symboles
impulsion d'erreur, 246
information mutuelle, 85, 249

interférence entre symboles, 60, 64, 343, 361
irrégularité (profil d'), 288, 298

LDPC (*Low Density Parity Check*), 18, 83, 202, 283
limite de Shannon, 2, 93, 208
LMS (*Least Mean Square*), 364
Log-MAP, 326, 353
logarithme du rapport de vraisemblance, 92, 225, 285, 295, 335, 352
longueur de contrainte, 168
LRC (Raised Cosine pulse), 35
LRV, voir logarithme du rapport de vraisemblance

m-binaire, voir code m-binaire
machine à états, 174
MAP, voir *maximum a posteriori*
mapping à entrée et sortie pondérées (SISO), 365
MAQ-16, 27, 105, 334
MAQ-M (modulation d'amplitude en quadrature), 27, 49
matrice de contrôle, 113, 202, 270, 287
matrice génératrice, 110, 290, 291
Max-Log-MAP, 226, 232, 277, 326, 353
maximum a posteriori, 166, 189, 230, 277, 326, 353, 354
maximum de vraisemblance, 6, 97, 134, 185, 243, 265, 350
MCT, voir modulation codée en treillis
MDA-M (modulation par déplacement d'amplitude à M états), 22
MDF-M (modulation de fréquence à M états), 29
MDP-2, 25, 47, 97, 137, 229, 365
MDP-4, 48, 97, 101, 137, 229, 244, 377
MDP-8, 103, 244, 362
MDP-M (modulation par déplacement de phase à M états), 25
mémoire de code, 187, 207
MEQM (Minimisation de l'Erreur Quadratique Moyenne), 342, 347, 363, 382
métrique de branche, 56, 185, 232, 326, 356
métrique de nœud, 186
MLSD (*Maximum Likelihood Sequence Detection*), 345
MMSE (*Minimum Mean Square Error*), voir MEQM
modulation codée à bits entrelacés, voir BICM
modulation codée en treillis, 325
modulation turbocodée, 331
Morse, 3
mot de code, 5, 109, 209, 260, 302
mots croisés (turbo), 204
MSK (*minimum shift keying*), 32
multiplicité, 14, 89, 176, 247
multiporteuses, 78

niveau de protection, 333
Nyquist (critère de), 63

OFDM (*Orthogonal Frequency Division Multiplexing*), 78, 344
OOK (*On Off Keying*), 22
ou-exclusif, 5

papillon, 173
permutation, 5, 203, 209, 216, 217, 223, 290, 332
Peterson (algorithme de), 144
PIC (*Parallel Interference Cancellation*), 383
PNP (processeur de nœud de parité), 302, 314
PNV (processeur de nœud de variable), 302, 313
poids de Hamming, 6, 113, 175
poinçonnage, 167, 202, 241, 332
polynôme générateur, 121, 125, 130, 169
processeur de traitement de signal (DSP), 10, 364
processeurs de traitement de signal (DSP), 235
produit scalaire, 10
propagation de croyance, 277, 283

quantification, 8, 294

Reddy-Robinson (algorithme de), 262
redondance, 1, 109, 169, 201, 209, 289
remontée du treillis, 186
rendement de codage, 7, 109, 289
représentation polynomiale, 120, 170
réseau d'interconnexion, 305
RTZ (*return to zero*), 176, 209

seuil de convergence, 245, 352
shannon (unité), 3
SIC (*Successive Interference Cancellation*), 383
SISO (*Soft-Input/Soft-Output*), 12, 198, 229, 279, 351
sortie ferme, 11, 184
sortie pondérée, 11, 184, 265
sortie souple, 11, 283
SOVA (*soft-output Viterbi algorithm*), 184, 208, 351
spectre de distances, 177
sphere-packing, 98
SubMAP, voir Max-Log-MAP
syndrome, 132, 135, 143, 262

tail-biting, 191
taux de redondance, 2
TEB (taux d'erreurs binaires), 12
temps de cohérence, 73
TEP (taux d'erreurs de paquets), 12
terminaison, 192, 215
TMCT, voir turbo-modulation codée en treillis
tout 0 (séquence), 5, 176, 200, 246, 298
treillis, 18, 56, 172, 302, 346
turbo-CDMA, 384
turbo décodage de codes produits, 268
turbo-détection, 353
turbo-détection multi-utilisateurs, 378
turbo-égalisation, 342, 351
turbo-égalisation selon la Minimisation de l'Erreur Quadratique Moyenne (MEQM), 363
turbo-MAQ-16, 327
turbo-MDP-8, 327
turbo-modulation codée en treillis, 325
turbo-synchronisation, 389
turbocode, 12, 18, 83, 207, 211, 278, 293
turbo-égalisation adaptative, 374
turbo-égalisation selon le Maximum A Posteriori (MAP), 353
turbo-estimation de canal, 389

UMTS, 228

Viterbi (algorithme), 12

Collection *IRIS*
Dirigée par Nicolas Puech

Ouvrages parus :

– *Méthodes numériques pour le calcul scientifique. Programmes en Matlab*
A. Quarteroni, R. Sacco, F. Saleri, Springer-Verlag France, 2000

– *Calcul formel avec MuPAD*
F. Maltey, Springer-Verlag France, 2002

– *Architecture et micro-architecture des processeurs*
B. Goossens, Springer-Verlag France, 2002

– *Introduction aux mathématiques discrètes*
J. Matousek, J. Nesetril, Springer-Verlag France, 2004

– *Les virus informatiques : théorie, pratique et applications*
É. Filiol, Springer-Verlag France, 2004

– *Introduction pratique aux bases de données relationnelles. Deuxième édition*
A. Meier, Springer-Verlag France, 2006

– *Bio-informatique moléculaire. Une approche algorithmique*
P.A. Pevzner, Springer-Verlag France, 2006

– *Algorithmes d'approximation*
V. Vazirani, Springer-Verlag France, 2006

– *Techniques virales avancées*
É. Filiol, Springer-Verlag France, 2007

À paraître :

– *Introduction à Scilab. Deuxième édition*
J.P. Chancelier, F. Delebecque, C. Gomez, M. Goursat, R. Nikouhah, S. Steer, Springer-Verlag France, 2007

– *Forces de la programmation orientée objet. De la théorie à la pratique*
J. Pasquier, P. Fuhrer, A. Gachet, Springer-Verlag France, 2007

Achevé d'imprimer sur les presses de l'Imprimerie BARNÉOUD
B.P. 44 - 53960 BONCHAMP-LÈS-LAVAL
Dépôt légal : février 2007 - N° d'imprimeur : 701072
Imprimé en France