Towards ideal codes : looking for new turbo code schemes

Dhouha Kbaier

N°d'ordre : 2011telb0204

Sous le sceau de l'Université européenne de Bretagne

Télécom Bretagne

En habilitation conjointe avec l'Université de Bretagne-Sud

École Doctorale – sicma

Towards ideal codes: looking for new turbo code schemes

Thèse de Doctorat

Mention : Sciences et Technologies de l'Information et de la Communication

Présentée par **Dhouha Kbaier Ben Ismail**

Département : Électronique

Laboratoire : Lab-STICC Pôle: CACS

Directrice de thèse : Catherine Douillard

Jury :

Mme Marion Berbineau, directrice de recherche du Leost à l'IFSTTAR (Rapporteur)
M. Vahid Meghdadi, maître de conférences HDR à l'ENSIL (Rapporteur)
Mme Catherine Douillard, professeur à Télécom Bretagne
Mme Sylvie Kerouédan, maître de conférences à Télécom Bretagne
M. Christophe Jego, professeur des universités à l'ENSEIRB-MATMECA
Mme Laura Conde Canencia, maître de conférences à l'UBS

Acknowledgements

FIRST of all, I would like to thank all the members of the Electronics Department for welcoming me in your laboratory. I have met a lot of people outside and inside the work sphere that contributed to make the Ph.D adventure possible and enjoyable.

I would like to thank my supervisors, Prof. CATHERINE DOUILLARD and Dr. SYLVIE KEROUÉDAN, for their motivation, support, patience and guidance during these three years. I am grateful to my supervisors for giving me the opportunity to develop my research experience. It was always a pleasure to share with you new results. Thank you for reviewing the articles and draft versions of this thesis. Thank you for your writing skills and diligence for the corrections of my manuscript.

Furthermore, I would like to thank the members of my reading committee, Dr. VAHID MEGHDADI and Prof. MARION BERBINEAU for taking the time to assess my manuscript. Your advice for some final corrections was greatly appreciated. I am also thankful to the members of my Ph.D defense committee : Prof. CHRISTOPHE JEGO and Dr. LAURA CONDE CANENCIA for their valuable comments and suggestions.

Although doing a Ph.D is somehow a lonely process, you need to collaborate with many people whose skills allow the project to mature. I would like to thank Dr. ABDEL NOUR CHARBEL and Dr. YANNICK SAOUTER for useful discussions and valuable suggestions on the design of suitable permutations for irregular turbo codes. Special thanks to Prof. CLAUDE BERROU. You are the proof that kindness and determination can be united in one person. Thank you for all the fruitful discussions. Thank you for the smile and the energy you gave me every time you enter to my office. Thank you for your highest interest to my research. Your constant support helped me through rough times. I will never forget your famous sentence "Bonjour jeunes gens, comment avance la science dans ce bureau ?" and all the discussions from science to philosophy...

I thank all those who strongly contributed to the progress of this thesis by their advice and several discussions. To all the Ph.D students, I want to say that it was a great pleasure to work with you. Thanks to my wonderful colleagues who share the same office with me: VINCENT and NICOLAS. Thank you for good times and nice company. Thanks to HAÏFA, OSCAR, CAMILO, RACHID, ATIF, PURUSH and SALIM. TRUNG: You have already learnt many words in Arabic. CAMILO was even jealous and decided to catch. ROUA and AMMAR: I think that I am now an expert in the Lebanese cooking thanks to your advice and your menus of the weekend! Thank you for GHISLAINE who cleans every day the offices and makes them pleasant. Thank you for her chocolates, her bouquets of lily of the valley and all the happiness she brought to me.

Last but not least, I would like to thank my parents who have always given me the strength and wisdom to be sincere in my work, for setting high moral standards and supporting me through their hard work, and for their unselfish love and affection; my husband: thank you for your support, encouragement, help and love; my two sisters; my dear uncles for their encouragement and love. Thanks to all the members of my family for what I have accomplished so far.

Abstract

WHILE turbo codes (TCs) offer performance very close to the Shannon limit in the so-called waterfall region, they suffer from a flattening effect due to a poor minimum distance. In future system generations, low error rates will be required to open the way to real-time and demanding applications, such as TV broadcasting or videoconferencing. Therefore, state-of-the-art TCs are no longer suitable for these kinds of applications and more powerful coding schemes are required. At the same time, a reasonable complexity should be preserved.

The first part of this thesis is dedicated to explore a new hybrid concatenation structure combining both parallel and serial concatenation based on a 3-dimensional (3D) code, simply derived from the classical TC by concatenating a rate-1 post-encoder at its output. First, we search for efficient post-encoder structures by means of EXtrinsic Information Transfer (EXIT) charts, especially when transmissions over non Gaussian channels (fading channels, erasure channels) are considered. Other key parameters of the 3D TC are sensibly selected. Various simulations show that 3D TCs have a better asymptotical behaviour with respect to classical TCs. An optimization method, in the case of the 3GPP2 code, allows the minimum distance of the 3D TC to be even more increased. On the other hand, a loss in the convergence threshold and an increase in complexity are observed. In order to reduce the observable loss of convergence, a time varying post-encoder is proposed. The time-varying technique reduces the loss of convergence by 10% to 50% of its value expressed in dB. However, there is no need to use this technique, when the 3D TC is associated with a high order modulation such as M-PSK or M-QAM. Indeed, a specific Gray mapping where the systematic bits and the post-encoded parity bits are placed in the best protected binary positions of the modulation scheme, allows the loss in convergence to be transformed into a gain at low signal to noise ratios (SNRs). Thus, we obtain 3D TCs which perform better than the classical TCs in both the waterfall and the error floor regions.

The second part of this study deals with irregular TCs. Here, the problem is the opposite of the precedent. Although irregular TCs can achieve performance closer to capacity, their asymptotic performance is very poor. First of all, the degree profile is selected by means of EXIT diagrams. Then, the design of powerful permutations suited for such code structures is considered. Graph-based permutations using the Dijkstra's algorithm and an estimation of the minimum distance, improve the distance properties of these codes. Nevertheless, this task takes a lot of time for the large blocks, and a memory is required to store the interleaved addresses. To take advantage of the results in the first part of the thesis, and in order to combine both studies, a new modified structure is

proposed. The association of irregular TCs with the same post-encoder used for 3D TCs results in irregular turbo coding schemes which perform better than regular TCs at low and high SNRs at the same time.

Keywords: turbo code, 3-dimensional turbo code, 3GPP2 code, minimum distance, convergence threshold, EXIT chart, time-varying trellis, modulation, decoding complexity, irregular turbo code, degree profile, Dijkstra's algorithm, correlation graph.

Résumé

Au début des années 90, les turbocodes (TCs) [18] ont révolutionné le domaine du codage de canal. Ils ont été adoptés dans plusieurs standards de télécommunications (3GPP, DVB-RCS/RCT, WiMAX, ...) [1, 2, 4, 5]. Le décodage de ces codes, constitués de la concaténation parallèle de deux codes convolutifs séparés par un entrelaceur, fait appel à un processus itératif basé sur deux décodeurs élémentaires s'échangeant des informations afin d'améliorer la correction au fil des itérations.

Tandis que les TCs existants présentent des performances très proches des limites théoriques pour les taux d'erreurs moyens et élevés, atteindre les taux d'erreurs très faibles requis par les futures applications de diffusion numérique ou de visioconférence requiert l'utilisation de codes plus puissants. Néanmoins, une complexité de décodage raisonnable doit être maintenue. L'objectif principal de la thèse consiste à investiguer de nouvelles structures dérivées des TCs classiques afin d'atteindre des taux d'erreurs très faibles. Il s'agit alors de trouver des architectures qui conduisent à un compromis performance/complexité de décodage non encore atteint avec de tels codes.

Les turbocodes 3D

Une première partie de l'étude est consacrée à l'investigation d'une nouvelle structure concaténée hybride 3D combinant les principes de concaténation parallèle et série. Celle-ci est simplement dérivée d'un TC classique en concaténant un post-codeur de rendement 1 à sa sortie [20, 21]. Dans la figure 0.1, on retrouve le schéma de principe d'un turbocodeur 3D.

Comparé au TC classique, le TC 3D comporte en plus des deux codeurs élémentaires concaténés en parallèle et séparés par un entrelaceur :

- ◇ Un convertisseur parallèle/série (P/S) qui prend périodiquement les bits de parités à post-coder et les groupe dans un seul bloc de P bits,
- ◇ Une permutation Π' qui permute les bits de parité avant de les envoyer au post-codeur,
- ◇ Et un post-codeur de rendement 1. Seule une fraction λ ($0 < \lambda \leq 1$) des bits de parité en provenance de chaque codeur est post-codée.

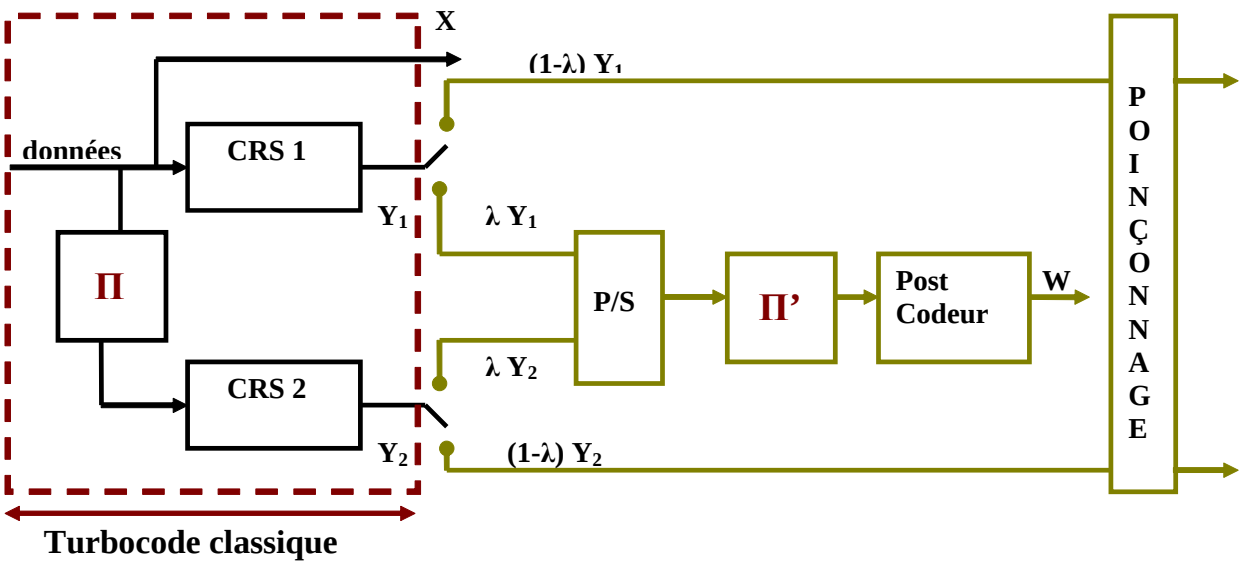


Figure 0.1: Schéma de principe d'un turbocode 3D.

Choix du post-codeur

Le choix du post-codeur est crucial pour la performance du code. Il doit satisfaire les conditions suivantes :

1. Son décodeur doit être simple, ajoutant peu de complexité au turbo décodeur classique, tout en en étant capable de manipuler des informations pondérées en entrée et en sortie.

2. Puisque des taux d'erreurs très faibles sont recherchés, il faudra éviter les effets de bord au niveau du décodeur. Par la suite, le post-codeur doit être un code en bloc homogène.

3. À la première itération (donc sans aucune information de redondance en entrée), le pré-décodeur associé au post-codeur de rendement 1 ne doit pas entraîner une grande amplification d'erreur, afin d'éviter une importante perte en convergence.

Dans notre analyse, les diagrammes EXIT [103] nous permettent de rechercher des structures de post-codeurs adaptés aux transmissions aussi bien sur canal gaussien que sur canal à évanouissements de type Rayleigh. Le post-codeur $(5, 4)$, dont les polynômes générateurs en octal sont 5 pour la récursivité et 4 pour la redondance, a été sélectionné dans différentes simulations du TC 3D.

Choix de λ

Comme le montre la figure 0.2, la valeur de λ à choisir est un compromis entre la perte en convergence et la distance minimale souhaitée. Le choix d'une grande valeur de λ

pénalise le décodeur d'un point de vue seuil de convergence. Cela résulte du fait que le décodeur associé au post-codeur, ne profite d'aucune information de redondance à la première itération, et multiplie par la suite les erreurs pendant le premier traitement. Cependant, plus λ est grand et meilleures seront les performances asymptotiques du code.

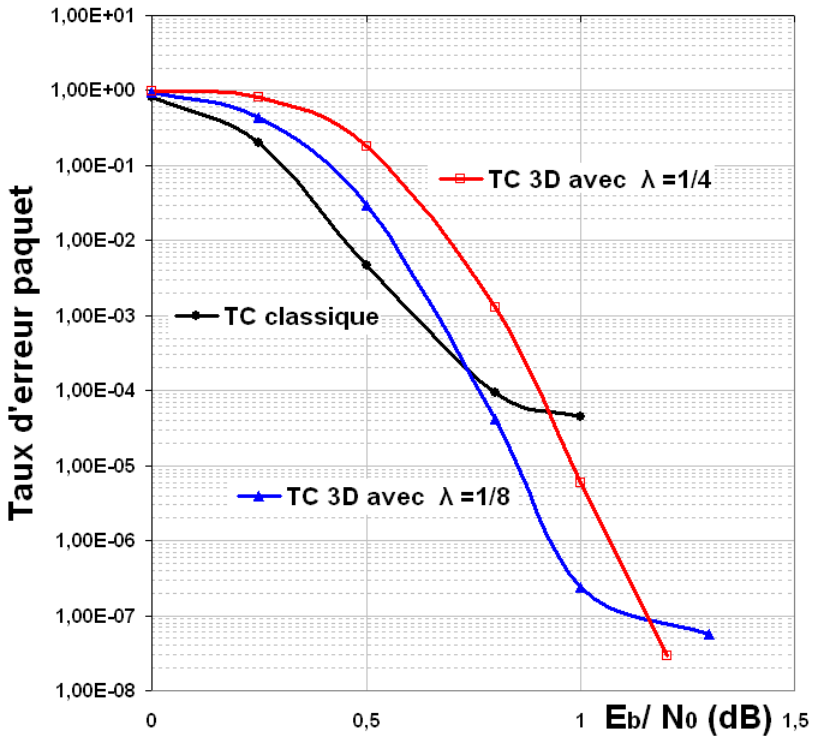


Figure 0.2: Comparaison des performances d'un TC classique et d'un TC 3D sur canal gaussien avec un entrelacement 3GPP2 pour un bloc de $k = 3066$ bits, un rendement $R = 1/3$ et deux valeurs de λ différentes: $\lambda = 1/4$ et $\lambda = 1/8$. L'algorithme de décodage est le MAP avec 10 itérations.

Choix de Π'

Tout d'abord, la permutation Π' intervient car on se retrouve dans un schéma de codage avec concaténation de codes. Comme tout entrelacement, Π' évite de perdre des données entières lorsque les erreurs sont produites en rafales. De plus, Π' joue un rôle important pour décorréler l'information extrinsèque sur les bits de parités post-codées.

Afin d'optimiser Π', nous avons testé différents entrelaceurs comme l'entrelaceur aléatoire et d'autres permutations plus structurées comme la permutation régulière. Nous avons constaté que le spread est la propriété la plus importante. Comme la permutation régulière atteint la borne supérieure du spread [24], c'est cette permutation qui a été sélectionnée dans différentes simulations du TC 3D. Π' est donc définie par la relation de congruence suivante :

$$i = \Pi'(j) = P_0 i + i_0 \,\%\, P$$

où P est le nombre de bits de parités post-codées, i $(1 \leq i \leq P)$ l'adresse dans l'ordre naturel, j $(1 \leq j \leq P)$ l'adresse dans l'ordre entrelacé, P_0 un entier premier avec P et i_0 l'indice de départ.

Vers une optimisation des turbocodes 3D

Les TCs 3D offrent de meilleures performances que les TCs classiques pour des rapports signal à bruit élevés sauf dans le cas des rendements élevés. Afin d'améliorer les performances asymptotiques de ces codes, nous avons proposé une méthode d'optimisation du spectre de distance. Cette méthode s'applique à n'importe quelle famille de TCs du moment où le spectre de distance présente des mots de codes ayant de faibles multiplicités. Dans le cas du code 3GPP2 [3], les bits de fermeture font que le début et la fin du bloc représentent des points singuliers dans le treillis et causent la troncature des mots de codes. On obtient donc des mots de codes de multiplicités faibles. L'application de la méthode proposée permet d'augmenter d'autant plus la distance minimale du code 3D en optimisant le spectre de distance.

Pour le TC 3D, on observe en revanche une dégradation du seuil de convergence et une augmentation de la complexité. La complexité additionnelle est principalement due à l'implémentation du prédécodeur (associé au post-codeur) et l'échange de l'information extrinsèque sur les bits de parités post-codées entre le prédécodeur et le turbodécodeur classique (car il faut calculer ces informations extrinsèques et les stocker). Nous avons fait une étude de la complexité et nous avons estimé la complexité calculatoire. Nous avons constaté que la complexité additionnelle relative est d'autant moins importante quand le nombre de processeurs placés en parallèle augmente. Ceci dépend à la fois du débit et de la technologie implémentée.

En ce qui concerne le seuil de convergence du TC 3D, deux axes de recherche sont proposés afin de résoudre le problème de la perte en convergence:

- Le premier consiste à utiliser un post-codeur avec un treillis variant dans le temps. Les premiers travaux concernant le treillis variant dans le temps datent de 1974 où Costello [33] montre que cette technique permet d'améliorer le comportement des codes convolutifs. Cette voie a été explorée par la suite dans la littérature [62, 68, 76, 79]. Dans notre étude, nous nous intéressons également à un post-codeur avec treillis variant dans le temps. Dans notre version du treillis variant dans le temps, le polynôme de récursivité est toujours le même c'est-à-dire 5. Cependant, deux redondances $w_1 = 4$ et $w_2 = 7$ sont alternées dans le temps au lieu d'avoir une seule. Dans le treillis du code $(5, 7 : 4)$, nous avons observé deux chemins correspondant à la même séquence tout-zéro. Afin d'éviter toute ambiguité dans le processus de décodage, nous avons effectué une petite modification sur ce code:

  Au lieu d'alterner les redondances w_1 et w_2 au cours du temps, on remplace périodiquement w_1 par w_2 comme le montre la figure 0.3. Le choix de la période de remplacement L résulte d' un compromis convergence/distance. Ce choix peut être affiné par une étude EXIT en sélectionnant la valeur de L qui donne le seuil de convergence le plus faible.

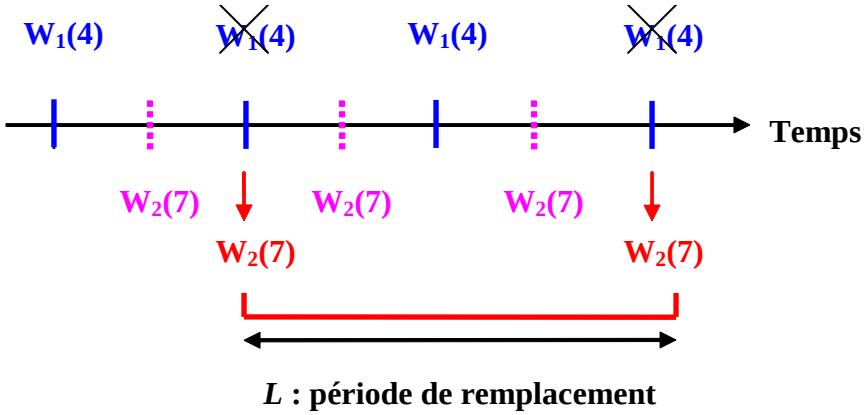


Figure 0.3: Post-codeur avec treillis variant dans le temps modifié.

L'application de la technique du treillis variant dans le temps permet de réduire la perte en convergence observée pour le TC 3D de 10% à 50% de sa valeur exprimée en dB. Nous avons systématiquement vérifié que les performances asymptotiques ne sont pas dégradées. En effet, pour une mémoire de code donnée, le choix du post-codeur n'influence pas la distance minimale du TC 3D. Cependant, plus la distance locale du post-codeur est élevée, meilleur sera le niveau de l'information extrinsèque échangée entre le prédécodeur et le turbo décodeur classique. La technique du treillis variant dans le temps accélère donc la convergence du TC 3D.

- ◇ Le deuxième axe consiste à associer le TC 3D avec des modulations d'ordre élevé comme les Modulations par Déplacement de Phase (MDP)-M ou les Modulations d'Amplitude sur deux porteuses en Quadrature (MAQ)-M. Lorsque les bits systématiques et les bits de parités post-codés sont affectés aux places binaires les mieux protégées par la modulation, la perte en convergence est transformée en gain. On obtient donc un TC 3D plus performant que le TC classique à la fois en termes de seuil de convergence et de performance asymptotique.

Les turbocodes irréguliers

Dans notre étude sur les TCs 3D, nous avons constaté qu'il est intéressant d'apporter de l'irrégularité dans un code afin d' en améliorer les performances. La seconde partie de la thèse est consacrée par la suite à l'étude des TCs irréguliers. Des travaux dans les années 2000 sur les codes LDPC irréguliers montrent un gain de codage significatif lorsque les degrés des nœuds de bits codés et les degrés des nœuds de parités ne sont pas tous identiques [71, 74]. Frey et MacKay [51] ont aussi introduit de l'irrégularité aux TCs. Sawaya et Boutros [92] ont repris leurs travaux afin d'améliorer les propriétés de distance des TCs irréguliers. Le problème ici est l'inverse du précédent. En effet, les TCs irréguliers ont de très bonnes performances pour des rapports signal à bruit faibles, mais leur performance asymptotique est mauvaise comparée aux TCs réguliers.

Dans un TC régulier, tous les bits d'information sont répétés un nombre identique de fois égal au nombre de codes constituants, c'est-à-dire égal au degré d'un bit d'information. Pour rendre ce TC irrégulier, il suffit, comme pour les codes LDPC, de modifier le degré de certains bits. Le schéma de principe d'un TC irrégulier est donné par la figure 0.4.

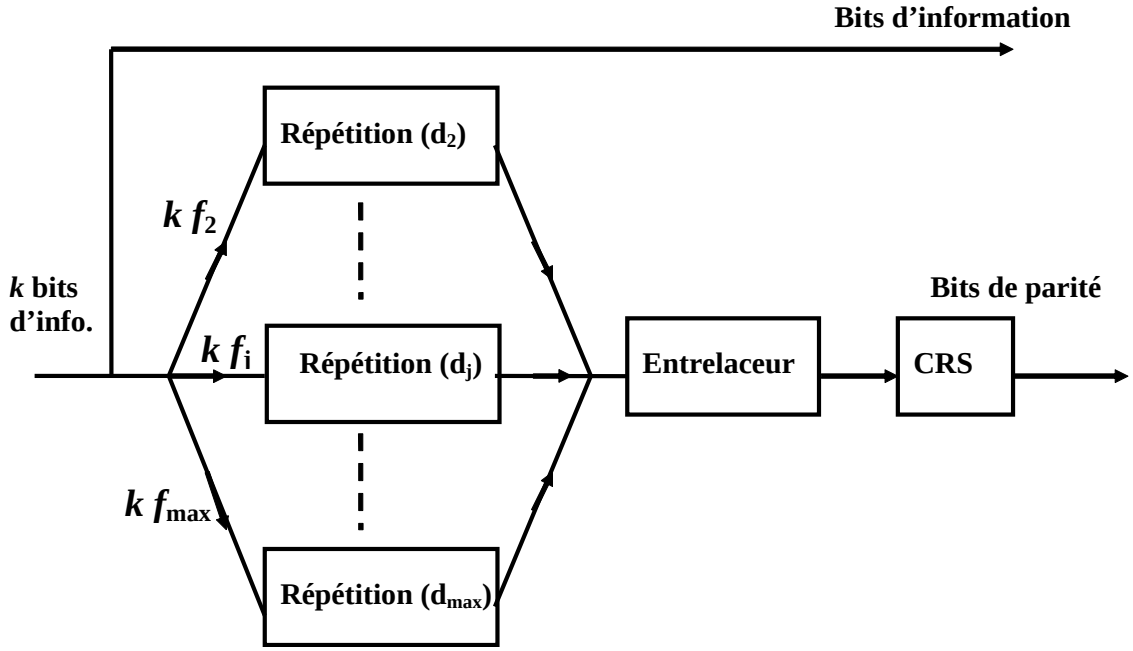


Figure 0.4: Schéma de principe d'un turbocode irrégulier.

Tout d'abord, le profil de degré est sélectionné à l'aide des diagrammes EXIT hiérarchiques. Comme pour les codes LDPC où la méthode de l'évolution de la densité a été utilisée pour optimiser le profil des degrés, l'EXIT représente l'outil habituel permettant de trouver les meilleurs codes élémentaires d'un TC parallèle. Nous avons donc utilisé les diagrammes EXIT pour analyser le profil des degrés.

Ensuite, une étude a été lancée pour la conception d'entrelaceurs adaptés à de tels codes afin d'améliorer leur distance minimale. Ces entrelaceurs, dont la construction est basée sur la notion de graphe en utilisant l'algorithme de Dijkstra [41] et une estimation de la distance du code (par la méthode de l'impulsion d'erreur [55]), montrent une nette amélioration des performances dans la zone du plancher d'erreur. Néanmoins, cette tâche prend beaucoup de temps pour les blocs de grande taille, et nécessite un espace mémoire pour stocker les adresses de l'entrelaceur.

Pour profiter des résultats de la première partie de la thèse et afin de réunir les deux études, une nouvelle structure modifiée de TCs irréguliers a été proposée. Il s'agit de concaténer le post-codeur de rendement 1 à la sortie de ces codes afin de gagner en distance minimale. Cette architecture permet d'avoir des TCs irréguliers plus performants que les TCs réguliers à la fois en termes de seuil de convergence et de performance asymptotique.

Par exemple, la Fig. 0.5 montre un TC irrégulier qui a une distance minimale de 50. Comparé au TC régulier ayant une distance minimale égale à 44, le gain est aussi de 0.2 dB dans la zone de convergence.

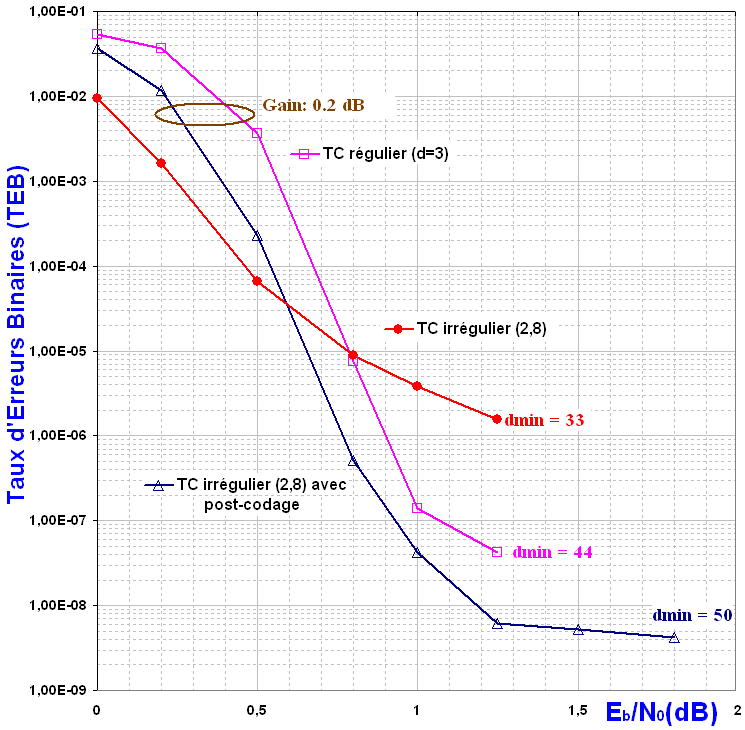


Figure 0.5: Comparaison des performances d'un TC régulier et d'un TC irrégulier sur canal gaussien pour un bloc de $k = 4096$ bits, un rendement $R = 1/4$ et $\lambda = 1/8$. Le degré moyen est 3. Le profil de degré est $\left(f_2 = \frac{5}{6},\ f_8 = \frac{1}{6}\right)$. La taille de l'entrelaceur 3GPP2 est 12282. L'algorithme de décodage est le MAP avec 10 itérations.

Perspectives

Plusieurs perspectives d'étude sont à considérer à partir du travail présenté dans ce rapport de thèse. Par exemple, on peut imaginer de nouvelles structures pour le TC 3D en utilisant un post-codeur 8 états avec treillis variant dans le temps. Cette structure, plus compliquée, est certainement plus puissante. La perte en convergence peut être conséquente. Cependant, ceci ne pose pas problème pour les rendements élevés où on espère gagner en distance avec cette structure nouvelle.

Comme nous avons associé les TCs 3D avec des modulations d'ordre supérieur, on peut s'intéresser également à l'association des TCs 3D avec des techniques de diversité: de la diversité spatiale comme dans les systèmes MIMO ou de la diversité de constellations comme les constellations tournées. Puisque la technique de constellations tournées, comme celles utilisées pour la deuxième génération de TV numérique terrestre, est particulièrement efficace dans des mauvaises conditions de transmission, l'utilisation d'un mapping de Gray spécifique comme expliqué dans le manuscrit peut compenser la perte dans le seuil de convergence et pourrait augmenter d'autant plus la valeur du gain dans le seuil de convergence observé sur les canaux à évanouissements. Je pense que la recherche dans cette direction donnerait des observations optimistes et des résultats intéressants.

D'un autre côté, toute l'analyse des TCs 3D a été consacrée au cas binaire. Il est

possible de l'étendre au cas double binaire (et m-binaire en général). En effet, les TCs double binaires ont des performances meilleures que les TCs classiques pour tout rapport signal-à-bruit. L'application de la méthode d'optimisation ainsi que de la technique du treillis variant temps dans le cas des TCs 3D double binaires semble prometteuse.

Lorsque nous avons associé les TCs irréguliers avec un post-codeur de rendement 1, nous avons considéré des motifs de post-codage réguliers. Cependant, nous avons ici des classes d'irrégularité différentes. On pourra étudier d'autres motifs de post-codage où on favorise les bits pilotes avec un degré $d > 2$, ou bien des motifs de postcodage où seuls les bits répétés $d = 2$ fois sont à prendre en compte.

Pour conclure, la conception de permutations appropriées pour les TCs irréguliers est un point de départ pour des recherches futures importantes. En effet, l'algorithme proposé est très prometteur. Cependant, il est nécessaire de trouver des techniques qui éliminent tôt dans le processus de recherche les entrelaceurs n'ayant pas de bonnes propriétés de distances. De cette façon, l'espace de recherche sera réduit et l'algorithme deviendrait applicable même pour les blocs de grande taille.

Mots Clés: turbocode, turbocode 3D, code 3GPP2, distance minimale, méthode de l'impulsion d'erreur, seuil de convergence, diagramme EXIT, treillis variant dans le temps, modulation, complexité de décodage, turbocode irrégulier, profil de degré, algorithme de Dijkstra, graphe de corrélation.

Contents

List of Figures

List of Tables

Glossary of Acronyms

3D TC	3-Dimensional Turbo Code
3G	Third Generation
3GPP	Third Generation Partnership Project
APP	*A Posteriori* Probability
ARP	Almost Regular Permutation
ARQ	Automatic Repeat reQuest
AWGN	Additive White Gaussian Noise
BCJR	Bahl, Cocke, Jelinek and Raviv
BER	Bit Error Rate
BICM	Bit-Interleaved Coded Modulation
BPSK	Binary Phase-Shift Keying
c-nodes	Check nodes
CCSDS	Consultative Committee for Space Data Systems
CDMA	Code Division Multiple Access
dB	Decibel
DD	Dithered-Diagonal
DRP	Dithered Relative Prime
DVB-RCS	Digital Video Broadcasting with Return Channel Via Satellite
DVB-RCT	Digital Video Broadcasting with Return Channel Terrestrial
EXIT	EXtrinsic Information Transfer
FER	Frame Error Rate

HSR	High-Spread Random
IMI	Input Mutual Information
LDPC	Low-Density Parity-Check
LFR	Linear Feedback Register
LLR	Log Likelihood Ratio
LSB	Least Significant Bit
LTE	Long Term Evolution
MAP	Maximum *A Posteriori*
MHD	Minimum Hamming Distance
MI	Mutual Information
ML	Maximum-Likelihood
MSB	Most Significant Bit
OMI	Output Mutual Information
PEG	Progressive Edge Growth
PTV	Periodic Time Varying
QAM	Quadrature Amplitude Modulation
QPP	Quadratic Permutation Polynomial
QPSK	Quadrature Phase-Shift Keying
RAM	Random-Access Memory
ROM	Read-Only Memory
RP	Relative Prime
RSC	Recursive Systematic Convolutional
RTZ	Return To Zero
SCCC	Serially Concatenated Convolutional Codes
SISO	Soft-Input Soft-Output
SNR	Signal-to-Noise Ratio
SOVA	Soft Output Viterbi Algorithm

TC	Turbo Code
TI	Time Invariant
TL	Truncated Length
UMTS	Universal Mobile Telecommunications System
v-nodes	Variable nodes
W-CDMA	Wideband Code Division Multiple Access

Glossary of symbols

The following mathematical symbols are used in this thesis.

Greek Symbols

α	Parameter used to implement a random variation
δ	Data value
$\hat{\delta}$	Hard decision provided by the decoder
η	Number of pilot groups
θ_i	Fraction of the data to pass through the predecoder
λ	Permeability rate
$\lambda_t(\delta)$	Soft information
$\lambda_t^e(\delta)$	Extrinsic soft information
μ_z	Mean of the extrinsic information
ν	Constraint/Memory length of the code
ξ	Additive error rate at the output of the time-varying predecoder
Π	Turbo code interleaver
Π^{-1}	De-interleaver (inverse interleaver)
Π'	Interleaver used to spread a fraction λ of the parity bits
ρ	Fraction of the surviving bits in y_1 and y_2 after puncturing
σ^2	Variance of the AWGN

Latin Symbols

A_d	Number of codewords with weight d
BER_{in}	Channel error rate
BER_{out}	BER at the output of the predecoder
$BER_{out,TV}$	BER at the output of the time-varying predecoder
$d_{Average}$	Average information bits degree
d_{free}	Free distance of the code
d_{max}	Maximum degree
d_{min}	Minimum distance of the code
D	Delay operator
D^M	Delay of M symbol times
E_b	Energy per information bit
$f(z \setminus x)$	Probability density function
G	Generator matrix
G_A	Asymptotic gain
H	Parity check matrix
$\boldsymbol{I}$	Identity matrix
$I(z,\, x)$	Mutual information
$L_a(d)$	*A priori* LLR
$L_c(x_{noisy})$	Channel LLR
$L_e(d)$	Extrinsic LLR
$L_t(\delta)$	*A posteriori* log-likelihood related to data at time step t
$L_t^e(\delta)$	Extrinsic information related to information symbols
L_t^y	Extrinsic LLR related to redundancy bit y at time step t
$met_t(s', s)$	Branch metrics
$M_t^F(s)$	Forward state metric
$M_t^B(s)$	Backward state metric

$n(d)$	Code multiplicity
N_0	One-sided noise power spectral density
$\Pr(d = i)$	*A priori* probability that the transmitted bit is equal to i
q_x	Number of quantization bits
R	Overall code rate of the code
R_i	Code rate of each constituent encoder RSC_i
$R_{Irregular}$	Code rate of the irregular TC
R_{max}	Highest achievable code rate of the 3D TC
S	Spread factor
S/N	Signal to noise ratio at the receiver
w_d	Bit multiplicity for codewords with Hamming weight d
x	Systematic bits
y_1	Redundancy bits from the first encoder
y_2	Redundancy bits from the second encoder
z	Extrinsic information

Operators

$\approx$	Approximately equal to
$\gg$	Much greater than
$\ll$	Much less than
$erfc(x)$	Complementary error function
max(.)	Returns the value of the biggest element of the argument
min(.)	Returns the value of the smallest element of the argument
$Q(x)$	Gaussian Q function

Introduction

In 1971, the whole community of coding and information theory was in phase with the famous speech of Professor Robert McEliece : *"Too many equations had been generated with too few consequences... Coding theorist professors had begotten more coding theory Ph.D.'s in their own image... no one else cared; it was time to see this perversion for what it was. Give up this fantasy and take up a useful occupation...* ***Coding is dead***." This assertion was contradicted twenty years later by some French engineers who made an incredible claim. They announced to have invented a method of "turbo coding" which could come breathtaking near the Shannon limit. Their simulation curves claimed the incredible performance, far beyond what was considered possible. During the conference, many experts laughed ; others asserted that the simulations were wrong. But professor McEliece said later : "*What blew everyone away about turbo codes is not just that they get so close to Shannon capacity but that they're so easy. How could we have overlooked them ? Berrou and Glavieux didn't know the problem was supposed to be hard, so they managed to find a new way to go about it.*".

The invention of turbo codes (TCs)[18] was a revival for the channel coding research community. Historical turbo codes, also sometimes called Parallel Concatenated Convolutional Codes (PCCCs), are based on a parallel concatenation of two Recursive Systematic Convolutional (RSC) codes separated by an interleaver. They are called "turbo" in reference to the analogy of their decoding principle with the turbo principle of a turbocompressed engine, which reuses the exhaust gas in order to improve efficiency. The turbo decoding principle calls for an iterative algorithm involving two component decoders exchanging information in order to improve the error correction performance with the decoding iterations. This iterative decoding principle was soon applied to other concatenations of codes separated by interleavers, such as Serial Concatenated Convolutional Codes (SCCCs) [11, 12], sometimes called serial turbo codes, or concatenation of block codes, also named block turbo codes [14, 83].

The near-capacity performance of turbo codes and their suitability for practical implementation explain their adoption in various communication standards as early as the late nineties : firstly, they were chosen in the telemetry coding standard by the CCSDS (Consultative Committee for Space Data Systems) [99], and for the medium to high data rate transmissions in the third generation mobile communication 3GPP (Third Generation Partnership Project)/UMTS (Universal Mobile Telecommunications System) standard [4]. They have further been adopted as part of the Digital Video Broadcast - Return Channel via Satellite and Terrestrial (DVB-RCS and DVB-RCT) links [1, 2], thus enabling broad-

band interactive satellite and terrestrial services. More recently, they were also selected for the next generation of 3GPP2/cdma2000 wireless communication systems [3] as well as for the IEEE 802.16 standard (WiMAX) [5] intended for broadband connections over long distances.

While the well-known DVB-RCS/DVB-RCT/WiMAX 8-state double-binary parallel turbo code offers performance very close to the Shannon limit in the so-called waterfall region, it suffers from a flattening effect around 10^{-5} of Frame Error Rate (FER) due to a poor Minimum Hamming Distance (MHD). In future system generations, lower error rates, down to 10^{-8}, will be required to open the way to real time and more demanding applications, such as TV broadcasting or videoconferencing. Therefore, state-of-the-art 8-state turbo codes are no longer suitable for these kinds of applications and more powerful coding schemes are required. At the same time, a reasonable complexity should be preserved. To solve the problem, the aim of the thesis is to explore a new hybrid concatenation structure combining both parallel and serial concatenation based on a 3-dimensional (3D) code, simply derived from the classical turbo code by concatenating a rate-1 post-encoder at its output. The manuscript is divided into four chapters.

The basics of turbo codes are reviewed in chapter 1. The classical structure of a turbo encoder and its different parameters are first presented. In particular, an overview of the interleavers used by turbo codes is discussed. Furthermore, the decoding process as well as the usual algorithms for the iterative data processing are described. In this chapter, the reader can also find a brief presentation of the tools used to determine the convergence and the asymptotic performance of turbo codes. This performance is illustrated in the case of the 3GPP2 turbo codes [3]. Finally, the problem related to the error floor is discussed, and methods to combate this phenomenon are enumerated.

Chapter 2 is an exploration of the 3D TCs properties. The principle of a 3D turbo encoder is presented, and its different parameters are discussed. In particular, the optimization of the interleaving law and the permeability rate is possible. More important, the choice of the post-encoder has to meet several requirements and can be sensibly selected by means of an EXIT analysis. In this chapter, the interest of the 3D TC is assessed through simulations. Union bounds on the minimum distance of 3D binary turbo codes with 3GPP2 interleavers are available. The decoding process is also discussed, and a detailed study of the 3D turbo decoder has been carried out in order to estimate the additional complexity.

In chapter 3, two main axes to improve 3D TCs are deepened. First, an optimization method to increase even more the minimum distance is presented. This method was applied in two particular cases of the 3GPP2 3D turbo code. Furthermore, convergence issues are discussed. In order to reduce the observable loss of convergence, a time varying post-encoder is proposed. At the end, the association of 3D TCs with specific high order modulations is analyzed to improve the performance in the waterfall region. The different stages are illustrated with simulation results, asymptotical bounds and EXIT charts.

Chapter 4 is an investigation of irregular turbo codes. The use of EXIT diagrams allows the search for good degrees profiles to be simplified and this search to be speeded

up compared with Monte Carlo simulations. Besides, the design of suitable permutations based on the Dijkstra's algorithm is detailed, and allows performance in the error floor to be improved. Finally, the association of irregular turbo codes with the same post-encoder used for 3D TCs results in irregular turbo coding schemes which perform better than regular turbo codes at low and high signal to noise ratios at the same time.

The final conclusion summarizes the contributions of this work and discusses perspectives for future research.

Chapter 1

Turbo codes: a breakthrough in digital communications

In 1948, Claude E. Shannon proved that the fundamental limit of digital transmission on channels with white noise is given by the classic channel capacity formula $C = W \log_2 (1 + {}^S/_N)$, where C is the capacity in bit/s, W is the channel bandwidth in Hz and ${}^S/_N$ is the signal to noise ratio at the receiver. Shannon also demonstrated the existence of an error correction system able to achieve this limit.

After more than forty years of extensive research, the concept of turbo coding developed by Claude Berrou and Alain Glavieux [17] finally proved that it was possible to reach the limit of channel capacity with an encoding scheme that could be constructed and decoded in practice. While turbo coding is not the only technique known to be able today to attain the channel capacity limit [32], it is certainly one of the most commonly used channel coding technique for data channels in contemporary mobile communication systems. According to the inventors, the turbo coding principle was born from the experimentation with the feedback concept applied to the error correcting problem using convolutional codes [16]. At the core of a turbo coding system there is a fundamental constitutive element called interleaver. It is a system that changes the positions of input data according to an established position permutation algorithm. Inside the turbo coding process the function of the interleaving block is to help in providing codes vectors with the highest possible level of randomness (ideally, independent vectors) [9] so that the resulting code resembles as close as possible the concept of random coding used by Claude E. Shannon in [96] to prove the channel capacity theorem. Therefore the interleaver is a fundamental element for the performance of a turbo code (TC) [95] and its understanding is a subject of high interest to the specification of physical layers for both wired and wireless transmission technologies.

This chapter is organized as follows : In section 1.1, we present the classical structure of a turbo encoder and its different parameters. In particular, an overview of the interleavers used by turbo codes is discussed. In section 1.2, the decoding process is described as well as some algorithms for the iterative data processing in section 1.3. Tools to determine the convergence and the asymptotic performance of turbo codes can be found in section 1.4 ; whereas section 1.5 illustrates the performance of the turbo codes used by the third-

generation cellular standard cdma2000. Finally, section 1.6 raises some practical issues about how to combate the error floor and section 1.7 draws conclusions.

1.1 Parallel convolutional turbo codes

Turbo codes represent a class of parallel concatenation of two (or more) convolutional codes. Several parameters affect the performance of turbo codes such as component decoding algorithms, number of decoding iterations, generator polynomials and constraint lengths of the component encoders and the interleaver type. Turbo codes, due to their excellent error correcting capability, have been adopted by several standards such as CCSDS (Consultative Committee for Space Data Systems) for space communication, DVB-RCT (Digital Video Broadcasting – Return Channel Terrestrial), DVB-RCS for Satellite communications, 3GPP (Third Generation Partnership Project) communication systems: cdma2000 (Code Division Multiple Access), UMTS (Universal Mobile Telecommunications System), W-cdma (Wideband cdma) for cellular mobile, etc. Table 1.1 provides a detailed list of practical applications of turbo codes.

Application	Turbo code	Polynomials	Rate	Termination	Throughput
CCSDS	binary, 16-state	23, 33, 25, 37	1/6 , 1/4, 1/3 , 1/2	tail bits	1.6 Mbps
3G	binary, 8-state	13, 15, 17	1/4, 1/3 , 1/2	tail bits	2 Mbps
DVB-RCS	double-binary, 8-state	15, 13	1/3 up to 6/7	circular	2 Mbps
DVB-RCT	double-binary, 8-state	15, 13	1/2 , 3/4	circular	2 Mbps
Inmarsat (M4)	binary, 16-state	23, 35	1/2	no	64 kbps
Eutelsat (Skyplex)	double-binary, 8-state	15, 13	4/5 , 6/7	circular	

Table 1.1: Some current known applications of convolutional turbo codes.

1.1.1 Turbo encoder structure

The turbo encoder involves a parallel concatenation of at least two elementary RSC codes separated by an interleaver. Only one of the systematic outputs from the two component encoders is used to form a codeword, as the systematic output from the other component encoder is only a permuted version of the chosen systematic output. Fig. 1.1 shows the block diagram of a turbo encoder. The first encoder outputs the systematic x and recursive convolutional y_1 sequences while the second encoder discards its systematic sequence and only outputs the recursive convolutional y_2 sequence. The first constituent

encoder receives input bits directly, whereas the second constituent encoder is fed with input bits through the interleaver. For each input, three outputs are generated. The total coding rate is 1/3. This rate is however altered depending on possible puncturing of bits and tail bits from the second constituent encoder at termination, as described in subsections 1.1.3 and 1.1.4.

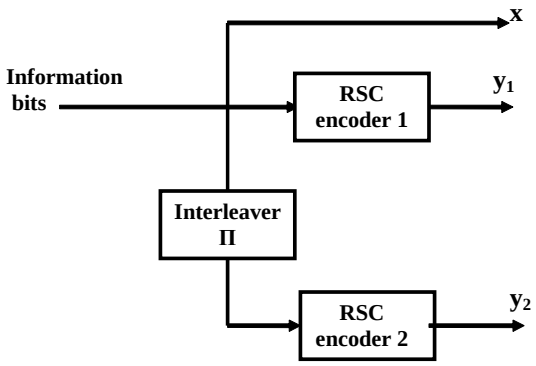


Figure 1.1: Generic block diagram of a turbo encoder.

1.1.2 Design of turbo code interleavers

To avoid losing the whole data when bursts of errors are produced, the interleaver allows spreading these errors in the time. Interleaving is also a key component of turbo codes. Since it affects the distance spectrum [47, 113], this component influences the performance of the overall coding scheme. The interleaver is used to provide randomness to the input sequences but also to increase the weights of the codewords. In fact, input patterns which produce low-weight words in one component code should map through the interleaver to patterns which produce hopefully high-weight words in the other component code. Fig. 1.2 shows an illustrative example.

From Fig. 1.2, the input sequence $\mathbf{x_i}$ produces output sequences $\mathbf{s_i}$ and $\mathbf{p_i}$. Table 1.2 shows how it is possible to increase the weight of the codeword when an interleaver is employed. For example, the sequences $\mathbf{x_1}$ and $\mathbf{x_2}$ are two interleaved input sequences of $\mathbf{x_0}$. They produce codewords with higher weight: 5 and 6 respectively instead of only 3 for the input sequence $\mathbf{x_0}$. In the sequel, we present some representative interleavers commonly used in turbo code design. The approaches used in the literature vary from random interleaving to highly structured interleaving.

The random (or pseudo-random) interleaver simply performs a random permutation of the elements without any restrictions. Fig. 1.3 shows an example where the length of the input sequence is 10. The interleaver writes in [1011001010] and reads out [1111000010].

Usually, a random interleaver yields low weight codewords caused by information weight-2 codewords. Dolinar *et al.* verified that the performance of a turbo coding scheme could be improved if the interleaver succeeds in eliminating input sequences which produce low weight codewords [47]. The S-random (or "spread") interleavers [13, 43, 45]

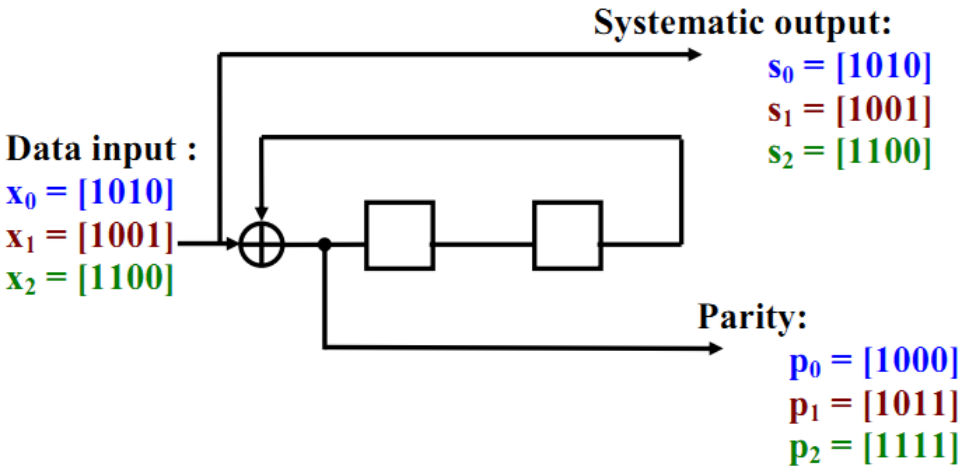


Figure 1.2: An illustrative example of an interleaver's capability.

Input sequence $\mathbf{x_i}$	Output sequence $\mathbf{s_i}$	Output sequence $\mathbf{p_i}$	Codeword weight
$\mathbf{x_0} = 1010$	$\mathbf{s_0} = 1010$	$\mathbf{p_0} = 1000$	3
$\mathbf{x_1} = 1001$	$\mathbf{s_1} = 1001$	$\mathbf{p_1} = 1011$	5
$\mathbf{x_2} = 1100$	$\mathbf{s_2} = 1100$	$\mathbf{p_2} = 1111$	6

Table 1.2: Input and output sequences for encoder in Figure 1.2.

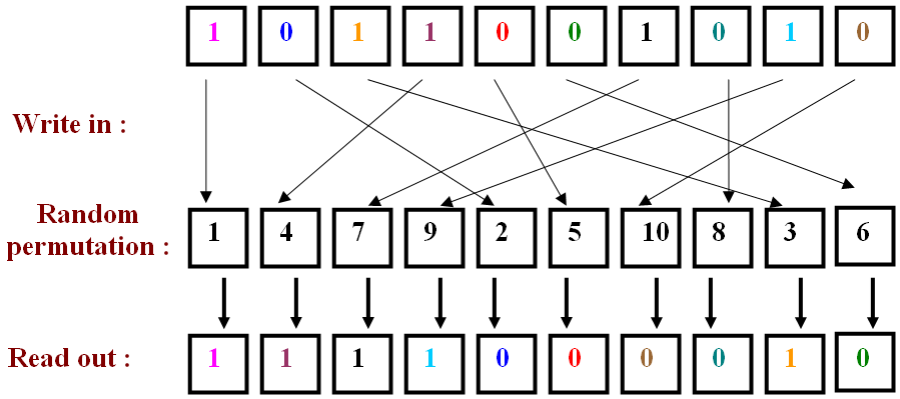


Figure 1.3: Example of random interleaver.

spread low weight input patterns to generate higher weight codewords. An extension to the S-random interleaver, has been proposed by Crozier [34]. If Π is the interleaving law, the spread factor S is applied for any two positions i and j according to the relation:

$$|\Pi(i) - \Pi(j)| + |i - j| > S$$

The above relation states that the closer the bits i and j are before interleaving the more they should be spread apart after interleaving. It can be shown that the theoretical maximum spread is: $S_{max} = \text{floor}\left(\sqrt{2k}\right)$ [24], where k is the length of the interleaver. As

an example, for a block length of $k = 20730$, which is the longest interleaver length for the cdma 2000 turbo code, the theoretical maximum spread is 203 (*i.e.* $S \leq 203$). A number of variations on the spread interleaver are presented in [8, 59, 81, 91].

Interleavers which are pseudorandom with constraints on spreading properties have been shown to provide good performance. But such "randomlike" interleavers may be hard to implement in an efficient manner especially for long data blocks, as all the addresses need to be stored in a memory. Another novel interleaver design based on the correlation between the extrinsic information is reported in [60]. Simulation results show that the correlation designed interleavers perform approximately 0.1 dB better than S-random interleavers.

In the literature, we can also find many structured interleaver approaches that have high-spread properties, such as dithered golden interleavers [39], dithered-relative prime (DRP) interleavers [36], and dithered-diagonal (DD) interleavers [34]. These methods are easy to implement and can be used to design interleavers with excellent spreading and good error performance. For instance, it was shown that for a block size of $k = 512$ data bits and an unpunctured code rate of 1/3, the flares in the packet error rate and bit error rate (BER)[1] curves can be kept below 10^{-8} and 10^{-10}, respectively [34].

Other deterministic interleaver design methods were suggested, such as the almost regular permutation (ARP) [22] and the quadratic permutation polynomial (QPP) [97] that has been adopted as an emerged solution to the requirements of the 3GPP Long Term Evolution (LTE) [78]. The previous list is not exhaustive. However, it shows that devising more sophistical internal permutations to lower the error floor is not an easy task.

1.1.3 Trellis termination

Convolutional encoders are *a priori* specified for continuous flows of information, which corresponds to an infinite sequence of information bits. If the information is formated by blocks, it is necessary to plan a suitable ending of the encoding process. When the convolutional encoder has no trellis termination, the final state is not known in advance and the corresponding decoding process finds it hard to correct the errors at the extremities of the block. To ensure that the last bits to be encoded are as well protected as the first ones, we add to the block of information some bits which purge the register at the end of coding and allow the encoder to retrieve a known state (the all-zero state). These bits are called tail bits.

For the turbo codes many solutions are possible: no trellis termination for each encoder, only the first encoder is terminated or both encoders are terminated. Another termination technique is to ensure the trellis start and end states are identical. This technique, referred to as tail-biting [112], has the main advantage of not requiring any extra bits to be transmitted. Tail-biting is used in several popular communications standards, such as the Wimax (IEEE 802.16), the DVB-RCS and DVB-RCT standards.

[1] The BER is simply the ratio of incorrect data bits divided by the total number of data bits transmitted.

1.1.4 Puncturing

The purpose of puncturing is to increase the overall code rate. This process, which should be fairly distributed between both encoders, consists in removing certain bits from the codeword. Puncturing must be as regular as possible and it maximizes the free distance of the code. Details about how to select the puncturing patterns that improve turbo code performance are available in [13, 40].

1.2 Turbo decoding

Two important innovations allowed the turbo codes to reach an excellent performance in the history of channel coding. The first one, as explained above in section 1.1, is the parallel concatenation of two RSC codes separated by an interleaver. The second being the process of Soft Input Soft Output (SISO) iterative decoding. The block diagram of the turbo decoder is shown in Fig. 1.4. Turbo codes get their name because the decoder uses feedback, like a turbo engine.

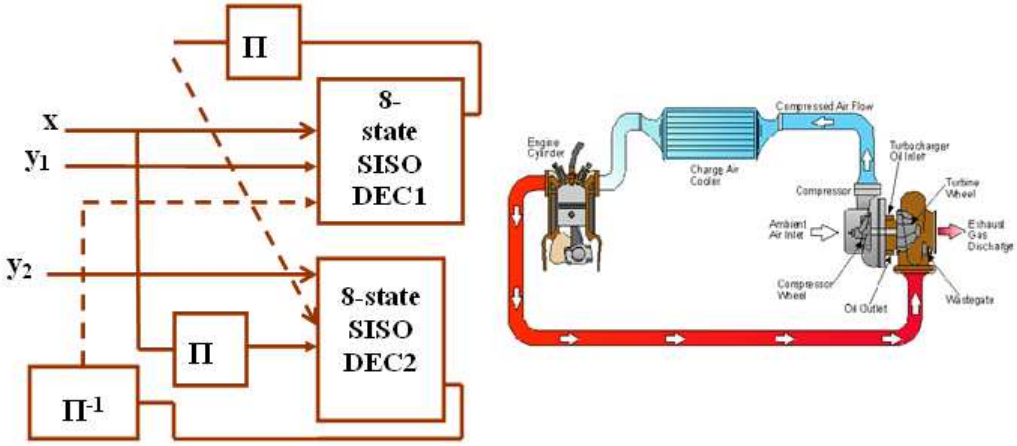


Figure 1.4: A turbo decoder and a mechanical turbo engine. Source of the turbo engine image : `http://www.modpark.com/resimler/2-resim/turbo1cc.jpg`.

There are two elementary decoders separated by the same interleaver. They correspond to the two RSC encoders and they exchange information along the iterations. In fact, the feedback allows each decoder to take advantage of all the information available. Therefore, a SISO decoder is necessary as it processes soft decisions at its unput and tries to make them more reliable, thanks to local redundancy (*i.e.*, y_1 or y_2). The information coming from the channel and referred to as *intrinsic information*, is used by the two decoders. An *extrinsic information* is produced by each decoder and transmitted to the other one. A good permutation has to guarantee a good exchange of the extrinsic information between the decoders.

The iterative decoding consists in decoding alternately both elementary codes and passing the information between the corresponding decoders. The inputs to the first decoder are the observed systematic bits, the parity bit stream from the first encoder and

the deinterleaved extrinsic information from the second decoder. The inputs to the second decoder are the interleaved systematic bit stream, the observed parity bit stream from the second RSC code and the interleaved extrinsic information from the first decoder. The first SISO decoder generates the soft output and subsequently an extrinsic information. The extrinsic information is interleaved and used by the second SISO decoder as the estimate of the *a priori* probability (APP). The second SISO decoder also produces the extrinsic information and passes it after deinterleaving to the first SISO decoder to be used during the subsequent decoding operation. To benefit at best from the information produced by each decoder, it was established that the exchange of soft decisions rather than hard decisions can lead to an excellent performance.

The mathematical foundations of the decoding process with soft decisions are based on the theorem of Bayes. Let us consider a non coded transmission using a binary phase-shift keying (BPSK) modulation over an Additive White Gaussian Noise (AWGN) channel. An information bit $d = 0$ is transmitted as $x = -1$ and an information bit $d = 1$ is transmitted as $x = +1$. The observation at the output of the noisy channel is given by : $x_{noisy} = x + n$, where n represents a sample of noise having a Gaussian distribution with standard deviation σ. Let $\Pr(d = 0)$ (resp. $\Pr(d = 1)$) the *a priori* probability that the transmitted bit is equal to 0 (resp. 1). The *a posteriori* probabilities or likelihoods are calculated using the the theorem of Bayes and are expressed in the following way :

$$P_0 = \Pr(d = 0 \,|\, x_{noisy}) = \frac{\Pr(x_{noisy} \,|\, d{=}0)\,\Pr(d{=}0)}{\Pr(x_{noisy})}$$

$$P_1 = \Pr(d = 1 \,|\, x_{noisy}) = \frac{\Pr(x_{noisy} \,|\, d{=}1)\,\Pr(d{=}1)}{\Pr(x_{noisy})}$$

These likelihoods can be considered as a refinement of the *a priori* knowledge on the value of the transmitted digit supplied by the observation of the channel. Then the optimal hard decision $\hat{d}$ in the sense of the maximum *a posteriori* (MAP) is :

$$\hat{d} = \begin{cases} 0 \text{ if } P_0 > P_1 \\ 1 \text{ otherwise.} \end{cases}$$

The soft decision is defined as the Logarithmic Likelihood Ratio (LLR) :

$$LLR = L(d \,|\, x_{noisy}) = \ln \frac{P_1}{P_0} = L_c(x_{noisy}) + L_a(d)$$

where $L_c(x_{noisy}) = \ln \frac{\Pr(x_{noisy} \,|\, d=1)}{\Pr(x_{noisy} \,|\, d=0)}$ is the channel LLR and $L_a(d) = \ln \frac{\Pr(d=1)}{\Pr(d=0)}$ is the *a priori* LLR. The criterion of decision can then be written :

$$\hat{d} = \begin{cases} 0 \text{ if } LLR > 0 \\ 1 \text{ otherwise.} \end{cases}$$

It follows that the sign of the soft decision determines the hard decision and that the absolute value of the soft decision determines the reliability of this decision. In particular, if the noise is Gaussian, the channel LLR has for expression :

$$L_c(x_{noisy}) = \ln \frac{\frac{1}{\sigma\sqrt{2\Pi}} \exp\left(-\frac{\left(x_{noisy}-1\right)^2}{2\sigma^2}\right)}{\frac{1}{\sigma\sqrt{2\Pi}} \exp\left(-\frac{\left(x_{noisy}+1\right)^2}{2\sigma^2}\right)} = \frac{2}{\sigma^2} x_{noisy}$$

Until now we considered a system of non-coded transmission. For a system of coded transmission, if the *a priori* information is distributed in an independent way, the soft output of a decoder can be expressed in the following way [18] :

$$L(d) = L_c(x_{noisy}) + L_a(d) + L_e(d)$$

where $L_e(d)$ is the extrinsic LLR representing the knowledge acquired thanks to the decoding process.

1.3 Algorithms for iterative (turbo) data processing

Some of the major SISO decoding approaches, developed for turbo decoding are: Maxium Aposteriori Probability (MAP), Log-MAP, Max-Log-MAP and Soft Output Viterbi Algorithm (SOVA).

1.3.1 BCJR or MAP algorithm

Bahl, Cocke, Jelinek and Raviv presented an optimal algorithm [7], often referred to as the BCJR algorithm, for estimating the *a posteriori* probabilities of states and state transitions of a Markov source observed through a discrete memoryless channel. This algorithm was later adapted by Berrou *et al.* to produce *a posteriori* probabilities in the case of the iterative decoding of convolutional codes. The BCJR algorithm is often referred to as the maximum *a posteriori* (MAP) algorithm. With regard to the Viterbi algorithm which determines the most probable information sequence [50, 108, 109], the MAP algorithm associates with every decoded bit an estimation of the reliability of this decision. This algorithm is used in general for the iterative decoding of convolutional codes. To simplify the description of the MAP algorithm, we consider a RSC encoder at coding rate $R = 1/2$ with memory length M (2^M internal states). Let u_k be the k^{th} information bit and c_k the corresponding parity bit. At the output of the channel, we receive the following sequence : $R_1^N = (R_1, ..., R_k, ..., R_N)$ where $R_k = (x_k, y_k)$, x_k and y_k are the observations received at time k corresponding respectively to the information bit u_k and the redundancy bit c_k.

The MAP decoder computes the ratio of conditional probabilities :

$$\Lambda(u_k) = \frac{\Pr\left(u_k=1 \backslash R_1^N\right)}{\Pr\left(u_k=0 \backslash R_1^N\right)}$$

To compute $\Lambda(u_k)$, we introduce the densities of conditional probabilities for the state m :

$$\text{Forward } \alpha_k(m) = \Pr\left(S_k = m \setminus R_1^N\right)$$

and

$$\text{Backward } \beta_k(m) = \Pr\left(S_k = m \setminus R_k^N\right).$$

We can show that :

$$\Lambda(u_k) = \frac{m\alpha_{k-1}(m)\gamma_k^1(m)\beta_k(f(1,m))}{m\alpha_{k-1}(m)\gamma_k^0(m)\beta_k(f(0,m))}$$

$\gamma_k^i(m)$ is the branch metric at time k corresponding to the transition from state m to state $f(i,m)$ when the information bit is $u_k = i$.

$\alpha_k(m)$ et $\beta_k(m)$ are calculated iteratively.

$$\alpha_k(m) = \sum_i \alpha_{k-1}(b(i,m))\gamma_k^i(b(i,m))$$

Where $\alpha_0(0) = 1$ and $\alpha_0(m) = 0$ for $m \neq 0$, if the initial state is 0. In fact, each alpha metric is the sum of the previous alphas multiplied by the branch metrics along each branch from all the previous states to the current state m. The computation of the beta metrics is similar to that of the alphas but starting at the end of the trellis and going in the reverse direction :

$$\beta_k(m) = \sum_i \beta_{k+1}(f(i,m))\gamma_k^i(m)$$

Where $\beta_N(0) = 1$ and $\beta_N(m) = 0$ for $m \neq 0$, if the final state is 0.

$f(i,m)$ is the final state corresponding to a starting state m when the information bit is $u_k = i$.

$b(i,m)$ is the starting state corresponding to a final state m when the information bit is $u_k = i$.

For more clarifications about these notations, a trellis representation is given in Fig. 1.5.

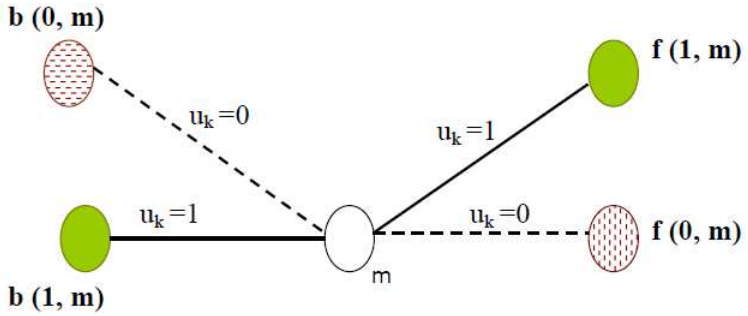


Figure 1.5: Calculation of the branch metrics.

Contrary to the Viterbi algorithm which concentrates on the most likely message, the MAP algorithm is one of the methods acting at the level of the information bits of the message. The advantages supplied by this algorithm are the minimization of the probability of error of every bit of the message and the estimation of the reliability of the decoding. The MAP algorithm is thus the optimal solution to estimate the state probabilities and the memory transitions of a convolutional encoder. In other words, the MAP algorithm is the optimal component decoder algorithm. However, it is complex for a hardware implementation of the decoder. In fact, the algorithm suffers from numerical problems because of the exponent in probability calculations and from a need to use a very high precision in the decoder to represent small vector elements.

1.3.2 LogMAP algorithm

The LogMAP algorithm, which uses the notion of *Jacobian Algorithm*, is a transformation of MAP into the logarithmic domain. In fact, multiplication is converted to addition and addition is converted to max*(.) operation :

$$\max *(x, y) = \log\left(e^{x} + e^{y}\right) = \max\left(x, y\right) + \log\left(1 + e^{-|x-y|}\right)$$

The numerical problems that occur in MAP are thus circumvented.

1.3.3 Max-Log-MAP algorithm

This algorithm is derived from the LogMAP algorithm, where the max*(.) operation is replaced by simply the maximum. In fact, only the path with maximum value is considered at each state in forward or backward recursions. Therefore, the Max-Log-MAP algorithm is simpler [58, 89] but performs worse than the MAP or the LogMAP algorithms. Compared to the LogMAP algorithm, a degradation of 0.5 dB in the error rate performance was observed in [89].

The extrinsic information exchanged between the constituent decoders can be scaled to improve the performance of turbo decoding for sub-optimal algorithms [31, 110]. Scaling factor modification has also been applied and tested on the Max-Log-Map algorithm. Authors in [110] used a constant scaling factor of 0.7 and reported a gain from 0.2 to 0.4 dB in the case of the 3GPP standard. In [31], the best scaling factors for each iteration were calculated for different signal to noise ratios (SNRs)[2] by off-line computation. It was shown that a turbo decoder using the modified Max-Log-Map performs within 0.05 dB of a turbo decoder using the Log-Map algorithm [31]. The performance improvement introduced by the scaling factor modification is explained as the correction of the accumulated bias due to maximum operation in the Max-Log-Map algorithm [31].

[2]The SNR is computed by dividing the energy per received data bit E_b by the single-sided noise spectral density N_0 of the channel.

1.4 Convergence and asymptotic performance

Two essential parameters allow estimating the performance of a concatenated error-correcting code and its decoder:

- ◇ The convergence threshold defined as the minimum signal to noise ratio where the performance of a coded system becomes better compared with the non coded transmission system. A good convergence corresponds to a low convergence threshold, because the performance of the system at high and average levels of noise are close to the theoretical limit.

- ◇ The asymptotic gain G_A indicates the maximum gap between the coded and non-coded error rate curves. When G_A is reached, the error rate curve with coding becomes parallel to the curve without coding. G_A can be expressed in the following way:

$$G_A \approx 10 \log (R\, d_{min})$$

  where R is the coding rate and d_{min} the Minimum Hamming Distance (MHD).

In fact, there is a compromise between good convergence and MHD. For average or high error rates, it is better to privilege the convergence threshold to the detriment of the minimum distance of the code; whereas at low error rates it is better to have a high minimum distance.

1.4.1 Graphical analysis of the convergence using the EXIT diagrams

To evaluate the performance of a coding scheme with iterative decoding, Monte Carlo simulations of the bit error rate can be carried out. However this means may require a lot of computing time and resources. The study of the evolution of the extrinsic information is rather an attractive solution for finer comparisons between different coding schemes. It allows to analyze the performance of the coding scheme, but also to supply an estimation of the bit error rate. This method was proposed at first by Stefan Ten Brink in [103]. Berrou *et al.* showed in [18] that when the decoding process converges, the extrinsic information can be modelled by a Gaussian distribution where the mean and the variance increase as the number of iterations increases. This modelling allows the extrinsic information z to be characterized only by its mean μ_z and its variance σ^2. The Gaussian approximation is verified up to a certain number of iterations, this number increases for the low values of E_b/N_0. The Mutual Information (MI) $I(z, x)$, as defined below, is used in EXtrinsic Information Transfer (EXIT) charts to predict the convergence behavior of iterative decoding [103, 106, 107]. It measures the quantity of information provided on average by the extrinsic information z on the information bits x.

Let $z \sim \mathcal{N}(\pm\mu_z; {\sigma_z}^2)$ be the *a priori* LLR of a SISO decoder. The associated MI is given by :

$$I(z,\, x) = \frac{1}{2} \sum_{x=\pm 1} \int_{\mathbb{R}} f(z \setminus x) \log_2\left(\frac{2f(z \setminus x)}{f(z \setminus x = +1) + f(z \setminus x = -1)}\right) dz$$

If the channel is symmetric (*i.e.* $f(z \setminus x = -1) = f(-z \setminus x = +1)$) and the probability density function is consistent (*i.e.,* $f(z \setminus x) = f(-z \setminus x)\, e^{-z}$), then :

$$I(z,\, x) = \int_{\mathbb{R}} f(z \setminus x = +1)\, \log_2\left(\frac{2f(z \setminus x = +1)}{f(z \setminus x = +1) + f(z \setminus x = -1)}\right) dz$$

$$I(z,\, x) = \int_{\mathbb{R}} f(z \setminus x = +1)\, \log_2\left(\frac{2f(z \setminus x = +1)}{f(z \setminus x = +1)(1 + e^{-z})}\right) dz$$

$$I(z,\, x) = 1 - \int_{\mathbb{R}} f(z \setminus x = +1)\, \log_2(1 + e^{-z})\, dz$$

$$I(z,\, x) = 1 - \frac{1}{\sqrt{4\Pi \mu_z}} \int_{\mathbb{R}} \log_2(1 + e^{-z}) \exp\left(-\frac{(z - \mu_z x)^2}{4\mu_z}\right) dz = J(\mu_z) \tag{1.1}$$

The hypothesis of the exponential symmetry imposes $\sigma_z^2 = 2\mu_z$. Then, the MI $I(z,\, x)$, can be expressed as follows :

$$I(z,\, x) = 1 - \frac{1}{\sqrt{2\Pi}\sigma_z} \int_{\mathbb{R}} \log_2(1 + e^{-z}) \exp\left(-\frac{(z - \frac{{\sigma_z}^2}{2})^2}{2{\sigma_z}^2}\right) dz = J(\sigma_z)$$

We obtain the expression of $J(\sigma_z)$. This function of equation (1.1) can also be expressed in the following way :

$$J(\sigma_z) = 1 - \mathbb{E}_z(\log_2(1 + e^{-z}))\ ,\ z \sim \mathcal{N}(\mu_z, 2\mu_z)$$

$J(\sigma_z)$ is monotonically increasing in σ and therefore it has a unique inverse function, $\sigma = J^{-1}(I)$. The J function and its inverse are important functions that are used extensively in the following chapters to compute different convergence thresholds. Unfortunately, they cannot be expressed in closed form, but they can be closely approximated by :

$$J(\sigma_z) \approx \left(1 - 2^{-H_1 {\sigma^2}^{H_2}}\right)^{H_3} \tag{1.2}$$

$$J^{-1}(I) \approx \left(-\frac{1}{H_1} \log_2\left(1 - I^{\frac{1}{H_3}}\right)\right)^{\frac{1}{H_2}}$$

Numerical optimization to minimize the total squared difference between equation (1.1) and equation (1.2) results in the parameter values $H_1 = 0.3073$, $H_2 = 0.8935$, and $H_3 = 1.1064$ [27]. The curve in Fig. 1.6 shows equation (1.1) and the indistinguishable approximation in equation (1.2).

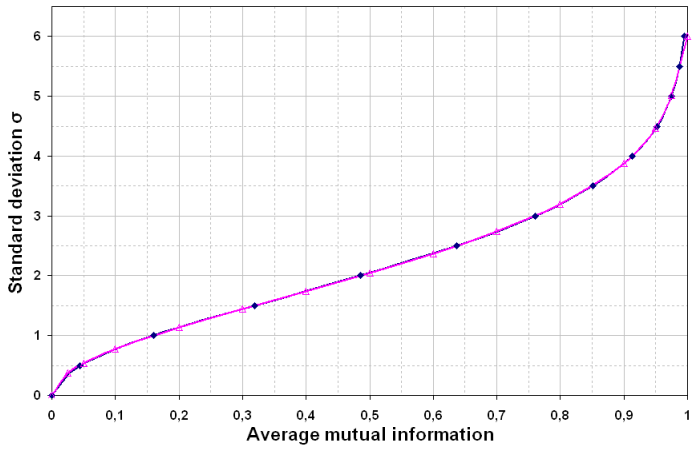


Figure 1.6: The J function and its approximation where $H_1 = 0.3073$, $H_2 = 0.8935$ and $H_3 = 1.1064$.

1.4.2 EXIT charts for turbo codes

To generate the EXIT chart of a TC, we have to consider the transfer characteristics of the extrinsic information for each SISO decoder. Fig. 1.7 describes how the mutual information is measured in practice. After encoding, the information and parity bits x and y are transmitted over a noisy channel. The decoder receives the transmitted values at the channel output. Information bits x are also transmitted as *a priori* knowledge, as if they were coming from the other decoder (lower input line to the decoder in Fig. 1.7).

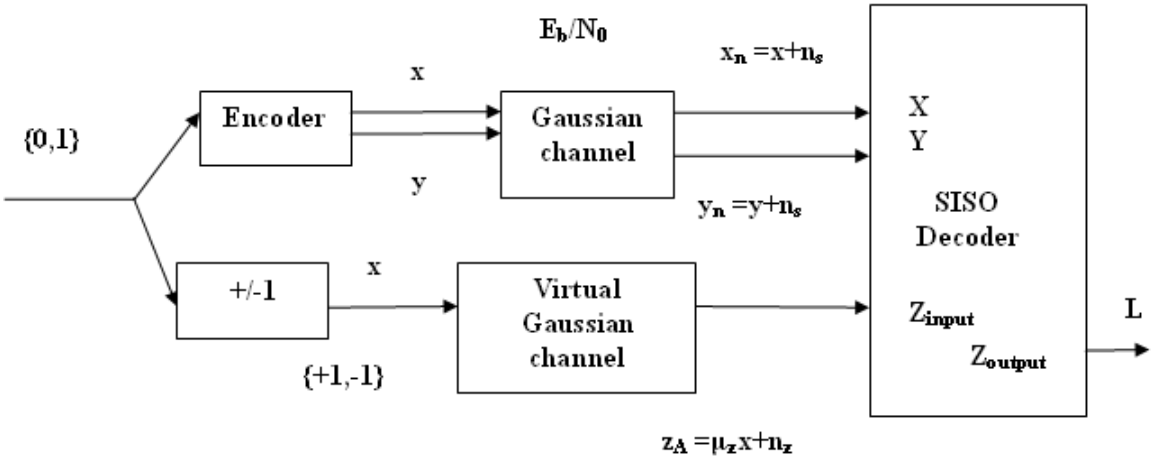


Figure 1.7: Measurement of mutual information at the component decoder level.

The mutual information between the bits x and the soft output values, L, of these bits after decoding can be computed from a large number N of samples [57]:

$$I(L,x) = 1 - \mathbb{E}\left(1 + \log_2\left(1 + \exp\left(-L\right)\right)\right) \approx \frac{1}{N}\sum_{k=1}^{N} \log_2\left(1 + \exp\left(-x_k\, L_k\right)\right)$$

Fig. 1.8 shows that the curves have intersection points different from the point (1,1) when the signal to noise ratio is low. Then, the iterative process starting with a zero average mutual information in entry cannot end in a perfect determination of the message. In the case of high enough signal to noise ratio, the two curves do not have any intersection outside the point of coordinates (1,1). In Fig. 1.8, a *tunnel* between the two curves is observed, meaning that convergence is possible at this SNR. The convergence threshold of the turbo code is the minimum signal to noise ratio where the only intersection point is (1,1).

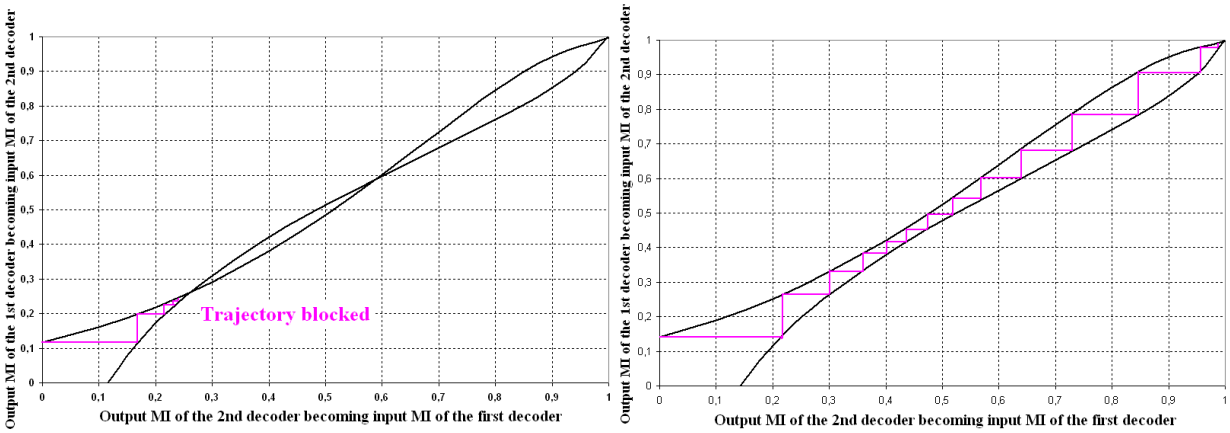


Figure 1.8: Turbo code EXIT charts and trajectory for different values of E_b/N_0.

1.4.3 Distance measurement methods for turbo codes

Consider the transmission of a linear binary turbo code over the AWGN channel using BPSK or quadrature phase-shift keying (QPSK) modulation. Applying maximum-likelihood (ML) decoding, the frame error rate and bit error rate are upper bounded by the union bounds [111] :

$$FER \leq \frac{1}{2} \sum_{d \geq d_{min}} A_d \, erfc\left(\sqrt{d_{min} \frac{k}{n} \frac{E_b}{N_0}}\right) \tag{1.3}$$

$$BER \leq \frac{1}{2} \sum_{d \geq d_{min}} \frac{w_d}{k} \, erfc\left(\sqrt{d_{min} \frac{k}{n} \frac{E_b}{N_0}}\right) \tag{1.4}$$

where :

- n is the codeword length in bits,
- k is the number of information bits,
- d_{min} is the minimum distance of the code,
- the multiplicity A_d is the number of codewords with weight d,

- ◇ the information bit multiplicity w_d is the sum of the Hamming weights of the A_d input sequences generating the codewords with Hamming weight d,
- ◇ the complementary error function erfc(x) is given by the expression $\frac{2}{\sqrt{\Pi}} \int_x^\infty e^{-t^2} dt$,
- ◇ E_b is the energy per information bit,
- ◇ and N_0 is the one-sided noise power spectral density.

The function $\text{erfc}(\sqrt{x})$ decreases exponentially with x. Thus, the first term or first few terms of equation (1.3) and equation (1.4) can be used to approximate FER and BER at high SNRs (*i.e.* at very low error rates). It is difficult to determine by Monte Carlo simulations the performance of a code that operates at very low error rates. Berrou *et al.* [23] introduced the innovative error impulse method, based on iterative decoding. The approach of Garello *et al.* in [55] represents a modification and an improvement of the error impulse error. In this sub-section, we discuss briefly Garello's distance measurement method called the all-zero iterative decoding algorithm. This technique is based on the transmission of an all-zero sequence corrupted by an error impulse and a noise. The difference with the method in [23] is that the amplitude of the error impulse is high enough to prevent the decoder from converging to the all-zero codeword. On the other hand, a Gaussian noise is added to the sequence in the entry of the decoder, which helps the decoder to converge towards non-zero codewords. It is necessary to adjust the level of noise to make the decoder converge to low weight codewords and thus to obtain a good estimation of the minimum distance, because the weight of the codeword on which the decoder converged represents a superior limit of the minimum distance. This algorithm works very well for interleavers for small and average distances, typically up to 50, and it is the algorithm we used in our study. The algorithm is the following :

1. Choose a value for $E_b/N_0 \in [2\,\text{dB}, 8\,\text{dB}]$ and compute the noise variance $\sigma^2 = \frac{N_0}{2} = \frac{1}{2\frac{k}{n}\frac{E_b}{N_0}}$.
2. Fix $d_{min} = 1000$.
3. For $i = 0$ to n /* i refers to the position of the impulse*/
   a) Let us denote the transmitted vector by e. Put $e_j = -1$ for all $j \neq i$.
   b) Place an error impluse in e_i. It consists of a positive real number with high amplitude.
   c) Add a white Gaussian noise with variance σ^2.
   d) Decode iteratively the received sequence.
   e) When the decoder converges to a valid codeword $\hat{c}_i$, compute its Hamming weight $w(\hat{c}_i)$.
      - ◇ If $w(\hat{c}_i) < d_{min}$ and $w(\hat{c}_i) > 0$, a new codeword is found. Set $d_{min} = w(\hat{c}_i)$, update the multiplicity and store $\hat{c}_i$.

- ◇ If $w(\hat{c}_i) \geq d_{min}$ and the codeword has not been detected yet ⇒ store it and set its multiplicity to 1.
- ◇ If $w(\hat{c}_i) \geq d_{min}$ and the codeword has already been detected ⇒ increment the corresponding multiplicity.

f) Increment i.

4. When the decoder does not converge to codewords:

a) Go to step 1.

b) If $E_b/N_0 < 8\,\text{dB}$, increment E_b/N_0 by 0.5 dB and try another value of σ^2.

An improvement of the GARELLO's method was proposed by CROZIER *et al.*, at the expense of higher computation time since they introduced a second and a third error impulse [37, 38].

1.5 Cdma2000 turbo code

One of the major third-generation cellular standard is cdma2000. It is one of the two most widely adopted third-generation cellular standards, the other being UMTS, and it is standardized by the Third Generation Partnership Project 2 (3GPP2) [3]. The size of the interleaver for the cdma2000 turbo code must be one of the following specific values: 378, 570, 762, 1146, 1530, 2298, 3066, 4602, 6138, 9210, 12282, or 20730 bits. The constituent RSC encoder used by the cdma2000 turbo code is shown in Fig. 1.9. As can be seen, this encoder has three output bits, one systematic and two parities, for each input bit. Thus, the code rate of this RSC encoder is $R = 1/3$, neglecting the tail bits.

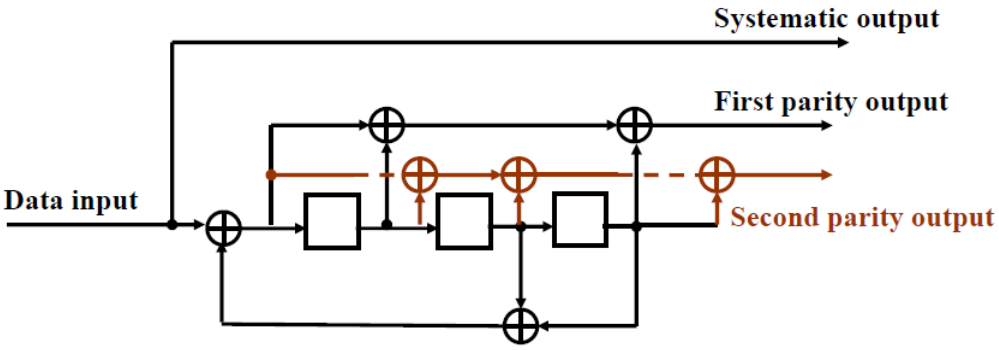


Figure 1.9: The rate-1/3 RSC encoder used by the cdma2000 turbo code.

The overall code rate for the cdma2000 turbo code is $R = 1/5$. Through puncturing, rates of $R = 1/2$, $1/3$, and $1/4$ can be achieved. For instance, to achieve rate $R = 1/3$, the encoder deletes the second parity output of each encoder and only the first parity outputs are transmitted. The puncturing mechanisms used to achieve rates $R = 1/2$ and $R = 1/4$ are slightly more complicated, but the details can be found in the specification [3]. For

reasons of simplicity in our simulations, we considered only the first parity for coding rates higher than $R = 1/3$. Table 1.3 gives some puncturing patterns adopted within the framework of the study of 3GPP2 turbo codes.

Code Rate	1/2	2/3	3/4	4/5
Puncturing Period	2	4	6	8
x	11	1111	111111	11111111
y_1	10	1000	100000	10000000
y_2	01	0010 or 1000	000100 or 100000	00001000 or 10000000

Table 1.3: Puncturing patterns for the data bits.

1.5.1 Cdma2000 interleaving

The description of the cdma2000 turbo interleaver is detailed in Appendix A. Also, an example is presented to illustrate the complete interleaving process defined by the standard. Fig. 1.10 shows the results plotted in a Cartesian plane. The x-axis represents the index and the y-axis the position of the output data.

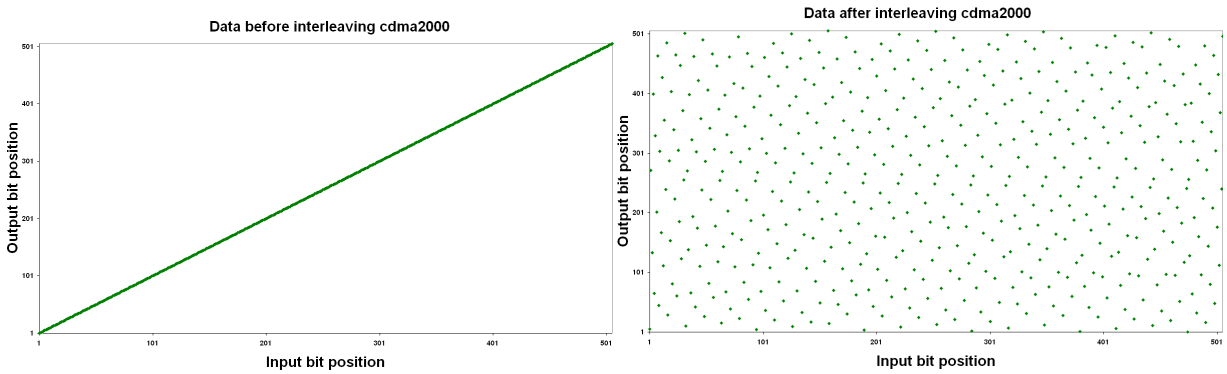


Figure 1.10: Input vector versus output vector before and after interleaving for blocks of 506 bits.

Although the output produced by the interleaver seems to have elements of randomness, it should be noticed that the transformation between input and output positions is deterministic.

1.5.2 Performance of cdma2000 turbo code

This section illustrates the performance of cdma2000 turbo code. Simulations were run to determine the performance of these turbo codes in AWGN with BPSK modulation. In each case, either the MAP or Max-Log-MAP algorithm was used. For each simulation, a curve showing the bit-error rate versus the per-bit signal to noise ratio was computed. Fig. 1.11 shows the performance of the 3GPP2 turbo code with an input frame size of

$k = 6138$ bits. This figure shows how performance improves as the number of decoder iterations increases. After one iteration, performance is quite poor, and the decoder is unable to achieve a BER lower than 10^{-1} even for high SNRs. However, as the decoder iterates, performance improves until at the tenth iteration it can achieve a BER of 10^{-6} at an SNR of only 1.25 dB. Note how each subsequent iteration improves performance, but that this improvement follows a law of diminishing returns. Thus, although an eleventh (or higher) iteration would provide slightly improved performance, the extra complexity and decoding delay is not justified.

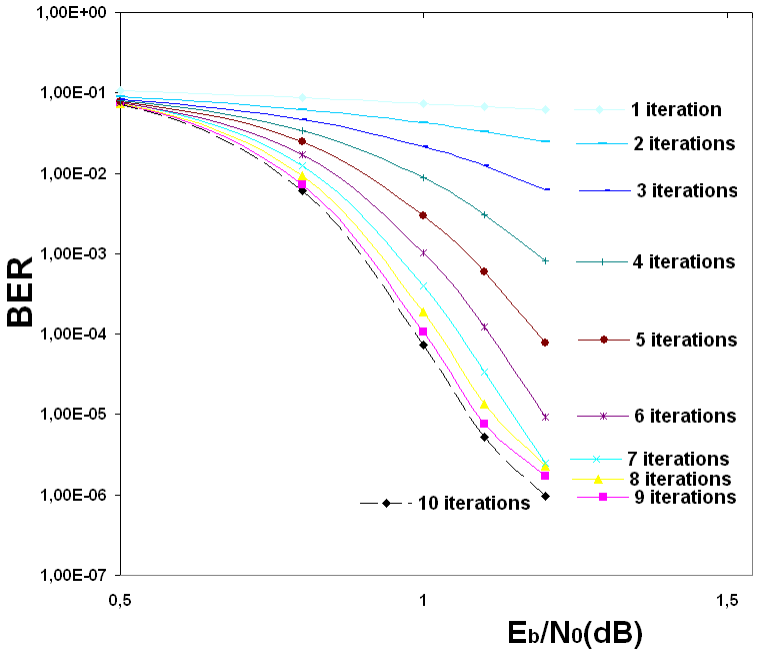


Figure 1.11: Bit-error performance of the 3GPP2 turbo code as the number of decoder iterations varies from 1 to 10. The encoder input word length is $k = 6138$ bits, code rate is $1/2$, modulation is BPSK, and channel is AWGN. The simulations use the MAP algorithm.

Fig. 1.12 shows the performance of the 3GPP2 turbo code as a function of input frame size k. As can be seen in this figure, the performance improves with increasing k. This is due to an increase in interleaver gain as the input frame size gets larger.

Table 1.4 lists the minimum E_b/N_0 required to achieve a BER of 10^{-3}, a target error rate in the case of 3GPP communication systems, for each of the four frame sizes shown in Fig. 1.12. While the BER curve falls off sharply with increasing SNR for moderate error rates (e.g., BER $> 10^{-5}$), the BER curve begins to flatten at higher SNRs. This characteristic can be observed in Fig. 1.12, where the BER was simulated down to very small values. It does not represent a problem since the 3GPP2 turbo code is used in Automatic ReQuest (ARQ) systems, which do not usually require very low error rates. However, this is not true for other applications where the error floor hinders the ability of a turbo code to achieve required very low error rates.

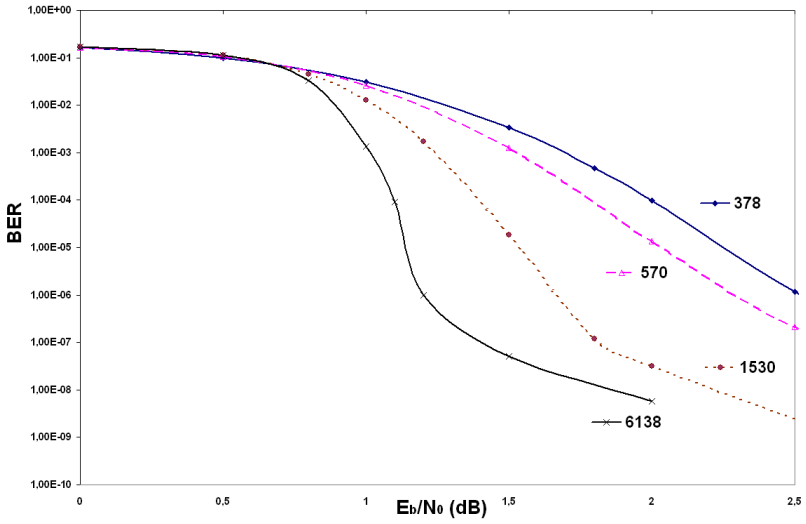


Figure 1.12: Bit-error performance of the 3GPP2 turbo code for various input code lengths at code rate 1/2. BPSK modulation is used over an AWGN channel. All simulations use the Max-Log-MAP algorithm with 10 decoding iterations.

Frame size k (bits)	Target BER $= 10^{-3}$
378	1.70
570	1.53
1530	1.24
6138	1.02

Table 1.4: Minimum value of E_b/N_0 (in dB) to achieve a BER$= 10^{-3}$ using the 3GPP2 turbo code.

1.6 How to combate the error floor ?

Turbo codes represent a huge advance in the field of forward-error-correction channel coding. The codes make use of three simple ideas: parallel concatenation of codes to allow simpler decoding, interleaving to provide better weight distribution and soft decoding to enable decoder interaction and iterative decoding.

The error rate curves of a turbo code can be divided into two main regions: the waterfall region where the error rate decreases rapidly; and the region of error floor where a change in the curve slope appears when the code reaches its asymptotic gain. This means that the error rate reaches a limit and stops improving even if the number of iterations is increased. The error floor is due to the presence of low-weight codewords. At low SNR, these codewords are insignificant, but as SNR increases, they begin to dominate the performance of the code [80]. It is naturally desirable to have turbo codes which have waterfalls as closest as possible to the channel capacity and low floors of error. Several ways can be used to combate the error flooring effects:

- ◇ One way is to use a slightly different RSC encoder with a more favorable distance

spectrum. However, in order to lower the error floor at high SNR, performance at low SNR will suffer. An interesting approach taken in [100] is to use two different RSC encoders. One RSC encoder is optimized to perform well at low SNR, while the other is optimized to reduce the error floor. The resulting asymmetric turbo code[3] provides a reasonable combination of performance at both a low and high SNR. Unfortunately, although the error floor has been reduced, it is still present.

- ◇ Increasing the MHD of a turbo code may involve devising more sophistical internal permutations. This is an appealing alternative to improve the MHD, since it does not incur any complexity penalty. Unfortunately, designing such powerful permutations is not an easy task, as explained in subsection 1.1.2.

- ◇ It may also involve using component encoders with a large number of states (16-state instead of 8-state components [15, 48]), at the price of doubling the decoding complexity each time we increase the memory length of the code by 1. This is not an appropriate solution because we want to raise the minimum distance and preserve a reasonable complexity at the same time.

- ◇ Another way to reduce the error floor is to arrange the two constituent encoders in a serial concatenation, rather than in a parallel concatenation [11]. Such serially concatenated convolutional codes yield higher minimum distances. However, performance at low SNR is considerably worse than it is for parallel concatenated codes. This penalty in convergence threshold might be unacceptable for several applications. An alternative to choosing between Serial Concatenated Convolutional Codes (SCCCs) and Parallel Concatenated Convolutional Codes (PCCCs) is to use hybrid turbo codes, which combine features of each type of code [44]. Several different hybrid concatenated structures have been proposed in the literature, e.g., [46]. Mixed structures, like those proposed in [56] or [70], are also possible.

- ◇ Last but not least, multiple concatenation using an increasing number of component encoders, can be used to eliminate low-weignt codewords and so improve the distance properties of the code. However, this will be paid in terms of loss in convergence and increase in complexity because each data has to be decoded more than twice.

1.7 Conclusion

The near-capacity performance of turbo codes and their suitability for practical implementation explain their adoption in various communication standards as early as the late nineties. In future system generations, low error rates will be required to open the way to real time and more demanding applications, such as TV broadcasting or videoconferencing. The MHD may not be sufficient to ensure large asymptotic gains at very low error rates. This phenomenon is the so-called flattening effect and it can be combated in several ways previously enumerated. In [20], an inner code was used to improve the

[3] If the component encoders are not identical then it is called an asymmetric turbo code.

distance properties of an outer turbo code. The idea is to mix the two last techniques in order to obtain a hybrid concatenated structure with an increasing number of component encoders. The next chapter focuses on this alternative. However, the techniques implemented to improve the floor usually degrade the convergence threshold. It is the compromise distance-convergence well identified in the literature. And in my thesis, I will be confronted many times to find a good balance between the two criteria, if it is not to be winning on both plans.

Chapter 2

Exploring 3-Dimensional turbo codes

TURBO codes [17] have been adopted in various communication standards [1, 2, 3, 4] due to their near-capacity performance and low decoding complexity. But they suffer from a flattening around 10^{-5} of Frame Error Rate (FER). In future system generations, lower error rates will be required to open the way to real time and more demanding applications, such as TV broadcasting or videoconferencing. In [20, 21], a 3-dimensional turbo code (3D TC) was introduced, combining both parallel and serial concatenation. It is simply derived from the classical turbo code by concatenating a rate-1 post-encoder at its output, which encodes only a fraction λ of the parity bits from the upper and lower constituent encoders. The fraction $1-\lambda$ of parity bits which is not re-encoded is directly sent to the channel or punctured to achieve the desired code rate. The 3D TC improves performance in the error floor compared to the TC, at the expense of an increase in complexity and a loss in convergence.

This chapter is organized as follows. In section 2.1, we present the 3D TC and its different parameters. The decoding process is also discussed in the same section. In section 2.2, the interest of the 3D TC is assessed through simulations in the case of the 3GPP2 code. Finally, a detailed study of the complexity increase of the 3D TC is presented in section 2.3.

2.1 Properties of 3-dimensional turbo codes

All the analysis presented below can be applied to any 3D TC in a straightforward manner. However, we focused in our simulations on the 3GPP2 code, an 8-state binary turbo code, used in the third generation (3G) cdma2000 mobile phone communication systems (see section 1.5). We remind that the 3GPP2 turbo code is built from the parallel concatenation of two 8-state RSC codes, with generator polynomials 13 (recursivity) and 15 (redundancy). The overall turbo code code rate before puncturing is 1/3 since only the first parity in each component code is considered (see section 1.5).

2.1.1 3D turbo encoder structure

A block diagram of the 3D turbo encoder is depicted in Fig. 2.1. A fraction λ of the parity bits from the upper and lower constituent encoders are grouped by a Parallel to Serial (P/S) multiplexer, permuted by a permutation Π', and encoded by an encoder of unity rate whose output is denoted by w. The iterative decoding of concatenated codes in general makes it important to have Π' in order to decorrelate the extrinsic information. In [20, 21], λ is referred to as the permeability rate. Usually, very simple regular permeability patterns are applied. For instance, if $\lambda = \frac{1}{8}$ the bits to be post-encoded are chosen in a regular basis {10000000} for both the upper and the lower encoders. Note that the permeability rate has an effect on the performance of the 3D TC similar to the doping ratio concept of [105].

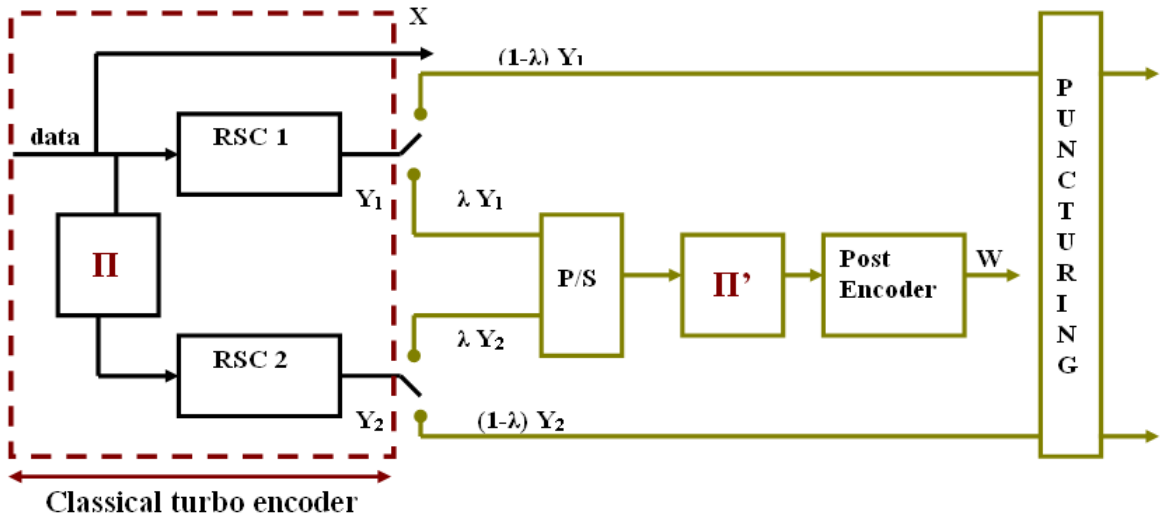


Figure 2.1: 3D turbo encoder structure.

2.1.2 3D turbo decoder

As depicted in Fig. 2.2, the classical turbo principle is used to decode the 3D turbo code. We have three elementary decoders corresponding to the three constituent encoders; and

all of them exchange extrinsic information. First, the 4-state SISO pre-decoder is activated to feed the two SISO decoders with extrinsic information about the post-encoded parity bits. The two decoders exchange extrinsic information about the systematic bits, as for the classical turbo procedure. They also provide the pre-decoder with extrinsic information about the post-encoded parity bits. The decoding process continues iteratively until all constituent decoders have converged, or a maximum number of iterations has been performed.

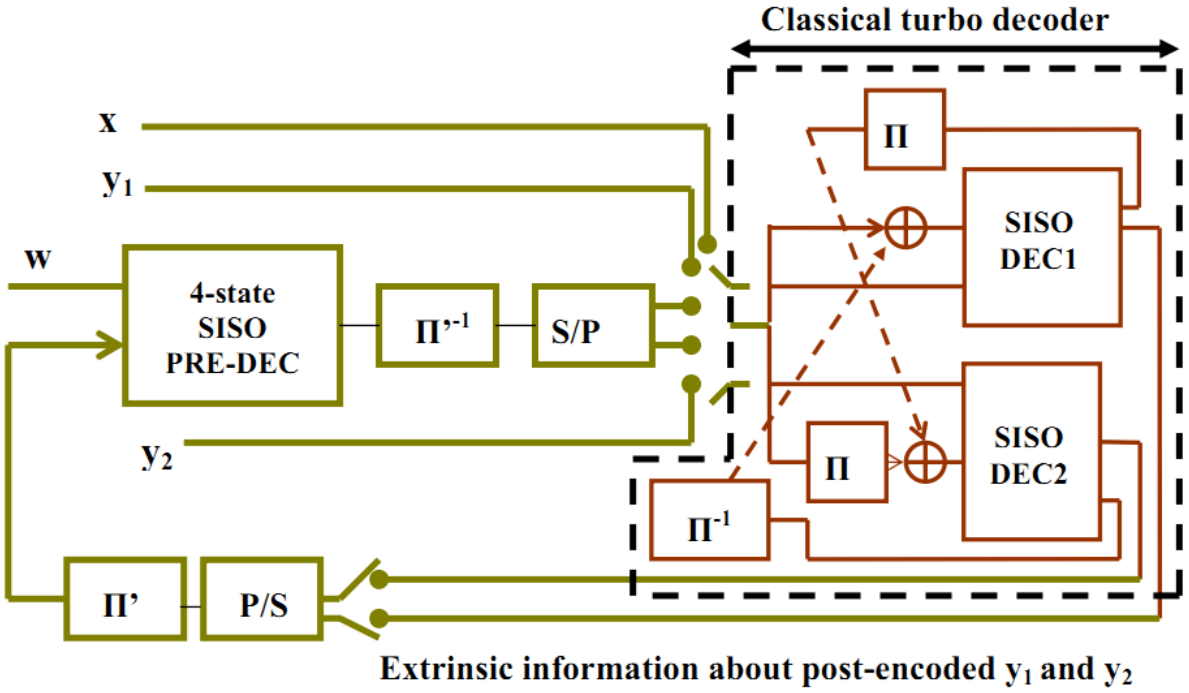


Figure 2.2: Structure of the 3D turbo decoder.

2.1.3 Choice of the permeability rate

The value of λ can be used to trade-off performance in the waterfall region with performance in the error floor region. The case $\lambda = 0$ corresponds to the standard parallel turbo code and the case $\lambda = 1$ is a serial concatenation between the classical turbo code and the post-encoder. Increasing $0 < \lambda < 1$ turns the code into more serial, hence increasing its minimum distance. However, a large value of λ penalizes the decoder from the convergence point of view. In fact, the more redundancies are post-encoded the less redundant information at the first iteration the decoder will have, then causing more errors at its output at the first processing.

Let P be the number of bits that pass through the post-encoder. The fraction θ of the codeword bits that are post-encoded is:

$$\theta = \frac{P}{n} = \frac{2\lambda k}{n} = 2\lambda R$$

where $n = \frac{k}{R}$ is the codeword length and R is the overall code rate of the 3D turbo code. We denote by R_i the code rate of each constituent encoder (*i.e.,* R_1 is the code rate of RSC1 and R_2 is the code rate of RSC2). The fraction θ_i of the data processed by the component decoder of each constituent encoder, that have to pass through the predecoder can be expressed as follows:

$$\theta_i = \lambda \frac{\frac{k}{2}}{k + \frac{k}{2}} = \lambda \left(\frac{k}{2}\right) \frac{R_i}{k}$$

The parallel concatenation, which associates two elementary codes with rates R_1 and R_2, has a global coding rate: $R = \frac{R_1 R_2}{R_1 + R_2 - R_1 R_2}$. Since the turbo code is symmetric (*i.e.,* $R_1 = R_2$), then $R = \frac{R_i^2}{2 \times R_i - R_i^2}$. Thus, we obtain $R_i = \frac{2R}{R+1}$, and θ_i can be expressed in the following way:

$$\theta_i = \lambda \frac{R}{R+1} \tag{2.1}$$

Now, if p is the probability of error at the channel output and φ is the ratio of the probability of error at the predecoder output divided by the probability of error at its input, the average probability of error p' at each decoder intrinsic input is:

$$p' = \varphi \theta_i p + (1 - \theta_i)\, p = (1 + (\varphi - 1)\, \theta_i)\, p$$

that is, from equation (2.1):

$$p' = \left(\frac{1 + (1 + (\varphi - 1)\, \lambda)\, R}{1 + R}\right) p \tag{2.2}$$

Equation (2.2) shows that the probability of error at each decoder intrinsic input is risen by a factor $\left(\frac{1+(1+(\varphi-1)\lambda)R}{1+R}\right)$, inducing a loss in convergence.

Fig. 2.3 illustrates the effect of λ on the error rate curves of a 3D TC: if we choose a large value of λ, the minimum distance is significantly increased. However, performance in the error floor region will be paid from a convergence point of view. Thus, a trade-off between convergence loss and required minimum distance has to be found. In our simulations, $\lambda = 1/8$ and $\lambda = 1/4$ are considered, since they represent a good trade-off between convergence and minimum Hamming distance.

In Fig. 2.4, we report the FER performance of the 3GPP2 3D TC to compare it with that of the 3GPP2 TC for the block size 3066 bits, at coding rate $R = 1/3$ for both $\lambda = 1/4$ and $\lambda = 1/8$. In our simulations, the 4-state convolutional code (5,4) was selected to be the post-encoder. Also a regular permutation Π' was used to spread to parity bits before feeding them to the post-encoder.

We observe a small loss of convergence in the waterfall region when the permeability rate is $\lambda = 1/8$. This loss of convergence increases with λ. It is about 0.26 dB for $\lambda = 1/4$. Furthermore, the all-zero iterative decoding algorithm (see 1.4.3) is applied to estimate performance in the error floor region without resorting to long simulations. In the simulated case, the use of 3GPP2 3D TC results in an increase in the MHD by 65 %, from MHD = 23 to MHD = 38, for code rate $R = 1/3$, $\lambda = 1/8$ and $k = 3066$ bits. The

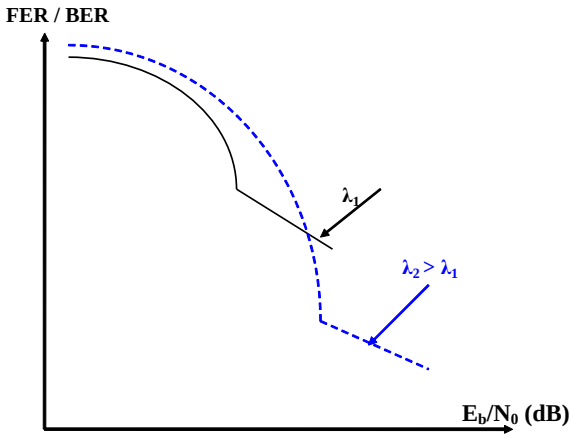


Figure 2.3: Choice of the permeability rate λ.

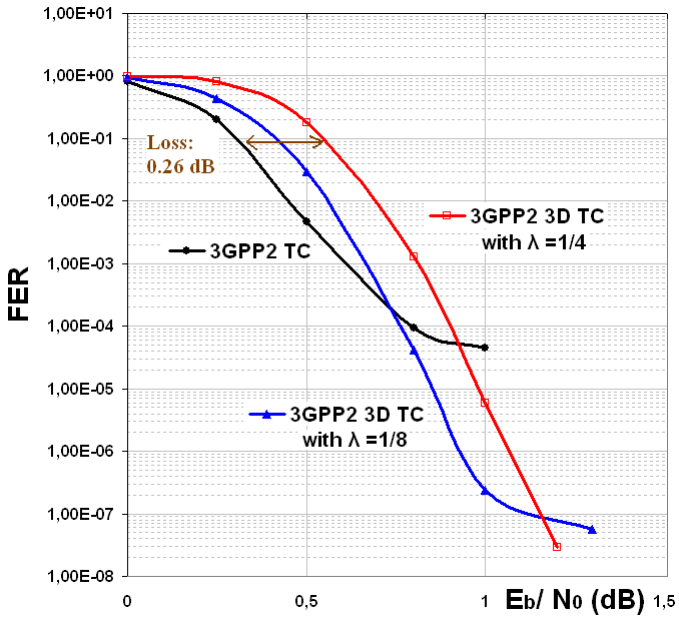


Figure 2.4: FER performance of the 3GPP2 3D TC with $\lambda = 1/4$ and $\lambda = 1/8$ for $k = 3066$ bits, $R = 1/3$ and comparison with the 3GPP2 TC. All simulations use the MAP algorithm with 10 decoding iterations.

gain obtained with $\lambda = 1/4$ is even larger as shown in Fig. 2.4. This example illustrates that λ is a key parameter to choose the compromise between distance and convergence.

Note:

Given λ, and without puncturing information bits, the highest achievable code rate is $R_{max} = \frac{1}{1+2\lambda}$, since the overall code rate of the 3D turbo code is given by:

$$R = \frac{1}{1 + 2\lambda + 2(1-\lambda)\rho}$$

where $0 \leq \rho \leq 1$ is the fraction of the surviving bits in y_1 and y_2 after puncturing. In fact we have k systematic bits, $2\lambda k$ post-encoded parity bits and $2(1-\lambda)\rho k$ surviving parity bits after puncturing. For example, if the overall coding rate is $R = \frac{4}{5}$, then it is necessary to have a permeability rate $\lambda \leq \frac{1}{8}$.

2.1.4 Choice of the post-encoder

The choice of the post-encoder influences the performance in both the waterfall and error floor regions. In general, the post-encoder must be simple to limit the complexity increase of the corresponding decoder. Low memory RSC codes satisfy this requirement. Besides, the code is made tail-biting (see 1.1.3) to prevent from any side effects as the initial state and the final state of the post-encoder are identical. This requirement is important for real–time and demanding applications, such as TV broadcasting or videoconferencing, where very low error rates are sought for. Therefore, the accumulator (*i.e.*, the code with memory one) has to be discarded since it cannot be made circular using tail-biting encoding. Last but not least, the post-encoder must not exhibit too much error amplification, to prevent from a high loss in convergence. In practice, a 4-state binary convolutional encoder is used. Three linear RSC codes having memory 2 are given in Fig. 2.5.

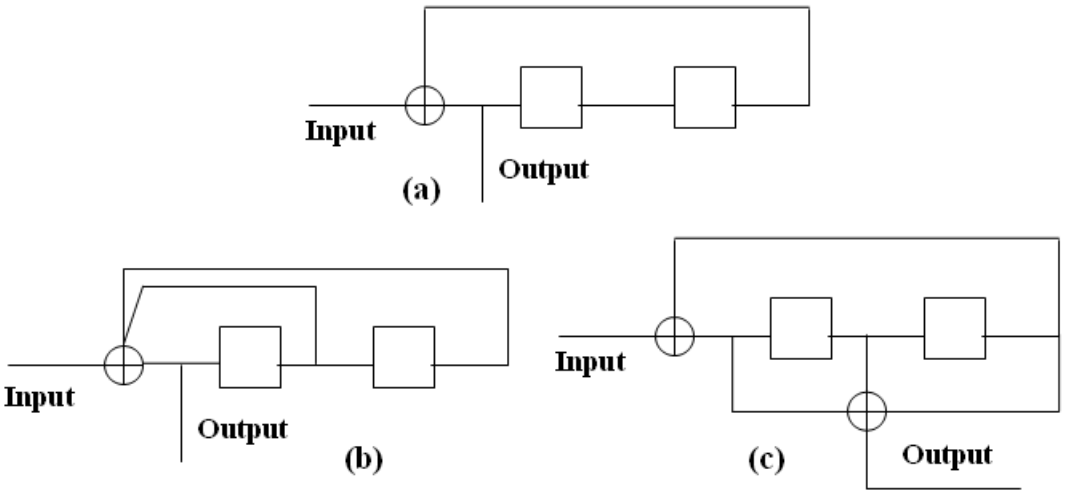


Figure 2.5: Possible linear post-encoder candidates with memory 2.

To complete the analysis in [20, 21], the choice of the post-encoder is justified by means of EXtrinsic Information Transfer (EXIT) analysis. This is one of the contributions of the thesis.

2.1.4.1 EXIT analysis

In Fig. 2.6, we report the EXIT curves for the three linear post-encoders of Fig. 2.5. When no *a priori* information is available at the input of the pre-decoder (*i.e.* first iteration), the Mutual Information (MI) at its output is higher for post-encoder (a). In fact, code (a)

has a corresponding decoder which only doubles the number of errors of its input at the first step of the iterative process, while code (b) will roughly triple the number of errors at the first step. The worst case occurs with code (c) because its decoder causes a mistake once every two bits in its entry. In fact for code (c) the feedforward polynomial (*i.e.*, 7 in octal or $1 + D + D^2$) and the feedbackward polynomial (*i.e.*, 5 in octal or $1 + D^2$) are not reciprocal[1]. In this case the corresponding decoder engenders 50% of errors at the first step of the iterative process, since it is impossible to extract 7 from the feedbackward polynomial, *i.e.*, 5.

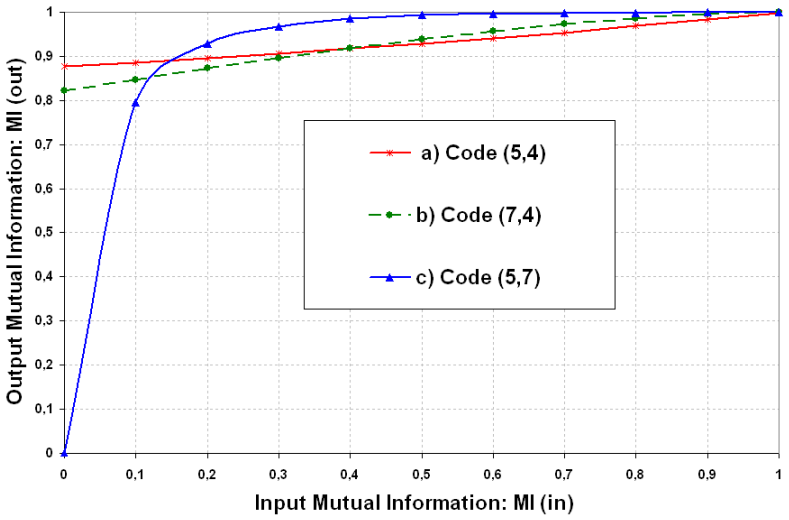
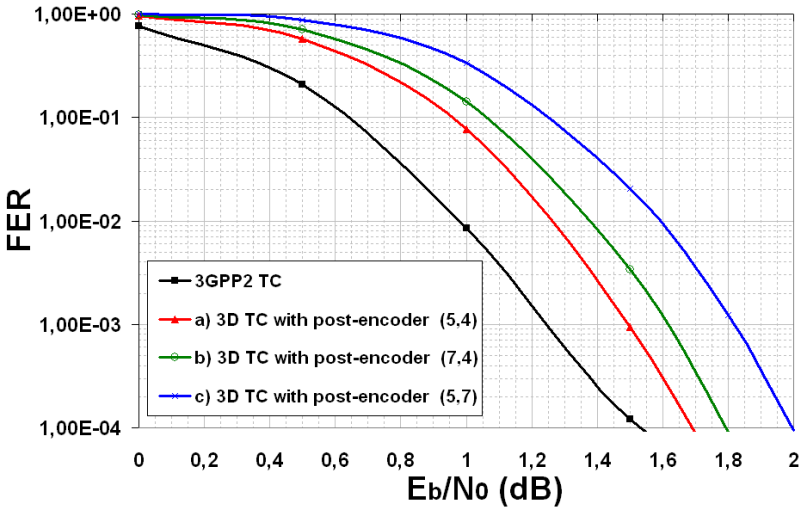


Figure 2.6: EXIT curves for different linear post-encoders.

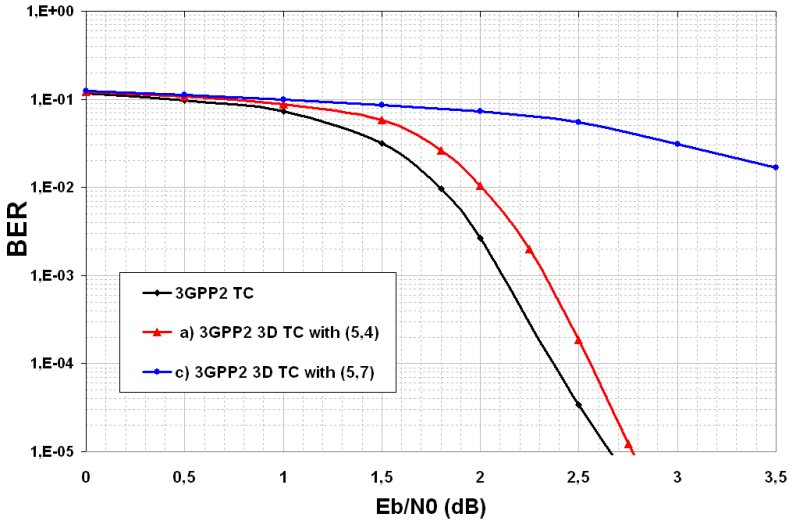
Let us assume that a post-encoder, where the MI at its output is zero when there is no MI at its input (such as code (c)), has been selected. The worst case occurs when all the parity bits are post-encoded, which corresponds to high coding rates such as code rate $R = 2/3$ for $\lambda = 1/4$ or code rate $R = 4/5$ for $\lambda = 1/8$. In this case, the error rate at the output of the corresponding pre-decoder at the first iteration will be 0.5. And the turbo decoder will have no parity to decode with at the first step of the iterative process. It will just be something catastrophic as the performance will not be improved through the iterative process! Therefore, the EXIT analysis is a very important tool to select a post-encoder convenient at low but also at high coding rates.

In Fig. 2.7a, we report the FER performance of the 3GPP2 3D TC to compare it with that of the 3GPP2 TC for the block size 570 bits, at coding rate $R = 1/3$ and $\lambda = 1/4$. We observe a loss of convergence in the waterfall region when the post-encoder of Fig. 2.5 (a) is used. As expected, this loss of convergence increases when the post-encoder of Fig. 2.5 (b) is used. The largest loss of convergence is observed when the code of Fig. 2.5 (c) is used. Similar simulations (see Fig. 2.7b) at code rate $R = 2/3$ for $\lambda = 1/4$, confirm that the 3D TC does not converge when the code of Fig. 2.5 (c) is selected to be the post-encoder.

[1]This means that we can not express $G_1(D) = 1 + D + D^2$ as $D^M \times G_2(\frac{1}{D}) = D^M \times (1 + \frac{1}{D^2})$ or vice-versa.



(a) FER performance for $k = 570$ bits and $R = 1/3$. All simulations use the Max-Log-MAP algorithm with 10 decoding iterations.



(b) BER performance for $k = 1146$ bits and $R = 2/3$. All simulations use the MAP algorithm with 10 decoding iterations.

Figure 2.7: Error rate curves of the 3GPP2 3D TC with $\lambda = 1/4$ and comparison with the 3GPP2 TC.

Therefore, due to its better convergence, code (a) with generator polynomial 5 (recursivity) and 4 (redundancy) has been selected to be the post-encoder in different simulations of the 3D TC. However, the main drawback is that code (a) cannot ensure tail-biting encoding in order to properly deal with blocks of data. In other words, we can not ensure that the initial state and the final state of the post-encoder are identical. Thus, state mapping encoding has been introduced in [98]. The problem can easily be resolved by an exchange of metrics at the end of the forward and backward recursions. Details are provided in Appendix B.

2.1.4.2 Statistics about the predecoder corresponding to the selected post-encoder

A thorough analysis of the 3D decoder has been carried out to determine in which cases the pre-decoder, corresponding to the post-encoder of Fig. 2.5 (a), engenders an error in the decoder. In other words, if the 3D decoder makes an erroneous decision, is the pre-decoder responsible for it or not? In order to have clarifications about this topic, statistics have been carried out for $\lambda = 1/8$. In the case of $\lambda = 1/4$, similar results are obtained. However, the simulation time is more significant especially in the error floor region since lower error rates can be reached compared with $\lambda = 1/8$[2]. Also, different signal to noise ratios are taken under consideration in our analysis. Table 2.1 summarizes the results of this investigation. First, it is observed that for a low signal to noise ratio, corresponding to the region where the 3D TC looses in convergence compared to the classical TC, the errors due to the pre-decoder represent less than 3.5% at the first and the second iteration of the total errors committed by the 3D decoder. Then, for a medium signal to noise ratio, corresponding to the region where the performance of the 3D TC is better than that of the classical TC, it is similarly observed that the errors due to the pre-decoder represent less than 2.5% at the first and the second iteration of the total errors committed by the 3D decoder. Finally, in the error floor region, all the committed errors are not due to the pre-decoder which works perfectly.

E_b/N_0 (dB)	1	2	2.5
Zone of the error rate curve	Loss in convergence threshold	3D TC performs better than classical TC	Error floor
Iteration 1	3.34 %	2.15%	0%
Iteration 2	2.66%	1.55%	
Iteration >=3	0%	0%	

Table 2.1: Statistics about the predecoder for blocks of $k = 1530$ bits, $\lambda = 1/8$ and more than 10^{12} simulated binary samples.

As a conclusion, the relative number of errors due to the pre-decoder decreases with the SNR. They are estimated, at the first and the second iteration, to be less than 3.5%. Up to the third iteration for low and medium SNR and similarly at high SNR, the pre-decoder is not responsible at all for the errors committed by the 3D decoder. This observation opens the way for the use of other rate-1 post-encoders with lower correction capability which may perform better from a convergence point of view.

[2]For $\lambda = 1/8$ and $k = 1530$ bits, the FER of the 3D TC reaches the value of 10^{-8}. In order to have reliable results, more than ten erroneous blocks have to be simulated. This corresponds to more than 10^{12} simulated binary samples.

2.1.5 Permutations of a 3D turbo code

The 3D TC is characterized by two permutations denoted by Π and Π', as shown in Fig. 2.1. In theory, both permutations should be jointly optimized. However, Π is the internal permutation of the TC, and we keep Π unchanged with regard to the original code for reasons of backward compatibility. Π' is used to spread a fraction λ of the parity bits before feeding them to the post-encoder. In other words, Π' is used to spread $P = 2\lambda k$ parity bits at the output of the turbo code before post-encoding. The main role of the permutation Π' is to avoid that the pre-decoder returns packages of errors to the entry of the main decoder.

To illustrate the situation, let us consider a rate-1/3 turbo code with regular rectangular permutation Π (row-wise writing and column-wise reading). The main interest of our choice is that it is easy to visualize the permutation Π in this case. The constituent encoders are 8-state encoders whose period is 7 and the recursivity generator is 15. The input weight 4 square error pattern shown in Fig 2.8 is a "composite" error pattern consisting of four identical input weight 2 elementary Return To Zero (RTZ) sequences: 10000001. The weight of the corresponding redundancy sequences produced before and after permutation Π, y_1 and y_2, is equal to 6. The resulting total weight of this codeword, equal to 28, allows low error rates to be reached. However, when this code is punctured to coding rate 1/2, the resulting weight, 16, may be not high enough for some applications.

The role of 3D part of the encoder is to take a small fraction of the parity bits in y_1 and y_2, to interleave and then to re-encode them with the post-encoder. Hopefully, a few 1s of the redundancy part of the error pattern in Fig 2.8 will be moved away to each other and will produce a significant of additionnal 1s when post-encoded, thus increasing the total codeword weight.

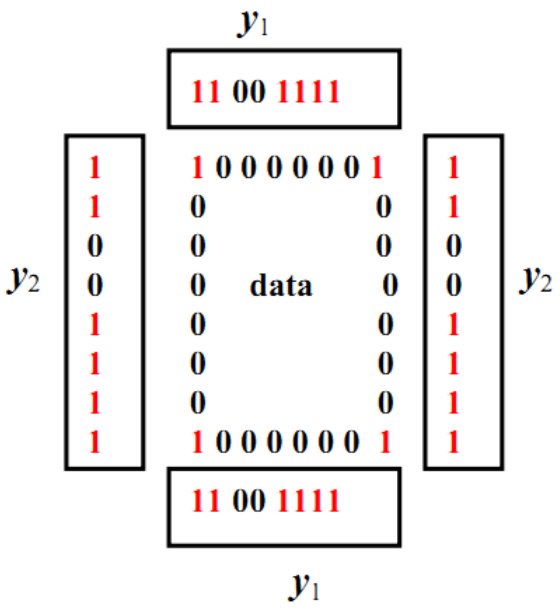


Figure 2.8: Possible error patterns of input weight 4 for a rate 1/3 turbo code.

Accordingly, a good permutation Π′ must break the regularity of rectangular composite

patterns like those of Fig 2.8 in order to increase the weight of the elementary error patterns and thereby the minimum distance of the 3D TC. To optimize Π', different types of interleavers were tested starting from random permutations to more structured permutations such as the regular interleaver.

Let i and j be the address in the natural order, and in the permuted order, respectively. For the regular permutation in circular form, we assume Π' to be defined by the following congruence relation: $i = \Pi'(j) = P_0 j + i_0 \,\%\, P$, where i_0 is the starting index, and P_0 is an integer relatively prime with P. For each block length, these parameters have to be carefully chosen to guarantee a large spread (see 1.1.2).

In fact, it was observed through the different simulations that the important property is the spread and performance of the code associated to an interleaver is degraded by low values of spread. The regular permutation achieves the maximum spread value of $\sqrt{2P}$ [24], where P is the size of the frame to be post-encoded. So it performs better than a random interleaver in terms of MHD and convergence.

Example:

For blocks of $k = 6138$ bits and $\lambda = 1/4$, we have $P = 2\lambda k = 3069$ bits. The parameters of the regular permutation are: $P_0 \approx \sqrt{2P} \Rightarrow P_0 = 79$ and $i_0 \approx \frac{P_0}{2} \Rightarrow i_0 = 37$.

Fig. 2.9 shows the simulated FER performance of the 3GPP2 3D TC with random and regular interleavers Π' for code rate $R = 1/2$, $\lambda = 1/8$ and $k = 762$ bits.

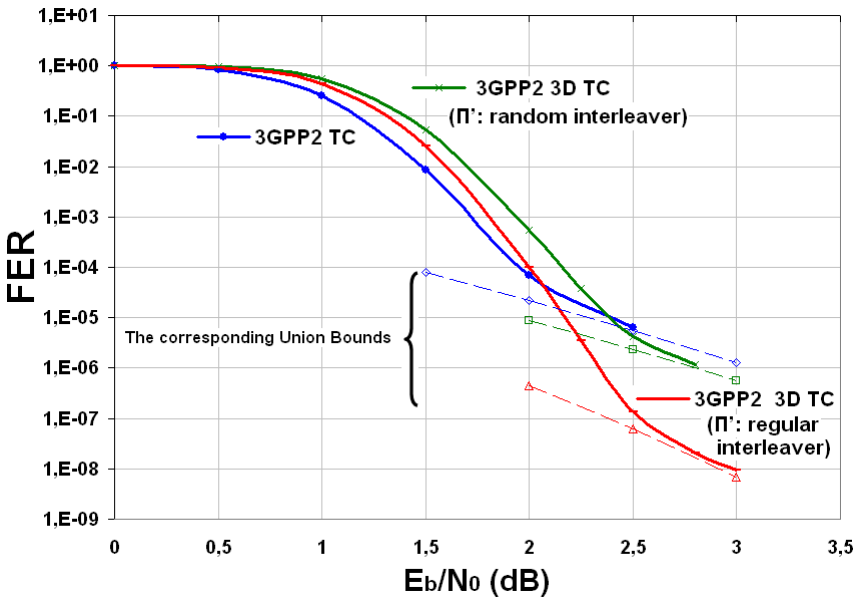


Figure 2.9: FER performance of the 3GPP2 3D TC with $\lambda = 1/8$ for $k = 762$ bits, $R = 1/2$ and comparison with the 3GPP2 TC. All simulations use the Max-Log-MAP algorithm with 10 decoding iterations.

The 3GPP2 3D TC using a random permutation Π' does not perform well in terms of MHD, but also in terms of convergence. However, the use of a regular permutation Π' results in an increase in the MHD of the 3GPP2 3D TC compared to the standardized 3GPP2 turbo code. In Fig. 2.9, an increase by more than 60 % is observed, which provides a gain of more than two decades in the error floor. These simulation results were confirmed with the asymptotical bounds as shown in Fig. 2.9. In fact, for transmission over the Gaussian channel, the FER can be upperbounded by the union bound:

$$FER < \frac{1}{2} \sum_{d>=d_{min}} n(d)\, erfc\left(R\, d\, \frac{E_b}{N_0}\right)$$

where $n(d)$ is the code multiplicity (number of codewords with weight d), and erfc(x) is the complementary error function. Here again, the all-zero iterative decoding algorithm (see 1.4.3) was applied to obtain the distance spectrum.

2.2 Performance of cdma2000 3D turbo codes

We have investigated the distance gain and the effect on turbo code convergence threshold for different block sizes, coding rates and permeability rates. Similarly to the case of double-binary codes in [20, 21], we have observed that the addition of the post-encoder improves the asymptotical behavior of the 3GPP2 turbo code in many cases.

Table 2.2 presents examples of MHD values obtained with this code for different block sizes and coding rates, using the all-zero iterative decoding algorithm. The authors in [64, 65] analyzed the asymptotic weight distribution of 3D TCs and showed that their typical minimum distance may, depending on certain parameters, asymptotically grow linearly with the block length.

Note that there are few boxes in Table 2.2 containing the *not applicable* abbreviation N/A. In fact, if the coding rate is $R = 4/5$, we can not insure a post-encoding with $\lambda = 1/4$ since half of the post-encoded bits are going to be punctured (see 2.1.3). To avoid the problem, we can puncture the systematic bits but it would lead to additionnal convergence loss.

We can observe that the direct application of the third coding dimension to the existing code leads to an increase of its minimum distance, except in the case of high coding rates. Table 2.2 shows that it is attractive to increase λ, since larger minimum distances are obtained. However, this will be paid in terms of loss in convergence threshold, as explained in 2.1.3. Also, the increase in complexity is not negligible. The aim of the next section 2.3 is to carry a thorough study in order to estimate the additional complexity. This represents another contribution of the thesis.

$k = 570$ **bits**	$R = \frac{1}{3}$	$R = \frac{1}{2}$	$R = \frac{2}{3}$	$R = \frac{4}{5}$
3GPP2 TC	17	10	6	4
3D TC ($\lambda = \frac{1}{4}$)	29	17	13	**N/A**
3D TC ($\lambda = \frac{1}{8}$)	27	17	9	4
3D TC ($\lambda = \frac{1}{16}$)	25	15	8	4

$k = 762$ **bits**	$R = \frac{1}{3}$	$R = \frac{1}{2}$	$R = \frac{2}{3}$	$R = \frac{4}{5}$
3GPP2 TC	19	11	6	4
3D TC ($\lambda = \frac{1}{4}$)	39	23	9	**N/A**
3D TC ($\lambda = \frac{1}{8}$)	30	18	8	4
3D TC ($\lambda = \frac{1}{16}$)	21	14	6	4

$k = 1530$ **bits**	$R = \frac{1}{3}$	$R = \frac{1}{2}$	$R = \frac{2}{3}$	$R = \frac{4}{5}$
3GPP2 TC	28	14	8	5
3D TC ($\lambda = \frac{1}{8}$)	36	18	8	5

$k = 2298$ **bits**	$R = \frac{1}{3}$	$R = \frac{1}{2}$	$R = \frac{2}{3}$	$R = \frac{4}{5}$
3GPP2 TC	25	11	8	
3D TC ($\lambda = \frac{1}{8}$)	36	18	8	

$k = 3066$ **bits**	$R = \frac{1}{3}$	$R = \frac{1}{2}$	$R = \frac{2}{3}$	$R = \frac{4}{5}$
3GPP2 TC	23	13	7	5
3D TC ($\lambda = \frac{1}{8}$)	38	21	10	6

$k = 6138$ **bits**	$R = \frac{1}{3}$	$R = \frac{1}{2}$	$R = \frac{2}{3}$	$R = \frac{4}{5}$
3GPP2 TC	30	15	9	5
3D TC ($\lambda = \frac{1}{8}$)	38	30	10	5

Table 2.2: Minimum Hamming distance values for the 3GPP2 and the 3GPP2 3D turbo codes for different coding rates and block sizes.

2.3 3D turbo codes hardware implementation issues: decoder architecture and complexity analysis

In [20, 21], the complexity increase was estimated to be less than 10% with respect to classical 2-dimensional TC. In this section, we propose an appropriate hardware architecture of a 3D turbo decoder and the corresponding complexity analysis. In fact, compared to a classical turbo decoder, the additional complexity of the 3D turbo decoder is mainly due to the implementation of the binary 4-state decoder but also to the calculation of the extrinsic information about the post-encoded parity bits.

2.3.1 3D turbo decoder architecture

The typical overall turbo decoder architecture is composed of three modules, represented in Fig. 2.10. First, the input module receives the input frames and transmits them to the decoder module. It requires a double input buffer, in order to receive the next frame while decoding the current one. The input buffer is divided into as many memory banks (MB) as the number of processors placed in parallel (*i.e.* $Proc$). This parallelism allows having different throughputs according to the application. Then, the decoder module performs I iterations on the frame stored in the input module and writes the decoded codeword into the output module. This module contains $Proc$ SISO processors and an extrinsic memory decomposed into as many memory banks as the number of physical processors (not represented in the figure). A finite state machine (not represented) controls the processors. For each iteration, the set of $Proc$ processors has to perform the decoding of the component codes. At the end, the output module stores the hard decisions produced by the decoder module and sends them to the output of the decoder. In the case of 3D TC, since the pre-decoder has much less data to process than the main SISO decoders (only $\lambda = 1/8$ or $\lambda = 1/4$) of the parity bits are re-encoded by the post-encoder), no parallelism is considered for the pre-decoder.

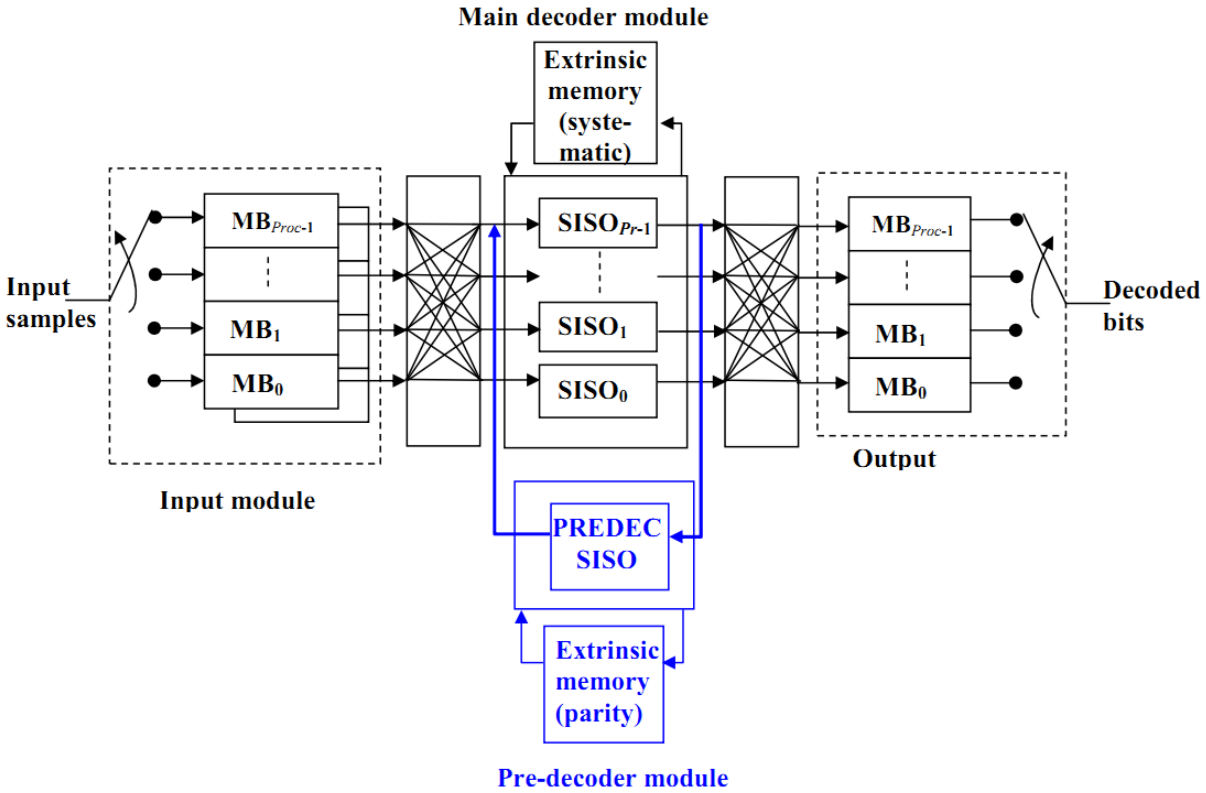


Figure 2.10: Generic 3D turbo decoder organization for Pr processors.

2.3.2 Max-Log-MAP decoder complexity analysis

To analyze the complexity of 3D TCs, let us consider a RSC code with the following parameters: ν is the memory length of the code, n is the number of coded bits provided by the encoder at each trellis stage when no puncturing is performed, and k is the trellis length. For a classical binary turbo code, k is also the length of information sequence, in terms of binary bits.

This subsection details the different steps of the decoding process and the associated decoding complexity, in terms of arithmetic and logical operations. We assume a transmission over an AWGN channel with noise variance σ^2, using BPSK modulation. However, the same decoder is used for transmissions over fading channels. Then, the channel coefficients are taken into account upstream of the decoder. The following description of the algorithm is also valid when high order modulations are considered, in the case where a Bit-Interleaved Coded Modulation (BICM) approach [28] is adopted.

◇ **Computation of branch metrics** $met_t(s', s)$ **:**

At time step t, the metric associated with trellis branch or transition (s', s) is defined as

$$met_t(s', s) = \pm x_t \pm y_{t,1} \pm ... \pm y_{t,n-1} + z_t \tag{2.3}$$

where x_t is the received systematic data, $y_{t,1} \pm ... \pm y_{t,n-1}$ are the $n-1$ received redundant data, and z_t is the *a priori* incoming information. The computation of the 2^n different values $\pm x_t \pm y_{t,1} \pm ... \pm y_{t,n-1}$ requires $2^{n+1} - 4$ additions/subtractions. The addition of the *a priori* term requires two extra additions. We assume that the computation of the branch metrics is performed twice, once for the forward recursion and once for the backward recursion.

◇ **Computation of forward and backward state metrics for each trellis stage** s**:**

The state metrics are computed recursively using the following relations:

Forward recursion:

$$M_t^F(s) = \min_{s' \in \{0,...,2^\nu - 1\}} \left(M_{t-1}^F(s') + met_{t-1}(s', s) \right) \tag{2.4}$$

Backward recursion:

$$M_t^B(s) = \min_{s' \in \{0,...,2^\nu - 1\}} \left(M_{t+1}^B(s') + met_{t-1}(s, s') \right) \tag{2.5}$$

According to the equations (2.4) and (2.5) above, the update of one forward state metric involves the comparison and selection of two concurrent paths that can be performed using two additions and one comparison-selection operation, implementing a tree structure. The update of backward state metrics requires the same number of operations.

◇ **Computation of the soft decisions (⇔** ***a posteriori*** **LLRs) and hard decisions**:

If we denote by $\lambda_t(\delta)$ the soft information defined as

$$\lambda_t(\delta) = \min_{(s',s)} \left(M_t^F(s') + met_t(s',s) + M_{t+1}^B(s) \right) \tag{2.6}$$

where $\delta \in \{0,1\}$, the *a posteriori* log-likelihood related to data at time step t is computed as

$$L_t(\delta) = \frac{1}{2} \left(\lambda_t(\delta) - \min_{\delta' \in \{0,1\}} \lambda_t(\delta') \right) \tag{2.7}$$

Term $\min_{\delta' \in \{0,1\}} \lambda_t(\delta')$ is a normalization term.

The hard decision provided by the decoder corresponds to the binary representation of δ that minimizes $\lambda_t(\delta)$ and makes $L_t(\delta)$ equal to zero.

$$\hat{\delta} = \arg \min_{\delta \in \{0,1\}} (L_t(\delta)) = \arg \min_{\delta \in \{0,1\}} (\lambda_t(\delta))$$

The computation of two *a posteriori* LLRs requires the computation of two values of $\lambda_t(\delta)$, $\delta \in \{0,1\}$. Relation (2.6) involves two additions for each transition in the trellis. This complexity can be reduced to one addition by observing that partial terms $M_t^F(s') + met_t(s',s)$ or $met_t(s',s) + M_{t+1}^B(s)$ are already available through the forward or backward recursion.

For each value of δ, the minimum value of 2^ν terms $M_t^F(s') + met_t(s',s) + M_{t+1}^B(s)$ has to be computed, resulting in $2^\nu - 1$ compare-select operations, using a tree structure. Consequently, the computation of two values for $\lambda_t(\delta)$ requires $2^{\nu+1}$ additions and $2(2^\nu - 1)$ comparisons and selections.

The computation of two *a posteriori* LLRs from equation (2.7) requires a compare and select tree to compute the *min* term, that is one compare and select operation, two subtractions and two divisions by 2. Actually the subtraction in the case of $\lambda_t(\hat{\delta})$ can be avoided, since $L_t(\hat{\delta}) = 0$ and the number of subtractions can be reduced to one. Divisions by 2 are not taken into account in the operator calculation, since they only come to remove the least significant bit.

The hard decision $\hat{\delta}$ can be directly inferred from the compare and select tree allowing the minimum value of $\lambda_t(\delta)$ to be computed.

◇ **Computation of extrinsic information $L_t^e(\delta)$ related to information bits**:

The extrinsic information computation is similar to the *a posteriori* log-likelihood $L_t(\delta)$, using the extrinsic branch metrics. We compute the extrinsic soft information $\lambda_t^e(\delta)$ defined as

$$\lambda_t^e(\delta) = \min_{(s',s)} \left(M_t^F(s') \pm y_{t,1} \pm ... \pm y_{t,n-1} + M_{t+1}^B(s) \right)$$

where the *min* operation is performed among 2^ν transitions corresponding to data value δ. Then, the extrinsic log-likelihood $L_t^e(\delta)$ is computed as follows

$$L_t^e(\delta) = \frac{1}{2} \left(\lambda_t^e(\delta) - \lambda_t^e(\hat{\delta}) \right)$$

The term subtracted to $\lambda_t^e(\delta)$ is the extrinsic value corresponding to the hard decision $\hat{\delta}$.

If we assume that terms $\pm x_t + z_k$ have already been made available during the branch metrics computation step (equation (2.3)), each piece of extrinsic information is obtained from the *a posteriori* LLR with two subtractions. The total extrinsic information computation is then performed using 2^2 subtractions.

◇ **Computation of extrinsic LLRs related to redundancy bits**:

In the case of 3D TCs, additional extrinsics related to re-encoded redundancy bits have to be computed by the main SISO decoders. Each additional extrinsic LLR is computed from the following relation:

$$L_t^y = \frac{1}{2}\left[\min_{(s',s)/y_t=0}\left(M_t^F(s') + met_t(s',s) + M_{t+1}^B(s)\right) - \min_{(s',s)/y_t=1}\left(M_t^F(s') + met_t(s',s) + M_{t+1}^B(s)\right)\right] - y_t$$

where y is the considered redundancy bit.

Observing that terms $M_t^F(s') + met_t(s',s) + M_{t+1}^B(s)$ are already available, we only have to compute the minimum value of these terms for value redundancy 0 and for redundancy 1. Thus two minimum values have to be computed among 2^ν terms, resulting in using two tree structures requiring $2^\nu - 1$ compare-select operations each. Then, the extrinsic LLR is computed by subtracting these two values, dividing by 2 and subtracting the received redundancy bit. Consequently, the computation of each additional extrinsic redundancy value requires two subtractions and $2\,(2^\nu - 1)$ compare-select operations.

Table 2.3 summarizes the resulting complexity for the process of a trellis stage, or equivalently of an information bit. The corresponding numerical values are given in Table 2.4. In order to compare the complexity of the different families of decoders, it is assumed that addition/subtraction and compare-select operators have similar hardware complexity. This complexity assessment does not take the size of the operators into account.

2.3.3 Memory requirements for the 3D turbo decoder

The memory requirements for the turbo decoder are the amount of both RAM and ROM memory. A very small amount of ROM memory is required to store the turbo code permutation parameters. This amount of memory is the same for all coding schemes under consideration. The RAM memory requirements are detailed below. We assume that the data at the input of the decoder are quantized on q_x bits.

◇ Two input buffers (single-port RAM) are necessary for each data sequence, including systematic and parity bits, stemming from the transmission channel (see Fig. 2.10). Thus, if k is the length of the information sequence, $\frac{2k}{R}$ input samples, quantized on q_x bits, have to be stored at the decoder input.

	Add (or substract)	Compare-select
Branch metrics (forward or backward recursion)	$2^{n+1}-2$	
One step of recursion (forward or backward)	$2^{\nu+1}$	2^{ν}
A posteriori LLRs and hard decision	$2^{\nu+1}+1$	$2^{\nu+1}-1$
Extrinsic LLRs for information bits	4	
Extrinsic LLRs for redundancy bits	2	$2^{\nu+1}-2$
Total computational requirement per information bit for classical TC	$3\times 2^{\nu+1}+2^{n+2}+1$	$2^{\nu+2}-1$
Total computational requirement per information bit for 3D TC	$3\times 2^{\nu+1}+2^{n+2}+3$	$3\times(2^{\nu+1}-1)$

Table 2.3: Computational complexity of the Max-Log-MAP algorithm. ν is the code memory and n is the total number of encoded bits at the decoder input, at each time step.

- ◇ In addition, $2k$ extrinsics (dual-port RAM) need to be stored (quantized on q_x+1 bits). For a 3D TC, additional extrinsics ($2\lambda k$) related to re-encoded redundancy bits need to be stored.
- ◇ Then, the hardware decision at the decoder output requires k memory bits (single-port RAM).
- ◇ Inside the SISO decoding processors, state metrics have to be stored at each iteration. The straightforward application of the Max-Log-MAP algorithm requires storing state metrics (either forward or backward). This can represent an unaffordable amount of memory for large k. In order to overcome this limitation, sliding window [10] processing can be implemented. The decoding length is then limited to a given truncated length TL rather than to the frame length k. This allows the overall decoding delay and the memory requirement to be reduced. Only $TL.2^{\nu}$ state metrics have to be stored. In practice, a window size equal to $TL=32$ represents a good trade-off between complexity and performance.

2.3.4 Summary

Table 2.4 compares the hardware complexity of the 3GPP2 turbo decoder and the corresponding 3D decoder when $\lambda = 1/8$ is used: $k=1530$ bits and $R=1/2$. Table 2.4 provides the complexity of the overall hardware dedicated to SISO decoding with the Max-Log-MAP algorithm in terms of add/compare-select operators; and the amount of RAM memory required for the implementation in terms of equivalent single-port RAM bit (we assume that one dual-port RAM bit is equivalent to two single-port RAM bits).

The number of SISO decoders placed in parallel, *Proc*, depends both on the required data throughput and on the hardware implementation technology. Table 2.4 presents complexity figures for $Proc=1$, $Proc=2$ and $Proc=4$.

	Overall SISO HW complexity (number of add / compare-select operators)		
	# *Proc* = 1	# *Proc* = 2	# *Proc* = 4
3GPP2	112 = C_1	224 = C_2	448 = C_4
3D TC based on 3GPP2	176 = C_1 + 57 %	304 = C_2 + 36 %	560 = C_4 + 25 %
	RAM (equivalent single-port memory in bits) Input quantization: 6 bits		
	# *Proc* = 1	# *Proc* = 2	# *Proc* = 4
3GPP2	101,510 = M_1	106,630 = M_2	126,251 = M_4
3D TC based on 3GPP2	= M_1 + 9.2 %	= M_2 + 8.8 %	= M_4 + 8.0 %

Table 2.4: Summary of complexity analysis for 3GPP2 and 3D turbo decoders for $k = 1530$ bits, $R = 1/2$ and $\lambda = 1/8$.

This complexity assessment does not take the size of the SISO internal operands into account. The implementation of the control part (state machines) and interleavers is not taken into account either. Note that the complexity of the state machines does not differ a lot between the different families of decoders. To conclude, the first estimation of the complexity increase in [20, 21] was optimistic. In fact, the more important the degree of parallelism, the less the impact in terms of relative additional complexity of using a 3D TC.

The authors in [69] provide a detailed comparison on the 3D decoder's complexity for a double binary TC. They consider the implementation complexity on FPGA and in 65nm ASIC technology. According to their approach, an additional complexity between 20% and 40%, depending on the implemented technique, is required for the 3D configuration compared to the classical turbo decoder. Their results are coherent with what we have obtained. This brings a complementary view of our analysis, dealing with computational complexity and memory requirements.

2.4 Conclusion

In this chapter, we presented a detailed study of the 3-dimensional turbo code. The 3D TC structure as well as the decoding process were discussed. This structure may be used in any receiver already using a turbo decoder, for example for the future generation of mobile TV, DVB-NGH. The main advantage is the potential reuse of hardware in mobile phones, where a 3GPP or 3GPP2 decoder is already available. Given the parent turbo code (the 3GPP2 turbo code in the case considered), the performance of the 3D TC depends on some key parameters. The interleaving law Π'(which permutes the parity bits before feeding them to the post-encoder) and the permeability rate λ have been properly

optimized. Besides, we discussed the different requirements that the post-encoder has to meet and how it is possible to choose a post-encoder by means of an EXIT analysis.

The most interesting property of the 3D TC is that it significantly improves performance in the error floor region with respect to the classical turbo code. Several upper bounds on the minimum distance of 3D binary turbo codes with 8-state upper and lower constituent encoders and 3GPP2 interleavers were presented. Besides, the different stages were illustrated with simulation results and asymptotical bounds. A thorough complexity analysis of the 3D decoder has been carried out in order to estimate the additional complexity. When high throughputs are required for a given application, several processors can be placed in parallel, which decreases the relative additional complexity of the 3D coding scheme.

In this chapter, the contributions of the thesis were the investigation of the interleaving law Π', the choice of a convenient post-encoder based on EXIT charts and the complexity analysis. The next chapter focuses on how to improve 3D TCs. The aim is to reduce the loss of convergence and to increase even more the distance especially in the case of high coding rates.

Chapter 3

Improving 3-Dimensional turbo codes

THE 3-dimensional turbo code provides improved asymptotic performance for a wide range of block lengths and coding rates, at the expense of an increase in complexity and a loss in convergence. In section 3.1, code optimization issues are discussed and we show that it is possible to improve even more the distance properties of the 3D TC by introducing an irregularity in the post-encoder permeability pattern. Then in section 3.2, we discuss convergence issues and we propose time varying 3-dimentional turbo codes as an alternative to reduce the observable loss of convergence. Finally, we analyse the association of 3D TCs with specific high order modulations in order to improve the performance of the 3D TC in the waterfall region.

3.1 Improving the asymptotic performance of the 3D turbo code

The use of 3D TCs results in an increase in the MHD with respect to classical TCs, except in the case of high coding rates (see 2.2). Two different study directions have been investigated in order to optimize 3D TCs: increasing even more the minimum Hamming distance and reducing the loss in convergence. In this section, we are interested in improving the asymptotic performance of the 3D TC especially for the 3GPP2 code family. In fact, we have observed that, for this turbo code, the error patterns with weight MHD often have very low multiplicities. This is mainly due to the tail bits since they represent singular points in the trellis and cause the codewords to be truncated. The first step involves eliminating these codewords by means of the adoption of a non regular post-encoding pattern. In subsection 3.1.1, the optimization method is detailed followed by two examples in 3.1.2 and 3.1.3.

3.1.1 Optimization method

To obtain the distance spectrum of the 3D TC, we apply the all-zero iterative decoding algorithm (see 1.4.3). We have observed in the distance spectrum that the first terms have a low multiplicity. The idea is to eliminate the corresponding codewords in order to increase the minimum distance d_{min}. Therefore, we have modified the pattern of post-encoding, which is no more regular, to generate more ones in the codeword with the lowest weight. Fig. 3.1 illustrates in a simple way the principle of the method explained above. The algorithm is the following:

1. Consider the codeword with the lowest weight.
2. Extract the addresses where the systematic bit x, the parity bit y or the post-encoded parity bit w is equal to one.
    a) If the systematic bit x is one and the corresponding parity bit y does not benefit from the regular post-encoding, include this address in the new post-encoding pattern.
    b) If the systematic bit x is zero, but the parity y is one, check whether the address of the parity bit benefits from the regular post-encoding. If not, include this address also in the new post-encoding pattern.
    c) If the post-encoded parity w is equal to one, do not modify the pattern for the corresponding address.
3. If there are several low weight codewords, go to step 2.
4. Finally, adapt the pattern of post-encoding in order to take into account the previous constraints. The addresses which will not any more benefit from the post-encoding are randomly selected. However, it is preferable to spread the modifications on all the length of the frame, not to discriminate a given region.

Note that each change of an address in the post-encoding pattern involves four exchanges of parity bits. Let us assume that N_{tot} is the total number of modifications necessary to generate the irregular pattern of post-encoding. Then, the distance of the code may be increased or in the worst case reduced by $4N_{tot}$.

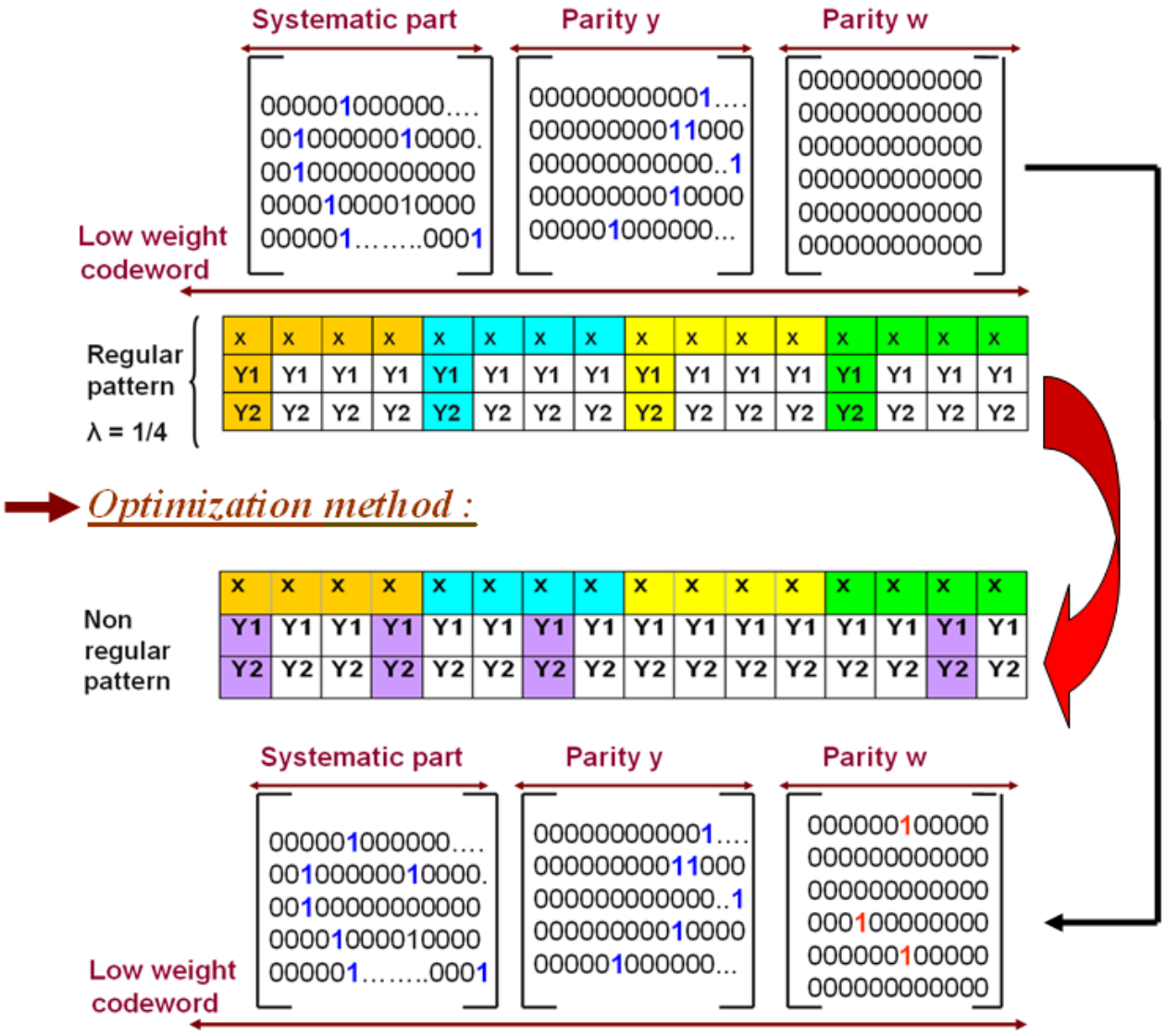


Figure 3.1: Illustration of the optimization method.

The following subsections 3.1.2 and 3.1.3 show the application of this optimization in two particular cases of the 3GPP2 3D code.

3.1.2 Example 1: optimization results for $k = 1530$ bits, $R = 1/2$ and $\lambda = 1/8$

Table 3.1 provides the first terms of the distance spectrum of the 3GPP2 3D TC obtained for $k = 1530$ bits. In this case, the post-encoding occurs regularly for the bits which address modulo 8 is equal to 1.

Table 3.1 shows that there is only one codeword with weight 18. We have noticed that there are four ones in the systematic part for the systematic bits at addresses {498, 512, 610 and 624}. There are also fourteen ones in the redundant part for the parity bits at addresses {350, 352, 356, 499, 501, 507, 511, 611, 613, 619, 623, 782, 784 and 788}. However, all the post-encoded bits are equal to zero. Therefore, we have changed the pattern of post-encoding to generate more ones and to eliminate the codeword with weight 18. Among the different possible patterns, we have finally introduced an irregularity in the previous pattern of post-encoding in order to postcode the bits at addresses {499, 501, 507, 511, 611, 613, 619 and 623} instead of {9, 81, 241, 401, 785, 961, 1121 and 1361}. We have chosen to spread our modifications on all the length of the frame, not to discriminate a given region.

Distance	18	20	21	22	23	24	25	26	27	28
Multiplicity	1	1	4	2	6	1	2	4	5	9

Table 3.1: First terms of the distance spectrum for a 3D TC where $k = 1530$ bits, $R = 1/2$ and $\lambda = 1/8$.

The codeword with weight 18 was eliminated. But, it was not possible to eliminate the codeword with weight 20 since one of the codewords with a just higher weight becomes a codeword with weight 20 each time we applied another possible pattern. Indeed, it was noticed that there are many common addresses containing ones in the different codewords of this distance spectrum. In other words, one codeword differ from the others by only few addresses. Several codewords, initially with high weight, appear in the new distance spectrum with a much lower weight which makes it difficult to find an irregular pattern of post-encoding that leads to a huge increase in the minimum distance. So, the search finally led to the increase of the minimum distance by 2 and the new distance of the optimized 3GPP2 3D TC is 20.

To conclude, the use of 3D TC results in an increase in the minimum Hamming distance d_{min} by more than 28 % for code rate $R = 1/2$, $\lambda = 1/8$ and $k = 1530$ bits (where k is the number of data bits), compared to the standardized 3GPP2 turbo code. Then, the optimization of the post-encoding pattern results in a total increase in d_{min} by more than 42 % (from $d_{min} = 14$ to $d_{min} = 20$) for code rate $R = 1/2$, $\lambda = 1/8$ and $k = 1530$ bits.

The FER performance of the 3GPP2 3D TC has been simulated with $\lambda = 1/8$. Then one bit out of eight is regularly picked from each of the parity streams starting with the first bit from each stream. For the 3D TC, the optimization of the post-encoding pattern provides a gain of 2.5 decades in the error floor compared with the classical TC as shown in Fig. 3.2 with the asymptotical bounds.

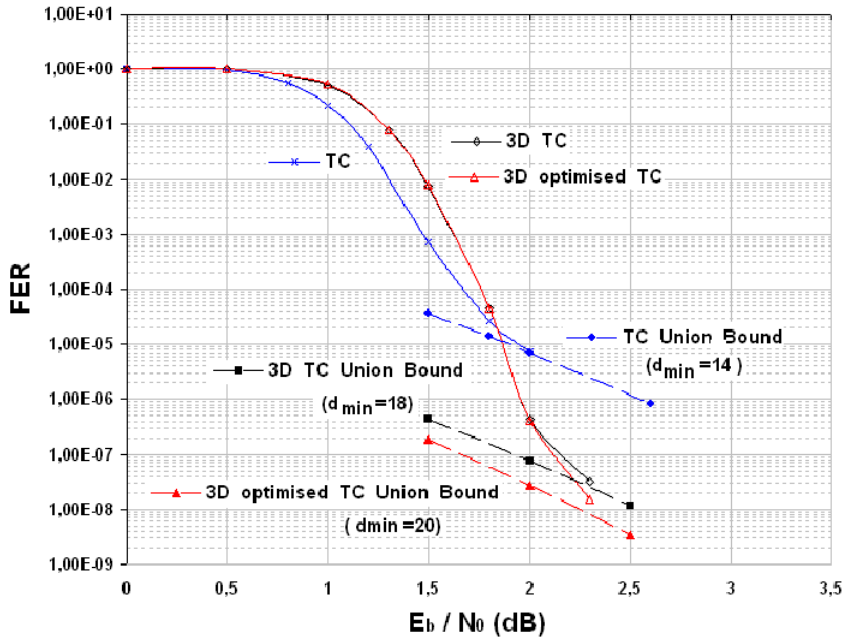


Figure 3.2: FER performance of the optimized 3D TC with $\lambda = 1/8$ for $k = 1530$ bits, $R = 1/2$ and comparison with the 3D TC and 3GPP2 standardized TC. All simulations use the Max-Log-MAP algorithm with 10 iterations.

3.1.3 Example 2: optimization results for $k = 1146$ bits, $R = 2/3$ and $\lambda = 1/4$

The same kind of optimization was performed with another frame length, $k = 1146$ bits, $R = 2/3$ and $\lambda = 1/4$. Table 3.2 provides the first terms of the distance spectrum that we have succeeded in detecting them by the application of all-zero iterative decoding algorithm.

Distance	12	15	21	27	47	48
Multiplicity	1	3	≥ 1	≥ 2	≥ 2	≥ 1

Table 3.2: First terms of the distance spectrum for a 3D TC where $k = 1146$ bits, $R = 2/3$ and $\lambda = 1/4$.

Table 3.2 shows that there is only one codeword with weight 12, three codewords with weight 15, at least one codeword with weight 21, at least two codewords with weight 27 and all the other codewords are with very high weight. In the codeword with the lowest weight (*i.e.* 12), we have noticed that the ones are concentrated in the systematic part at addresses {586, 587, 591, 650, 651,655, 763, 764, 768, 1019, 1020 and 1024}. All the parity bits y are equal to zero and do not benefit from the post-encoding which occurs regularly for the bits which address modulo 4 is equal to 1. To optimize the 3GPP2 3D TC, we have

slightly modified the permeability pattern in order to postcode the bits at addresses {586, 587, 650, 651, 763 and 764} instead of {9, 101, 581, 925, 1029 and 1133}. Also, we have chosen to spread our modifications on all the length of the frame, not to discriminate a given region. The idea was easier to implement, compared with $k = 1530$ bits, since there were few low weight codewords in the spectrum. The ones appear only in the systematic part of the codeword with weight 12, and the codewords in the distance spectrum are independant. That's why we have succeeded in eliminating all the codewords with lower weights at once.

The new minimum distance of the optimized 3D TC is 33 (see Fig. 3.3). This value has to be compared to 7 which is the distance of the standardized 3GPP2 TC. The use of optimized permeability patterns resulted in a huge increase in d_{min} for code rate $R = 2/3$, $\lambda = 1/4$ and $k = 1146$ bits. The spectrum has changed: some codewords disappeared and other codewords appeared with new distances ≥ 33.

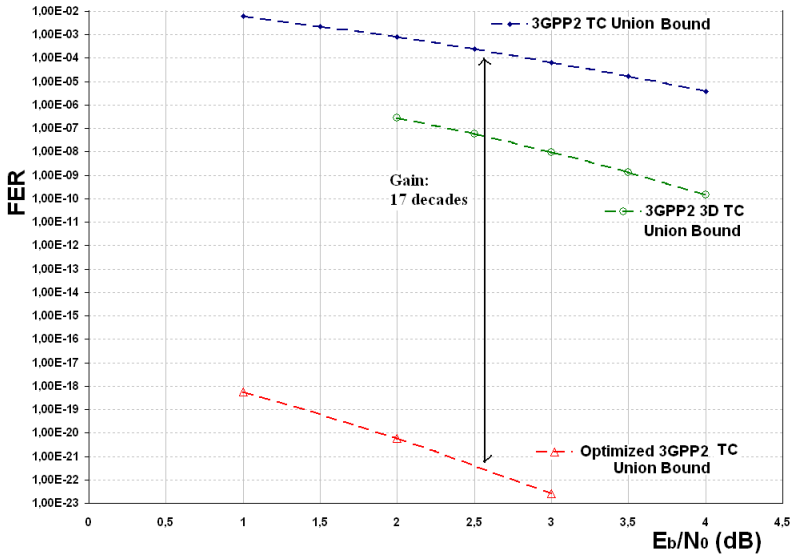


Figure 3.3: Asymptotical bounds of the optimized 3D TC, the 3D TC (with $\lambda = 1/4$) and the 3GPP2 classical TC for blocks of $k = 1146$ bits at coding rate $R = 2/3$.

Checking the asymptotic performance for $d_{min} = 33$ by Monte Carlo simulations turns out to be too time consuming when running the simulations on a classical PC computer. Thus, in this particular case, we have developped a specific technique to check that the new value of d_{min} is 33. In fact, we have exchanged six addresses in the pattern of post-encoding, which corresponds to 24 modifications in the parity bits. Thus, the worst case will be to loose 24 in the minimum Hamming distance. We have to consider all the codewords before optimization, to apply the new irregular pattern of post-encoding and to test whether the new weight is bigger than 33 or not. Furthermore, we have to do this verification only for the codewords with weight less than 57 before optimization. For codewords with weight more than 57 before optimization: even if we loose 24 in d_{min}, the distance remains higher than 33.

3.1.4 Conclusion

To conclude, a slight irregular pattern of post-encoding produces an improvement in the distance properties. These results are optimistic, and encourage to implementing the optimization method especially for high coding rates, in order to increase even more the MHD of the 3D turbo code.

Note that the optimization method presented above is applicable for any family of turbo codes provided that the distance spectrum has low multiplicities at the beginning, as for the 3GPP2 code family. If this opportunity arises, the modification of a few addresses in the pattern of post-encoding produces an improvement in the distance properties.

However, the method cannot be applied when an Almost Regular Permutation (ARP) is used like in WiMAX mobile communications. Since the bits distribution is periodic, the codewords multiplicity is at least equal to this period, *i.e*, ≥ 4. Also, better distances are obtained compared with the 3GPP2 code as a tail biting termination is used in this case.

3.2 Improving the convergence threshold of the 3D turbo code

The 3D TC improves performance in the error floor compared to the classical TC, at the expense of a loss in convergence and an increase in complexity. In this section, we first determine different convergence thresholds of the 3D TC for various coding rates and several values of λ. This allows the theoretical loss of convergence compared to classical TCs (*i.e.*, when $\lambda = 0$) to be calculated. Secondly, in order to reduce the observable loss in convergence at high error rates, the current post-encoder has been replaced by a time-varying trellis based encoder that allows the minimum distance of the post-encoder to be increased without augmenting error multiplication at its output. Finally, we show that there is no need to use the time varying technique when the code is associated with a high order modulation because the observable loss of convergence is removed when a specific permutation before the mapping is used.

3.2.1 Convergence threshold of binary 3D turbo codes

To generate the EXIT chart (see 1.4.2) of a TC, we have to consider the transfer characteristics of the extrinsic information for each SISO decoder. In the case of 3D TC, the two SISO decoders exchange extrinsic information about the systematic part of the received codeword, like for classical turbo decoding. But both of them exchange also extrinsic information about the post-encoded parity bits with the 4-state SISO pre-decoder. And we have to take into account in the EXIT chart that the extrinsic information about these parity bits is changing from an iteration to the other. In fact, the input MI changes each iteration as the predecoder feeds the two SISO decoders with new extrinsic information about the post-encoded parity bits. Consequently, the curves of mutual information exchange between the two decoders change every iteration.

For example, the EXIT charts of the classical turbo code and the corresponding 3D turbo code with parameters $R = \frac{2}{3}$ and $\lambda = \frac{1}{8}$ at an $\frac{E_b}{N_0}$ of 1.49 dB and 1.55 dB, respectively, are depicted in Fig. 3.4. Note that the curves remain almost unchanged after the fourth iteration. In both cases of Fig. 3.4, the tunnel between the two EXIT curves opens, predicting convergence thresholds around these values. These results were confirmed by the simulations of the code.

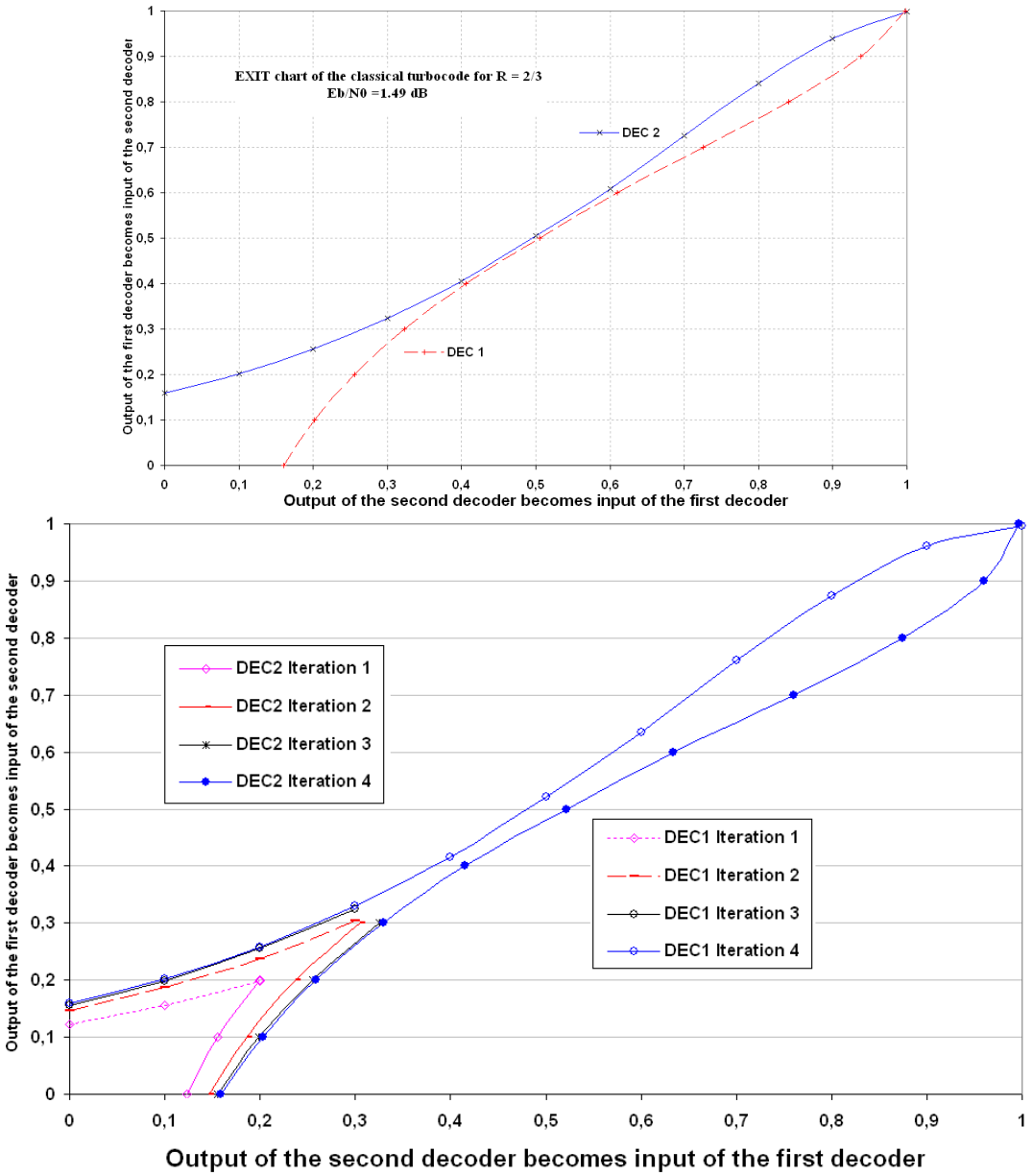


Figure 3.4: EXIT charts of the classical turbo code at $^{E_b}/_{N_0} = 1.49$ dB for code rate $R = {}^2/_3$ and the corresponding 3D TC with $\lambda = {}^1/_8$ at $^{E_b}/_{N_0} = 1.55$ dB.

Note:

While analyzing the 3D Exit charts, it was observed that the curves remain almost unchanged after a certain number of iterations, depending on the considered code rate. In the future, it is possible to further explore these EXIT charts. In fact, I noticed that the EXIT charts of the post-encoder and the classical TC intersect, meaning that the post-encoder is no more useful after a certain number of iterations. Consequently, a new 3D decoding process can be proposed where the predecoder is activated for only few iterations. Afterwards, the classical turbo decoder returns in position. This decoder is also interesting from a consumption point of view and allows the latency of the decoding process to be reduced.

In Fig. 3.5, we have plotted the FER performance of the 3D 3GPP2 TC to compare it with that of the 3GPP2 TC. As expected, we can observe a loss in convergence around 0.06 dB in the waterfall region. Compared with the classical TC , this value corresponds to the loss of the convergence threshold previously predicted by the EXIT charts for the 3-dimensional TC at coding rate $R = \frac{2}{3}$ and $\lambda = \frac{1}{8}$: 1.55 dB -1.49 dB. On the other hand, the low error rate performance is noticeably improved: a gain of 0.8 dB at FER $= 10^{-5}$.

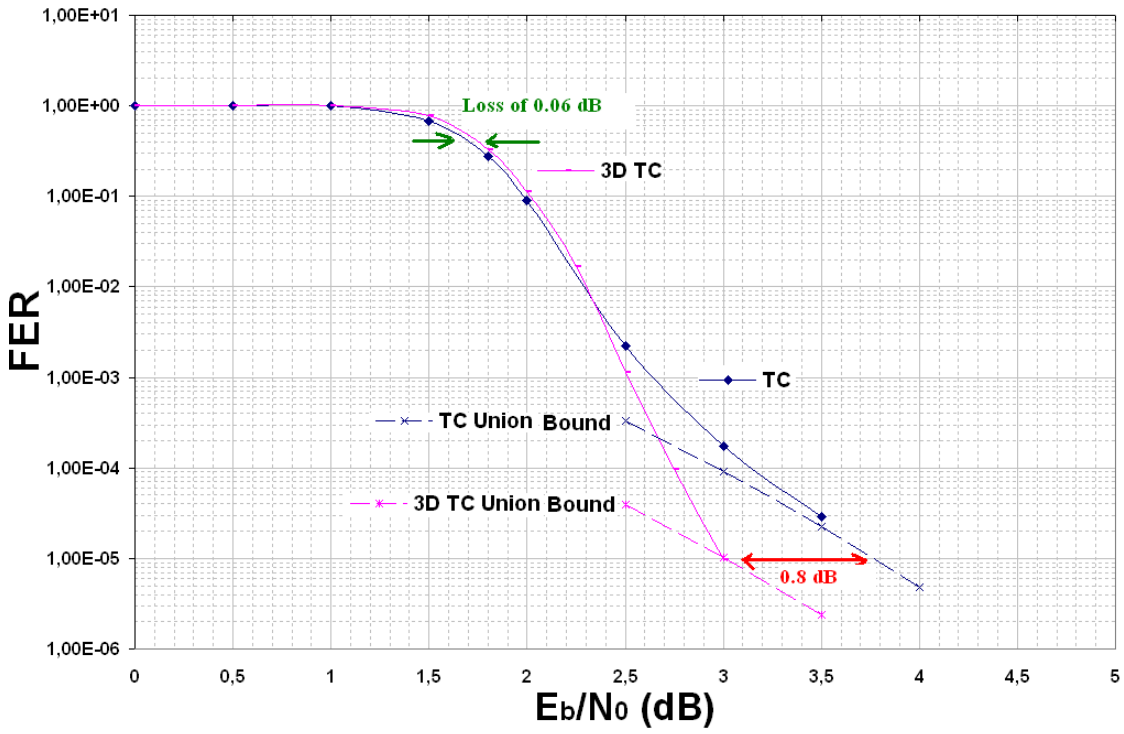


Figure 3.5: FER performance of the 3D 3GPP2 TC with $\lambda = 1/8$ for $k = 1146$ bits, $R = 2/3$ and comparison with the 3GPP2 TC. All simulations use the MAP algorithm with 10 iterations.

In the sequel, different convergence thresholds of 3GPP2 3D turbo codes are estimated through an EXIT chart analysis over the additive white Gaussian noise channel and over the Rayleigh fading channel as well.

Transmission over the AWGN channel:

Different convergence thresholds of 3GPP2 3D turbo codes, for several code rates and values of λ, are given in Table 3.3. As expected, for a fixed code rate R, the best convergence threshold is achieved by the classical turbo code ($\lambda = 0$), while increasing λ leads to poorer thresholds. In other words, the loss in the convergence threshold increases with λ.

	$R = \frac{1}{3}$	$R = \frac{1}{2}$	$R = \frac{2}{3}$	$R = \frac{4}{5}$
$\lambda = 1/4$	0.19 dB	0.82 dB	1.67 dB	2.58 dB
$\lambda = 1/8$	0.08 dB	0.73 dB	1.55 dB	2.45 dB
$\lambda = 0$	−0.07 dB	0.60 dB	1.49 dB	2.44 dB

Table 3.3: Convergence thresholds of 3GPP2 3D turbo codes over the AWGN channel.

Transmission over the Rayleigh fading channel:

The convergence thresholds of 3D turbo codes are also estimated through the same kind of EXIT chart analysis over the Rayleigh fading channel [103, 106]. In Table 3.4, we show the convergence thresholds of the 3D TCs for transmissions over the Rayleigh fading channel and the Gaussian channel at coding rate $R = \frac{1}{2}$ and $\lambda = 1/8$. Table 3.4 shows that the loss of convergence is more significant over the Rayleigh fading channel. Many simulations have been carried out to confirm this observation.

Note:

With error correcting coding and transmission over a slow flat Rayleigh fading channel, the diversity order equals the minimum Hamming distance of the code [6].

Convergence threshold	**Gaussian channel**	**Rayleigh fading channel**
TC ($R = 1/2$)	0.6	2.54
3D TC ($R = 1/2$, $\lambda = 1/8$)	0.73	2.73
Loss in convergence threshold	0.13 **dB**	0.19 **dB**

Table 3.4: Convergence thresholds of 3GPP2 3D turbo codes over the Rayleigh fading channel and comparison with the Gaussian channel.

This result represents a major drawback of 3D turbo codes. In fact, for applications such as terrestrial mobile radio communications where the simulations are carried out over fading channels, the loss of convergence is higher. However, the 3D turbo codes are

adapted to transmissions over Gaussian channels. Thus, they can be used in applications such as fixed satellite communications. In subsections 3.2.2 and 3.2.3, we propose two techniques to remedy these convergence issues.

3.2.2 Time varying 3D turbo codes

So far, we have only considered *fixed* or *time invariant* (TI) convolutional codes. Time varying (TV) convolutional encoders have generator polynomials which periodically vary with time. The idea appeared for the first time in the paper of Costello [33]. The author conjectured that non-systematic time varying convolutional codes have larger free distances, d_{free}, than fixed convolutional codes with the same rate and constraint length. This conjecture led to the search for periodic time varying (PTV) convolutional codes that are better than TI convolutional codes with respect to d_{free}. Little progress on the construction of such time-varying codes was reported in [76], where Mooser tried but was not able to find a periodic code with a larger d_{free} than any fixed code at coding rate $R = \frac{1}{2}$ and for a constraint length $\nu = 4$. By means of computer-aided search, Lee found in [68] some $d_{free} = 8$ PTV convolutional codes while randomly searching for good period-4 codes with the same set of parameters (*i.e.*, at coding rate $R = \frac{1}{2}$ and for a constraint length $\nu = 4$). Furthermore, Palazzo discovered a period-2 rate 2/3 memory 1 PTV convolutional code with $d_{free} = 4$ which is larger than $d_{free} = 3$ of best rate 2/3 TIC codes [79]. After an exhaustive search over PTV convolutional codes, Hu and Pérez found in [62] three period-2 rate 1/2 memory 7 PTV convolutional codes with d_{free} larger than any TI convolutional codes with that rate and memory. Based on the previous studies concerning time varying convolutional codes, we decided to investigate a simple time varying post-encoder in order to increase locally its minimum distance. This should improve the level of the extrinsic information provided by the predecoder to the two SISO decoders, which will hopefully reduce the observable loss of convergence threshold for 3D TCs.

3.2.2.1 Choice of the time varying post-encoder

In 2.1.4.1, we justified the choice of the post-encoder by means of EXIT analysis. Code (7,4) of Fig. 2.5 (b) was discarded since the 3D TC has the worst behaviour in the waterfall region with this post-encoder. Code (5,4) has a good performance in the waterfall region, whereas code (5,7) has the highest local minimum distance. In fact, the EXIT curves in Fig. 2.6 corresponding to the post-encoders of Fig. 2.5 (a) and 2.5 (c) cross around input MI 0.1. For high input MI the curve related to code (c) indicates a better behaviour in the error floor region. For this reason, the first idea is to combine the two encoders and this mixture results in a time varying post-encoder.

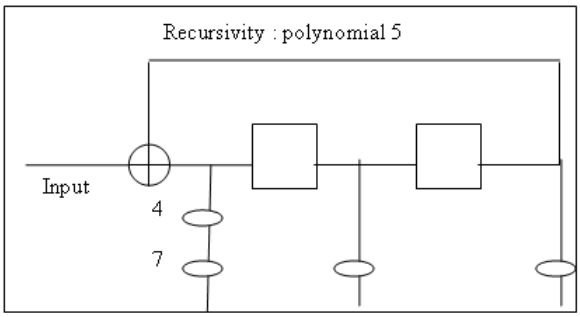


Figure 3.6: 4-state post-encoder with time-varying parity construction (5, 4 :7).

As shown in Fig. 3.6, the proposed time varying encoding technique consists in alternating two redundancies in time $W_1 = 4$ and $W_2 = 7$, instead of having only one. If we look at the trellis of the code (5,4:7) depicted in Fig. 3.7, we can easily identify two different paths corresponding to the all–zero sequence. So the decoder will not be able to distinguish between them and the distance of the code is only 2, compared with 3 for the RSC code (5,4). To avoid this problem, the idea proposed in [19] was to replace some redundancies 4 by other redundancies 7 to get closer to the code (5,7). The replacement period is denoted by L. Fig. 3.8 illustrates the principle of this technique.

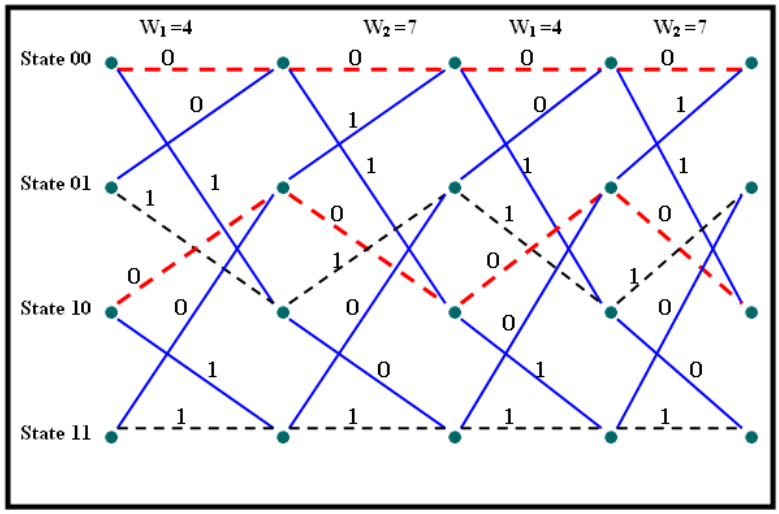


Figure 3.7: Trellis of the post-encoder (5, 4 :7). Redundancies 4 and 7 are alternated in time.

The following analysis allows us to determine the value of L. First, code (5,4) has a corresponding decoder which only doubles the number of errors of its input at the first iteration :

$$BER_{out} = 2\,BER_{in}$$

where BER_{in} is the channel error rate. Using a time varying post encoder increases the

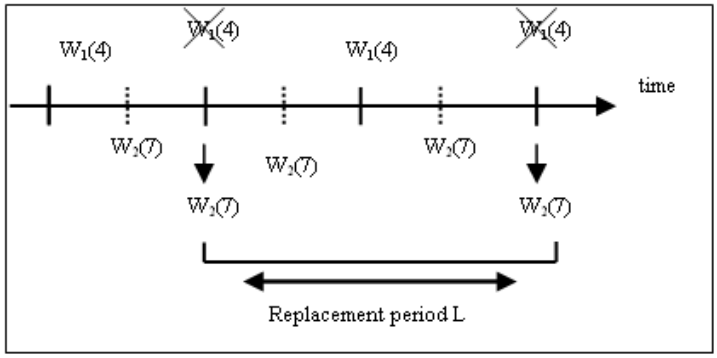


Figure 3.8: Modified time-varying post-encoding with a replacement period of L.

BER at its output. The BER at the first step is then expressed in the following way :

$$BER_{out,TV} = 2\ (BER_{in} + \xi) \tag{3.1}$$

where ξ is an additinal error rate at the output of the pre-decoder at the first step of the iterative process. On the other side, we have by definition :

$$BER_{out,TV} = \frac{\text{Number of erroneous bits for TV TC}}{\text{Number of blocks}} \times \frac{1}{k}$$

Since we generate three errors each time we replace a redundancy [19], we obtain :

$$BER_{out,TV} = \frac{\text{Number of erroneous bits without TV} + 3 \times \text{Number of replacements}}{\text{Number of blocks} \times k}$$

$$= BER_{out,\,without\,TV} + \frac{3 \times \text{Number of replacements}}{\text{Number of blocks} \times k}$$

$$= 2\,BER_{in} + \frac{3 \times \text{Number of replacements}}{\text{Number of blocks} \times k} \tag{3.2}$$

If we compare the last expression (3.2) to the first one (3.1), we can identify :

$$2 \times \xi = \frac{3 \times \text{Number of replacements}}{\text{Number of blocks} \times k}$$

Finally :

$$\xi = \frac{3}{2} \frac{\text{Number of replacements per block}}{k}$$

The number of replacements per block is equal to $\textbf{ceil}\,[\frac{(k-3)}{L}] + 1$. The term $k-3$ is due to the first replacement which occurs for the third parity (*ie.* instead of alternating $W_1, W_2, W_1, W_2, W_1, \ldots$, we will have $W_1, W_2, W_2 \ldots$).

At the end, a given bloc size k can be considered. The evolution of ξ according to L, given in Fig. 3.9, results from the formula:

$$\xi = \frac{3}{2} \times \frac{\mathbf{ceil}\left[\frac{(k-3)}{L}\right] + 1}{k}$$

Fig. 3.9 shows that the error rate ξ decreases when L increases. ξ influences the loss of convergence threshold of 3D TCs, whereas L is related to the distance properties. The asymptotic performance of a time varying post-encoder is better for small L values because the number of the redundancies W_2 exceeds that of the redundancies W_1. In this case, the time varying encoder is closer to the code (5,7) which has the higher distance, that is 5. However, from the convergence point of view, Fig. 3.9 shows that it is better to choose L high in order to reduce the additive error rate ξ. In fact, high values of L mean that the TV post-encoder gets closer to the code (5,4) which has a better behaviour in the waterfall region (see 2.1.4.1). Thus, the optimal value of L is a convergence/distance trade-off.

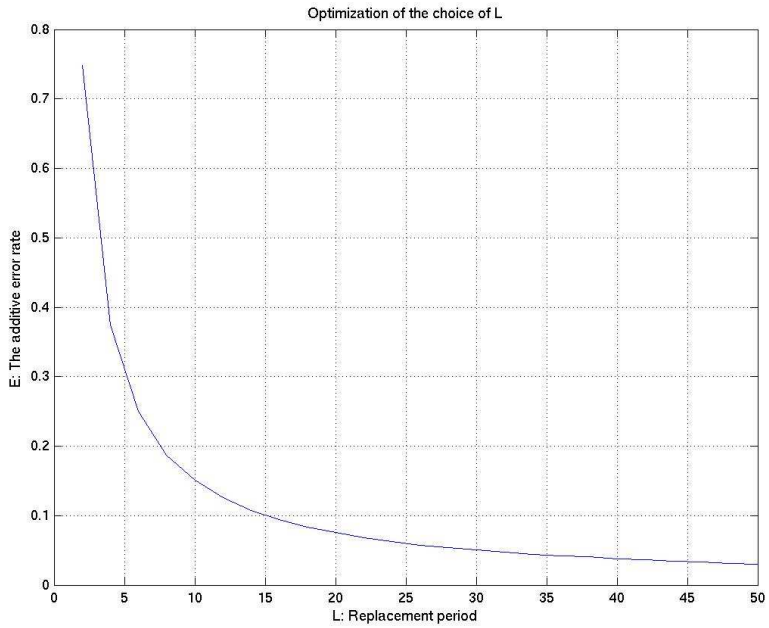


Figure 3.9: Estimation of ξ according to L for blocks of $k = 1530$ bits. ξ is an additional error rate at the output of the predecoder at the first iteration.

For the 3D TC, the loss in convergence is more significant for low code rates (such as for $R = 1/3$ or $R = 1/2$). In this case, we privilege a high value of $L = 30$ to limit the additional error rate ξ as much as possible. For the same block size and the same permeability rate λ, we privilege a small value of $L = 10$ for high code rates such as for $R = 4/5$.

3.2.2.2 EXIT analysis of time varying 3D turbo codes

Similarly to the EXIT of a 3D TC, we have plotted the EXIT of a time varying 3D TC. For the post encoder, two redundancies $W_1 = 4$ and $W_2 = 7$ are alternated in time, but W_1 is replaced by W_2 for all the periods of L. Therefore, the transfer characteristics of the two decoders are no more symmetric.

Fig. 3.10 shows an EXIT chart of a 3-dimensional time varying turbo code for $E_b/N_0 =$ 1.58 dB. Note that the transfer characteristics are asymmetric and remain almost unchanged after the seventh iteration. For $E_b/N_0 = 1.58$ dB, the tunnel between the EXIT curves is open, and the exchange of the extrinsic information continues along the iterations until we reach the intersection point (1,1). Consequently, the convergence threshold of the 3D TV turbo code is 1.58 dB for code rate $R = 2/3$ and $\lambda = 1/4$.

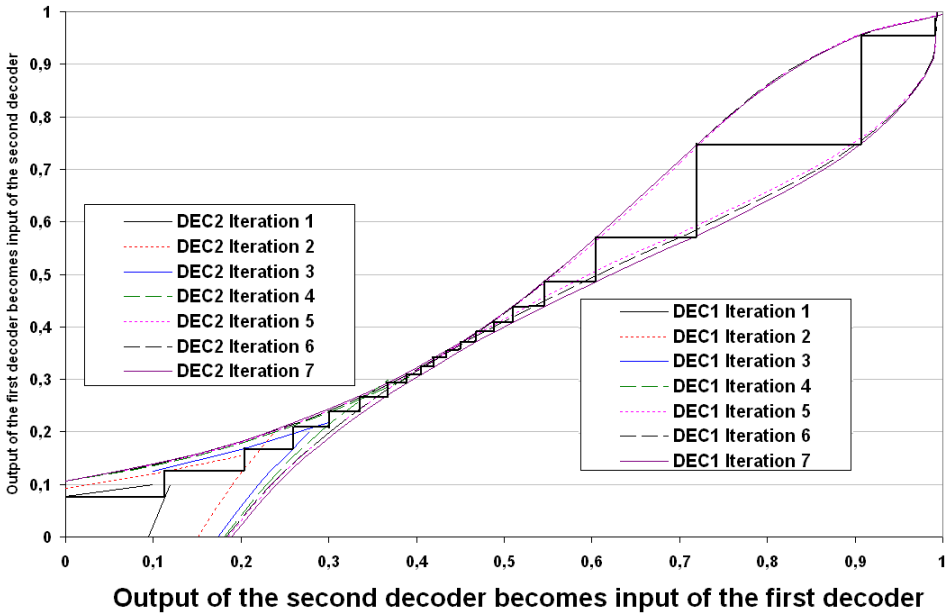


Figure 3.10: EXIT chart of the time varying 3D TC at code rate $R = 2/3$, $\lambda = 1/4$, $L = 30$ and $E_b/N_0 = 1.58$ dB.

On the other hand, the convergence threshold of the TC at code rate $R = 2/3$ is estimated around 1.49 dB and that of the 3D TC is about 1.67 dB (see Table 3.3). As a conclusion, the EXIT analysis shows that the use of time varying 3-dimensional 3GPP2 TC reduced the loss of convergence by 50 % from 0.18 dB $(1.67 - 1.49)$ to 0.09 dB $(1.58 - 1.49)$ at code rate $R = 2/3$ and $\lambda = 1/4$. These results were confirmed by simulations of the code, as shown in Fig. 3.11.

In addition, we have observed that the EXIT of a time varying 3D TC changes with the value of L. A perspective would be to use this tool to choose an optimal value of L. The selected value can be the L that produces the lowest convergence threshold.

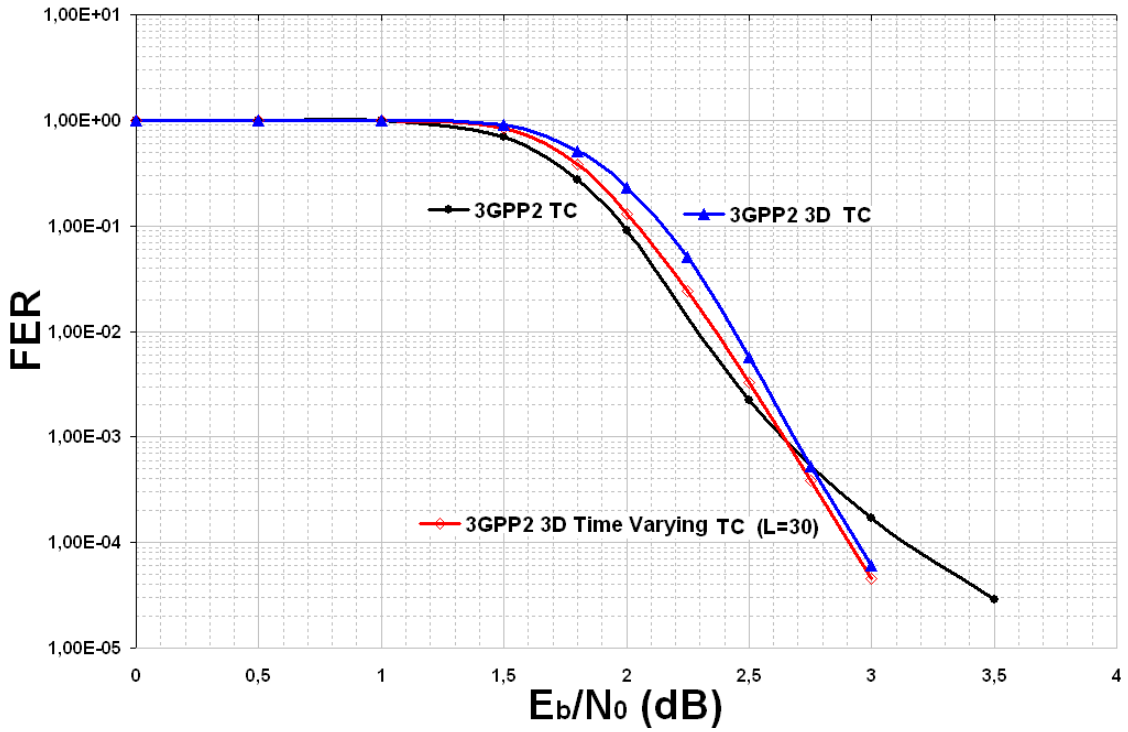


Figure 3.11: FER performance of the time varying 3D 3GPP2 TC with $\lambda = 1/4$ for $k = 1146$ bits, $R = 2/3$ and comparison with the 3GPP2 3D TC and 3GPP2 TC. All simulations use the MAP algorithm with 10 iterations.

3.2.2.3 Error rate performance of time varying 3D turbo codes

Simulations have been carried out in order to check the results predicted by the EXIT charts.

Example 1: simulation over the Gaussian channel

The FER performance of the time varying 3GPP2 3D TC has been simulated with $\lambda = 1/4$ and $L = 30$. Then one bit out of four is regularly picked from each of the parity streams starting with the first bit from each stream. For the post encoder, two redundancies $W_1 = 4$ and $W_2 = 7$ are alternated in time, but W_1 is replaced by W_2 for all the periods of $L = 30$. Fig. 3.11 shows that the use of time varying 3-dimensional 3GPP2 TC reduced the loss of convergence by 50% from 0.18 dB to 0.09 dB for $k = 1146$ bits at code rate $R = 2/3$ and $\lambda = 1/4$, compared with the 3GPP2 3D TC.

Example 2: simulation over the Rayleigh fading channel

Time varying 3GPP2 3D TCs have also been simulated over a Rayleigh fading channel. Fig. 3.12 shows the BER performance for blocks of 2298 bits at code rate $R = 1/3$ and $\lambda = 1/8$. In this case, the observable loss of convergence is reduced by 35% from 0.23 dB to 0.15 dB.

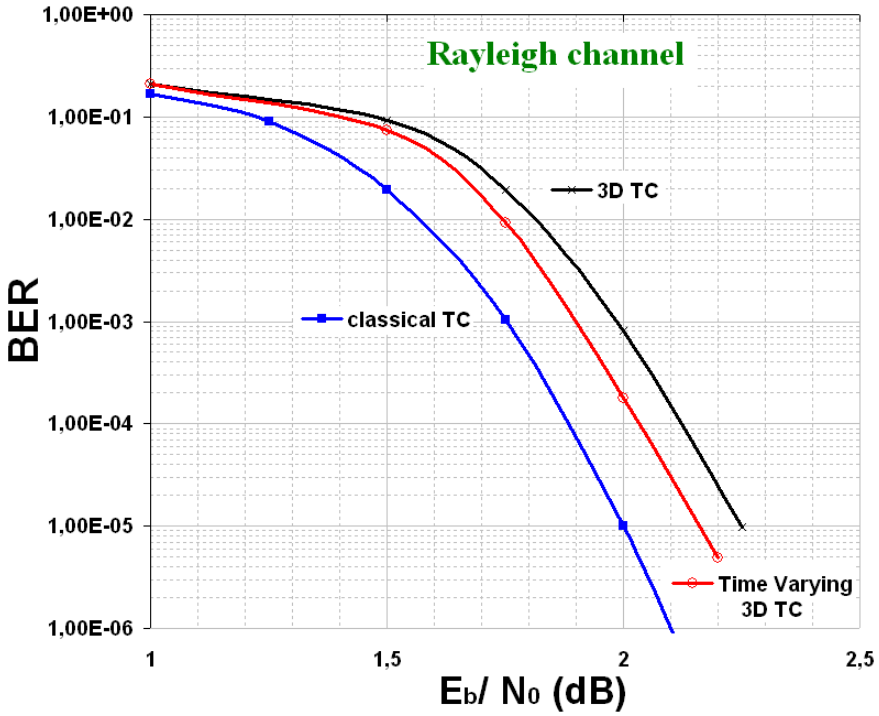


Figure 3.12: BER performance of the time varying 3GPP2 3D TC with $\lambda = 1/8$ for $k = 2298$ bits, $R = 1/3$ and comparison with the 3GPP2 3D TC and 3GPP2 TC. All simulations use the Max-Log-MAP algorithm with 10 iterations.

Other simulations have been carried out for different block sizes and coding rates of the 3GPP2 turbo code. Among the simulated cases, it was observed that the time varying parity construction reduces the observable loss of convergence by 10% to 50% of the value expressed in dB. We have systematically checked that the asymptotic performance is not degraded. In fact, the choice of the post-encoder does not influence a lot the minimum distance of the 3D TC for a fixed code memory and for a given permeability rate λ. For the time varying post-encoder, there is no reason to improve or degrade the asymptotic performance since we introduce few local modifications. However, the higher the local minimum distance of the post-encoder, the better the level of the extrinsic information which the predecoder supplies to the two SISO decoders. Therefore, the TV technique acts as a convergence accelerator of the 3D TC.

3.2.3 3D turbo codes for high spectral efficiency transmissions

Usually, when a code is associated with a modulation, the choice of the bits placed in the best protected binary positions of the modulation scheme influences the convergence threshold. What about 3D TCs? For this purpose, we have also investigated the association of the 3D TC with high order modulations. This structure is used for applications where high data throughputs are required such as the transmission of HD TV. In the most recent transmission systems, high bitrates require using high order modulations such as 16-QAM for 3GPP2, 64-QAM for LTE, 256-QAM for DVB-NGH, etc.

3.2.3.1 Transmission scheme

We consider the coded modulation scheme depicted in Fig. 3.13, based on the so-called pragmatic or BICM approach [28]. Fig. 3.13 shows a turbo encoder and a modulator that follows an interleaver and a Gray mapper. It is known that among the bits forming a Gray-labeled symbol in M-QAM or M-PSK modulations for $M > 4$, the average probability of error is not the same for all the bits. In fact, some bits forming the symbol have a smaller average probability of error than the others [82]. Therefore, three constellation mappings, all compliant with Gray labelling, were investigated. First, the mapping is uniformly distributed on the entire constellation. In a second configuration, the systematic bits are mapped to better protected places as a priority, and all the other bits are uniformly distributed. For the third mapping, the systematic bits are first mapped to better protected places as a priority. Then, if it is possible, the post-encoded parity bits are better protected by the considered modulation than the other non re-encoded parity bits. This choice is made because the systematic bits as well as the post-encoded parity bits are used by both decoders during the decoding process. Thus, protecting them is expected to reduce the loss of convergence. At the receiver side, the demapper computes the Log Likelihood Ratio (LLR) related to each bit of the information sequence. This symbol-to-bit LLR calculation is followed by a 3D turbo decoder using the MAP algorithm. The following subsections show the application of the proposed transmission scheme in some particular cases of the 3GPP2 3D code.

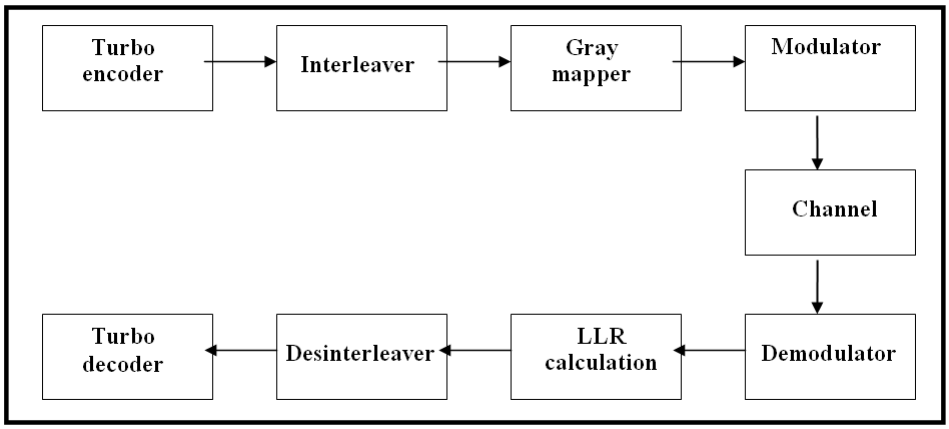


Figure 3.13: Transmission scheme.

3.2.3.2 Example 1: 3D TCs associated with a 16-QAM modulator for $k = 2298$ bits, $R = 1/3$ and $\lambda = 1/8$

Transmission over the Gaussian channel

Among the four bits forming a symbol with Gray labelling in 16-QAM, the average probability of error is smaller for the first and the third bits than for the second and fourth bits. Details can be found in Appendix C. To explore this property in the context of

3.2.3.1, FER performance of 3GPP2 3D TCs has been simulated with $\lambda = \frac{1}{8}$ at code rate $R = \frac{1}{3}$ for $k = 2298$ bits, as shown in Fig. 3.14.

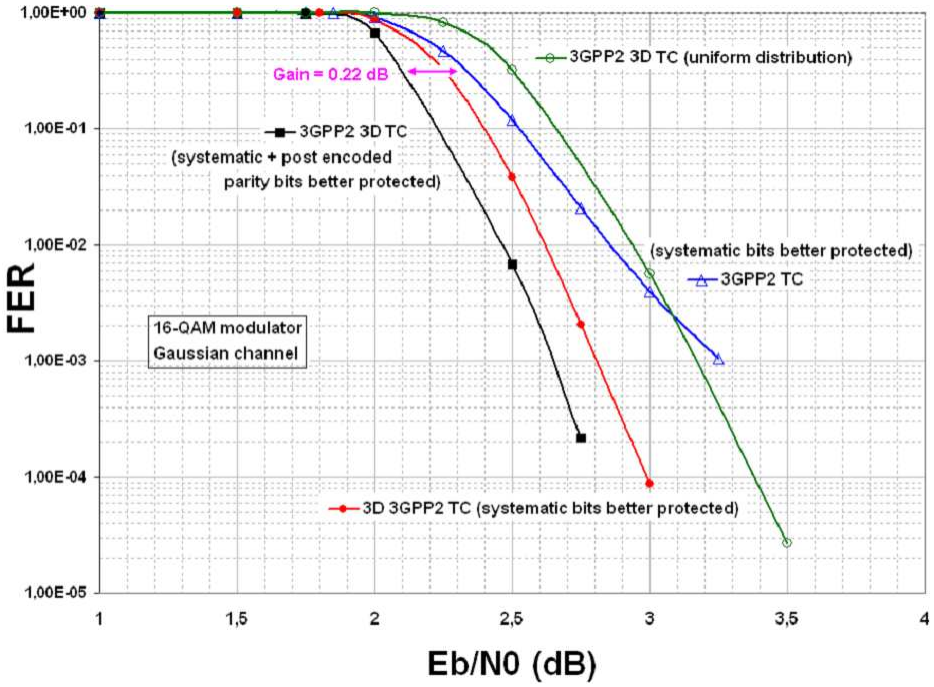
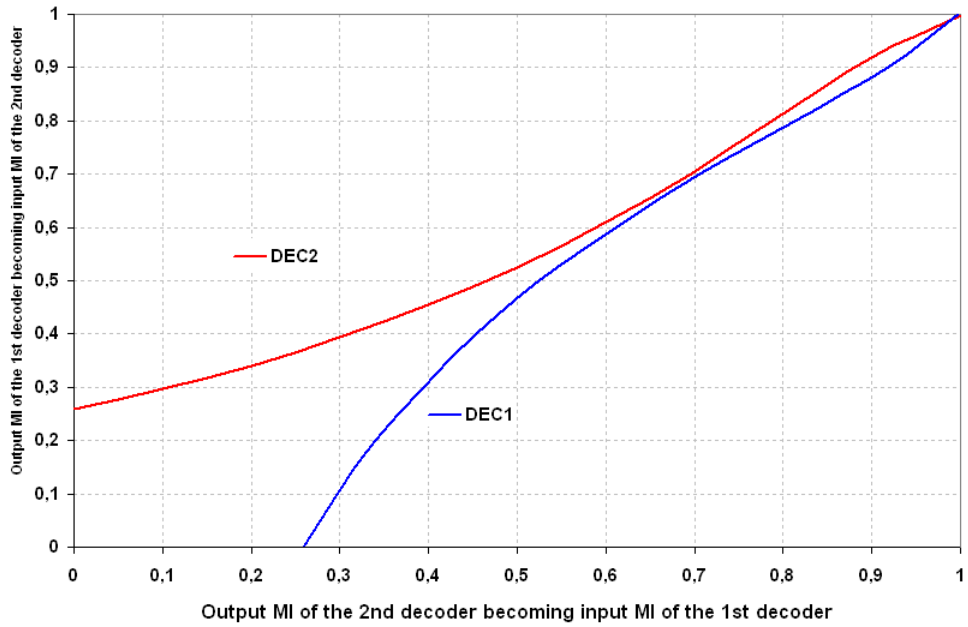


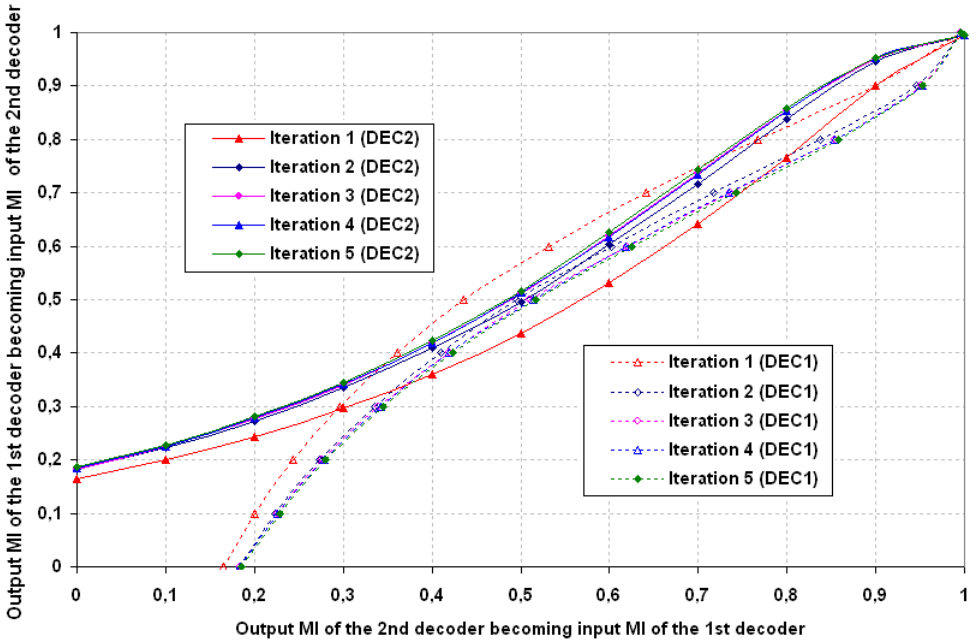
Figure 3.14: FER performance of the 3GPP2 3D TC with $\lambda = 1/8$ for $k = 2298$ bits, $R = 1/3$ and comparison with the 3GPP2 TC. All simulations use the MAP algorithm with 10 decoding iterations and 16-QAM modulation. Transmission is over the Gaussian channel.

It was observed through these simulations that the use of the 16-QAM modulation, where the systematic bits as well as the post-encoded parity bits are better protected than the other non re-encoded parity bits, allows the loss of convergence of the 3GPP2 3D TC to be reduced and even be transformed into a gain in the waterfall region of 0.22 dB compared with the standardized 3GPP2 TC. Whereas, for a QPSK modulation, the convergence loss of the 3D TC was estimated to 0.15 dB at code rate $R = \frac{1}{3}$ and $\lambda = \frac{1}{8}$.

These simulation results were confirmed by an EXIT chart analysis. In fact, Fig. 3.15a shows that the convergence threshold of a TC with 16-QAM modulation at code rate $R = \frac{1}{3}$ is 2.19 dB. Besides, the convergence threshold of the 3D TC with 16-QAM modulation, where the systematic bits and the post-encoded parity bits are protected as a priority, is 1.97 dB at code rate $R = \frac{1}{3}$ and $\lambda = \frac{1}{8}$ (see Fig. 3.15b). This confirms the observed gain of 0.22 dB in the waterfall region.



(a) EXIT charts of an 8-state TC for $E_b/N_0 = 2.19$ dB.



(b) EXIT chart of a 3D TC for $E_b/N_0 = 1.97$ dB and $\lambda = 1/8$. Note that after the third iteration, the transfer characteristics remain almost unchanged.

Figure 3.15: Comparison of the convergence thresholds of a classical turbo code with a 3D TC at code rate $R = 1/3$. A 16-QAM Gray mapping is used.

Transmission over the Rayleigh fading channel

Similar simulations over the Rayleigh fading channel, for the same block size $k = 2298$ at code rate $R = \frac{1}{3}$ and $\lambda = \frac{1}{8}$, are reported in Fig. 3.16. Here also, the loss of convergence

is transformed into a gain in the waterfall region compared to the classical TC, as shown in Fig. 3.16.

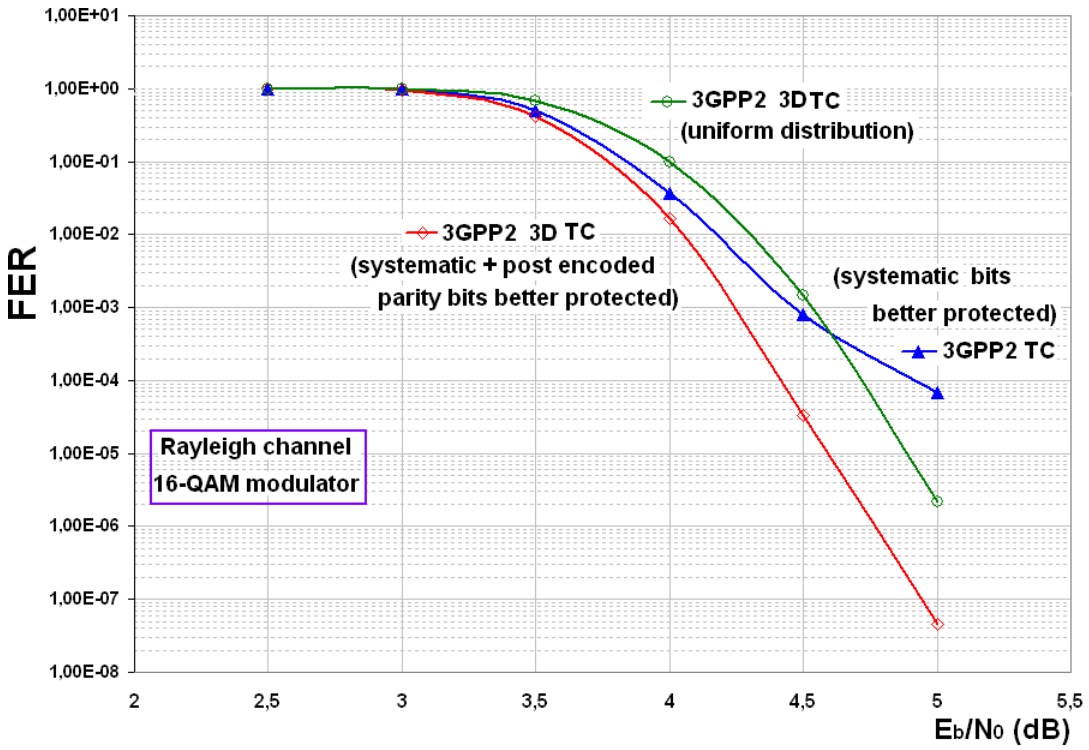


Figure 3.16: FER performance of the 3GPP2 3D TC with $\lambda = 1/8$ for $k = 2298$ bits, $R = 1/3$ and comparison with the 3GPP2 TC. All simulations use the MAP algorithm with 10 decoding iterations and 16-QAM modulation. Transmission is over the Rayleigh fading channel.

This important result can be generalized for transmissions over Gaussian and fading channels: 3D TCs associated with a 16-QAM modulation, where the systematic bits as well as the post-encoded parity bits are more protected than the other non re-encoded parity bits, allows the loss of convergence threshold to be removed and a gain is observed at all SNRs.

3.2.3.3 Example 2: 3D TCs associated with an 8-PSK modulator for $k = 1146$ bits, $R = 4/5$ and $\lambda = 1/8$

Among the three bits forming a symbol with Gray labelling in 8-PSK, the average probability of error is smaller for the first and the second bits than for the third bit. Details can also be found in Appendix C. We have simulated the FER performance of 3D TCs with $\lambda = \frac{1}{8}$ at code rate $R = \frac{4}{5}$ for $k = 1146$ bits. Transmission over a Rayleigh fading channel is considered. For this coding rate, the third configuration cannot be adopted and

we have to implement the second configuration[1]. Simulations in Fig. 3.17 show that that the use of an 8-PSK modulation, where the systematic bits are protected as a priority, allows the loss of convergence of the 3GPP2 3D TC to be transformed into a gain in the waterfall region of 0.5 dB compared with the 3GPP2 standardized TC. This gain is more significant than the one obtained with 16-QAM in the previous section.

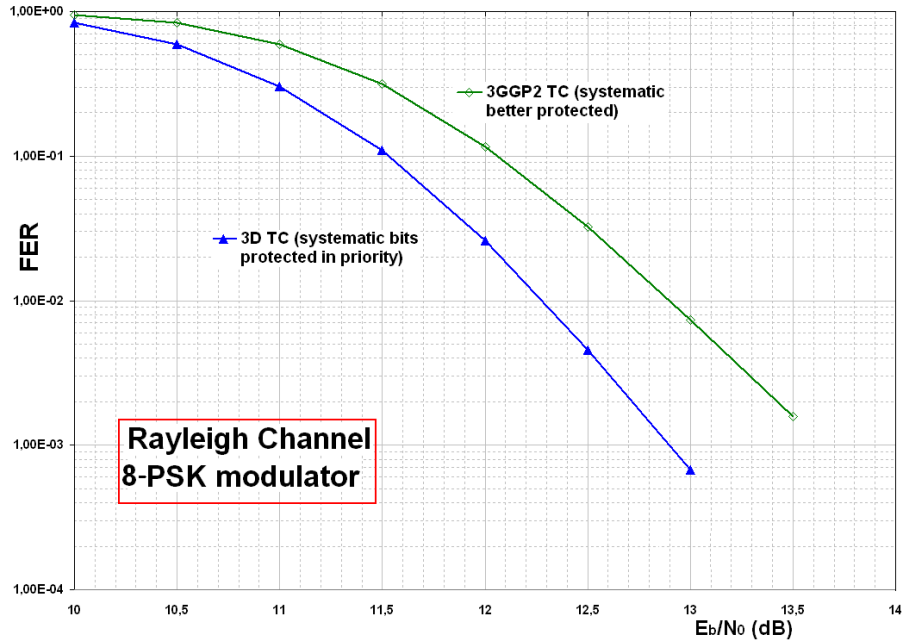


Figure 3.17: FER performance of the 3GPP2 3D TC with $\lambda = 1/8$ for $k = 1146$ bits, $R = 4/5$ and comparison with the 3GPP2 TC. All simulations use the MAP algorithm with 10 decoding iterations and an 8-PSK modulation. Transmission is over the Rayleigh fading channel.

3.2.3.4 Design rules for 3D turbo coded modulations

The investigation of the previous three Gray mappings allows some design rules to be defined. In fact, a loss of convergence is still observed when the bits in a symbol are uniformly distributed on the entire constellation. However, when the second configuration or the last one is used, the loss in the convergence threshold disappears and a gain in the waterfall region is observed. When these both configurations can be implemented, the configuration 3 must be used as far as possible. Otherwise, it is necessary to implement at least the configuration 2, as in the example presented above for the 8-PSK.

When the systematic bits are first protected as a priority and, if possible, the post-encoded bits, a significant gain is obtained. The technique is all the more interesting as

[1]In this case, 573 8-PSK symbols are required to transmit 1146 systematic bits, 288 post-encoded parity bits, 143 redundancies y_1 as well as 143 redundancies y_2. Note that the systematic bits are placed in the first two bits of all the symbols (because $2 \times 573 = 1146$). Consequently, each parity is transmitted in the third bit of a given symbol.

the coding rates are higher for the same value of λ, even for transmissions over fading channels. In other words, the problem related to the loss in the convergence threshold of 3D TCs can be solved.

3.3 Conclusion

It is naturally desirable to have turbo codes which have waterfalls as closest as possible to the channel capacity and low error floors. Therefore, this chapter deals with performance improvement of 3D TCs. One of my contributions is the optimization method introduced in order to increase even more the minimum Hamming distance of the 3D TC. Then, convergence issues are also discussed in this chapter. One major drawback of the 3D TC structure is that the loss of convergence is more significant for fading channels compared with Gaussian channels. It seems all the more necessary to find a solution to this problem. Another contribution consists in time varying 3-dimentional turbo codes that are proposed as an alternative to reduce the observable loss of convergence without degrading the asymptotic performance. This technique allows reducing the loss of convergence by 10% to 50% of the value expressed in dB. Furthermore, we analyze the association of 3D TCs with specific high order modulations to improve the performance of the 3D TC in the waterfall region. This contribution shows that when the code is associated with high order modulations, there is no need to use a time varying trellis and a specific mapping allows obtaining even a gain in the waterfall region. Therefore, the 3D TC is adapted to be used in high spectral efficiency transmission schemes.

This chapter represents an important part of my research work dealing with the performance improvement of a 3-dimensional turbo code based on the partial concatenation of the 3GPP2 code with a rate-1 post-encoder. I show that it is possible to build 3GPP2 3-dimensional turbo codes which have good performance in both regions. Performance comparisons are made between the 3GPP2 standardized turbo code and the corresponding 3D code. The different stages are illustrated with simulation results, asymptotical bounds and EXIT charts. This code structure is expected to reach a performance/complexity trade-off never yet attained.

To conclude, we notice that the notion of irregularity plays a major role to build 3D turbo codes having good performance at low and high SNRs. In fact, the optimization method is based on the use of a non regular pattern of post-encoding and allows the minimum distance to be increased. Besides, the time varying post-encoder (5, 4:7) with a little irregularity and also the association of the 3-dimensional turbo code with high order modulations, where both the systematic bits and the post-encoded parity bits are more protected than the other parity bits create a sort of irregularity in the Gray mapping. Both represent a success in reducing or even eliminating the problem of convergence loss. The next step of the study concerns the investigation of irregular turbo codes. The aim is to obtain an irregular TC which performs well in both the waterfall and the error floor regions.

Chapter 4

Irregular turbo codes

Low Density Parity Check (LDPC) codes [53], first proposed by Gallager in 1963, and later rediscovered by MacKay and Neal [72, 73], have been of great interest recently. Like turbo codes, LDPC codes can achieve near Shannon limit error performance [86, 88], and represent a very promising prospect for error control coding. Work on *irregular* LDPC codes has shown that by making the codeword bits participate in varying numbers of parity check equations, significant coding gain can be reached [71, 74, 85]. For code length equal to $n = 10^5$, Richardson *et al.* [86] proposed irregular LDPC codes that perform better than the original rate-$1/2$ turbo code. Their simulation results showed that irregular LDPC codes of length one million achieved a bit error probability equal to 10^{-6} less than 0.13 dB away from capacity.

Two years earlier, Frey *et al.* [51] had already introduced irregularity to turbo codes in order to achieve better performance. Very interesting results were found for code rates equal to $R = 1/3$ and $R = 1/2$. For example, by making the original rate-$1/2$ turbo code of Berrou *et al.* slightly irregular, Frey *et al.* obtained a coding gain of 0.23 dB (compared with the original code) at a block length of $n = 131,072$, bringing the irregular turbo code within 0.25 dB of capacity. However, an error floor can be observed at a bit error rate higher than 10^{-4}. To our knowledge, only one reference [92] deals with the problem of lowering the error floor of irregular TCs: Sawaya *et al.* introduced symbol-based iterative decoding. Their results show that the error floor in the performance of the rate-$1/2$ irregular TC proposed by Frey *et al.* in [51] is lowered significantly and appears at a bit error probability equal to 6×10^{-6} instead of 4×10^{-4} with code length $n = 131,072$ and 100 decoding iterations.

In this chapter, we begin by reviewing the basics of irregular turbo codes in section 4.1. Then, in section 4.2, a new method using the EXIT diagrams allows the search for good *degree profiles* to be simplified. In section 4.3, we design and simulate irregular turbo coding schemes with suitable interleavers in order to improve their distance properties. Finally, we propose a modified encoding procedure in section 4.4. The aim is to obtain irregular turbo codes which perform better than regular turbo codes in both the waterfall and the error floor regions.

4.1 Basics of irregular turbo codes

4.1.1 Another representation of turbo codes

In most cases, the two (or more) constituent RSC encoders of a parallel turbo code are identical. For this reason, the authors in [25, 26] proposed a "self-concatenated" turbo encoder depicted in Fig. 4.1. It consists of the concatenation of a repetition code and an RSC code separated by an interleaver. In fact, it is possible to merge the two trellis encoders and replace the initial interleaver with a double size interleaver preceeded by a 2-fold repetition (d-fold repetition in general). The interest of this second equivalent encoding structure of a classical TC lies in its simplicity and in the opportunity of introducing an irregular structure. By adopting the terminology used for the LDPC codes, a *regular TC* is related to the use of a uniform repetition in the equivalent encoding structure (see Fig. 4.1). On the other side, when the repetition degree d is not identical for all the information bits, the TC is said to be *irregular*.

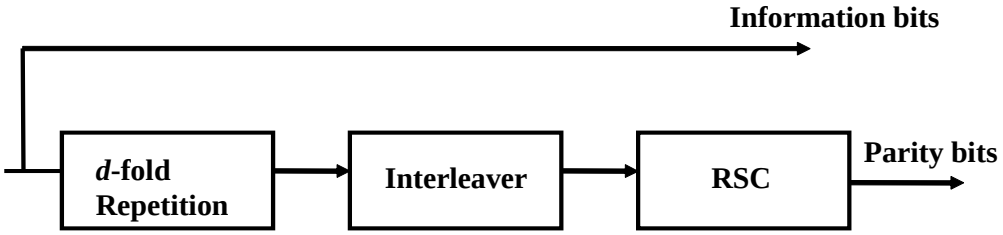


Figure 4.1: Equivalent encoding structure for a self-concatenated turbo encoder.

The classical turbo code corresponds to the case where all the information bits have the same degree $d = 2$. The performance of this regular turbo code is identical to the one we get using the standard turbo encoder when the interleaver length is sufficiently high. For short and medium block sizes, a few additional iterations are necessary to achieve the same performance as for the classical parallel turbo code. This is due to the decoding process because the extrinsic information is not available at the first iteration. It is possible to implement for example the shuffled iterative decoding [77, 114] in order to achieve the same performance as for the classical turbo encoder and avoid adding iterations in practice. In shuffled decoding, the decoder updates the extrinsic information as soon as possible, without waiting for all the copies of a given data to be processed. In other words, the decoder does not wait until the next iteration to send extrinsic messages. However, as it does not represent a major problem, we have chosen to increase the number of iterations in our simulations by two additional iterations for the classical sequential decoding.

For instance, the BER performance of the 3GPP2 turbo code has been simulated for blocks of 2298 bits at coding rate $R = 1/3$. Fig. 4.2 shows that the use of the equivalent encoding structure for a regular turbo code requires two additional iterations (*i.e.* twelve instead of ten) to reach the performance of the classical parallel turbo code.

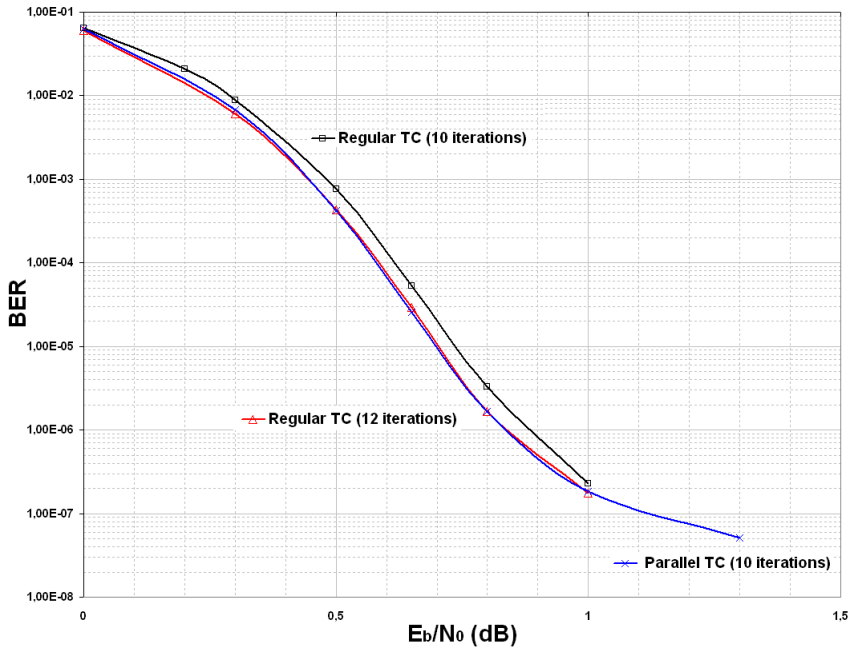


Figure 4.2: BER performance of two equivalent encoding structures for a regular 3GPP2 turbo code. All simulations use the Max-Log-MAP algorithm for blocks of 2298 bits and coding rate $R = 1/3$.

4.1.2 Irregular turbo encoder

The general structure of an irregular turbo encoder is given in Fig. 4.3. Note that the irregular turbo code implemented in Fig. 4.3 is a generalization of Repeat-Accumulate-Codes presented in [42]. It consists of the cascade of a non-uniform repetition, an interleaver and an RSC code. In fact, the introduced irregularity makes it possible to improve the performance of a turbo code by inserting some bits inside the RSC trellis with a degree $d > 2$, called elite or pilot bits. However, making the code irregular entails an increase in the rates of the constituent codes. Therefore, for usual coding rate values, only a small fraction of information bits is repeated $d > 2$ times, *e.g.*, $d = 8$ to avoid increasing the number of low weight codewords. Thanks to their higher degree, the pilot bits include d (*i.e.* eight in the example) extrinsics instead of two, and are thus extremely well protected. However, increasing the degree d of the pilot bits is a two-fold weapon. When an error occurs, it is amplified and propagated all around which multiplies the errors. This phenomenon is called the *correlation effect*.

To construct an irregular turbo code, Fig. 4.3 shows that we need to divide the information bits into classes, each class j having a specific degree $d_j = j$. The fraction of bits of degree d_j in a class j is denoted by f_j, where $f_j \in [0, 1]$. A degree profile consists of all the degrees d_j and their corresponding non-zero fractions f_j. In the sequel, we represent a degree profile by the vector $(f_2, f_3, ..., f_{max})$ or the vector $(2, 3, ..., max)$.

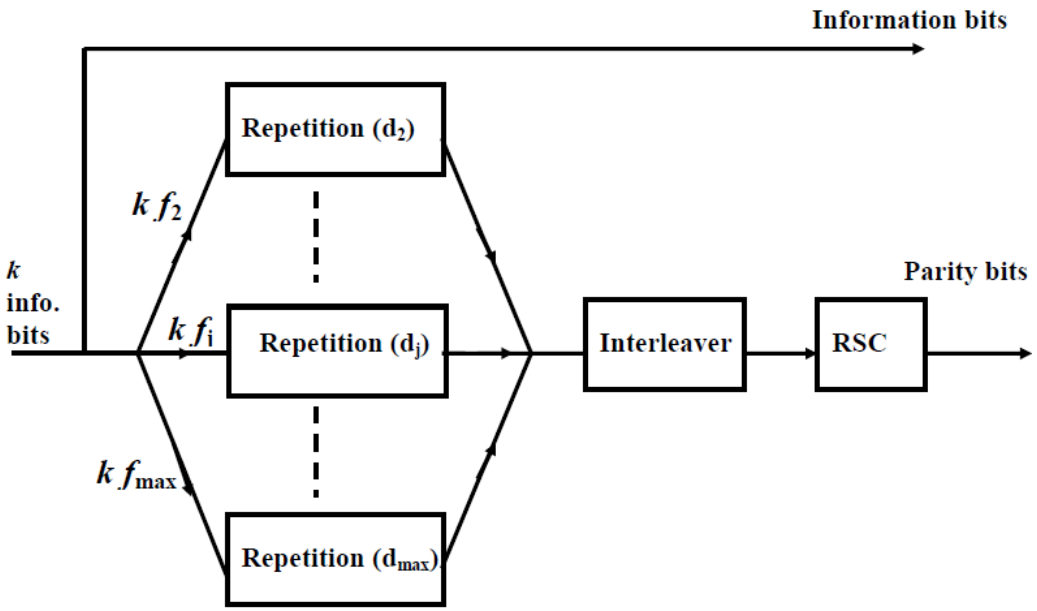


Figure 4.3: Encoding scheme of an irregular turbo code.

The average information bits degree is:

$$d_{Average} = \sum_{j=2}^{max} d_j \, f_j$$

If $d_j < 2$, there is no means to exchange extrinsic information. Thus, we keep the minimum degree equal to two in order to refer to the classical parallel turbo code. The maximum degree is d_{max}. The following simple relations are satisfied by irregular turbo code parameters:

$$\sum_{j=2}^{max} f_j = 1 \quad \text{and} \quad n = \sum_{j=2}^{max} d_j \, f_j k = k \times d_{Average} \tag{4.1}$$

where k denotes the total number of information bits and n is the interleaver size. Let R be the code rate of the RSC code. There are $k\, d_{Average}$ bits at the input of the constituent code, and the number of parity bits is:

$$\frac{1}{R} k \, d_{Average} - k \, d_{Average} = k \, d_{Average} \left(\frac{1}{R} - 1 \right) \tag{4.2}$$

Now, the number of parity bits for the RSC code is also equal to the one for the irregular turbo code:

$$n - k = \frac{1}{R_{Irregular}} k - k = k \left(\frac{1}{R_{Irregular}} - 1 \right) \tag{4.3}$$

where $R_{Irregular}$ is the rate of the irregular turbo code and it can be expressed from equations (4.2) and (4.3) as:

$$R_{Irregular} = \frac{1}{1 + \left(\frac{1}{R} - 1\right)\, d_{Average}}$$

We can also deduce another formula of the average degree:

$$d_{Average} = \frac{\frac{1}{R_{Irregular}} - 1}{\frac{1}{R} - 1} \tag{4.4}$$

Example

Let us analyse the construction of a rate-1/4 turbo code. We assume that the average degree of information bits is equal to $d_{Average} = 3$. If we consider a degree profile where only the degree $d = 3$ has a non-zero fraction f_3, the TC is regular. To make the code irregular, different configurations with limited number of irregularity classes are taken into account, namely the initial class of degree $d = 2$ and only one or two classes of high degree bits. Let us examine a degree profile where $d = 2$, $d = 3$ and $d = 4$ have non-zero fractions. We compute the corresponding fractions from equations (4.1) and we obtain: $f_2 = f_3 = f_4 = \frac{1}{3}$.

To increase the overall code rate of the irregular TC, a percentage of the parity bits must be punctured. A high percentage of puncturing will cause the minimum distance of the code to decrease significantly and the error floor may appear at high bit error probability. Therefore, if we have to choose between different degree profiles, it is preferable to select only those which do not require any puncturing.

4.1.3 Irregular turbo decoder

To decode irregular turbo codes, only one SISO decoder is employed since there is only one RSC constituent code. Fig. 4.4 shows a block diagram of the decoder.

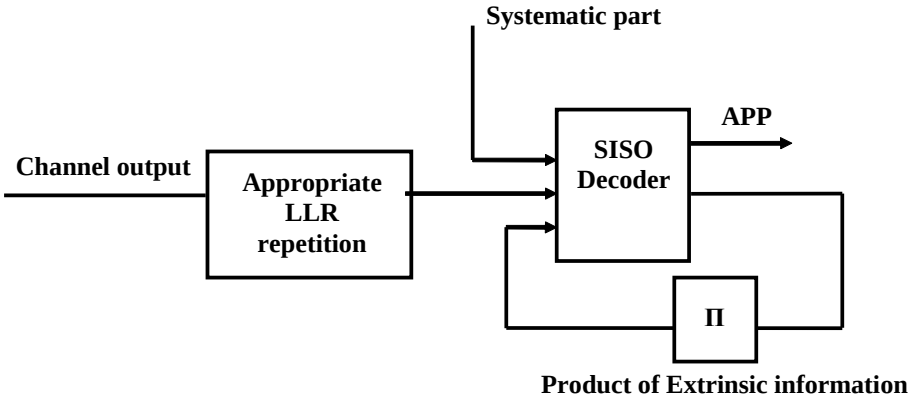


Figure 4.4: Iterative decoding principle of a self-concatenated code.

First, the decoder computes the channel output Log-Likelihood Ratios (LLRs) for all the coded bits, and applies an appropriate repetition to each LLR. If the coded bit has

degree d, then the corresponding likelihood is repeated d times. Next, the LLRs are fed to the SISO decoder. As the considered code is systematic, the decoder receives also information about the systematic part. At the first iteration, the *a priori* information probability is fixed to 1/2. The BCJR algorithm for the convolutional code can start and the forward-backward algorithm computes the *a posteriori* probability. Here again, if the coded bit has degree d, the algorithm produces d extrinsic probabilities or LLRs. For instance, a regular turbo decoder with $d_{Average} = 3$ computes three extrinsics for each degree-3 bit. In general, the extrinsic information corresponding to each bit is the product of the $d-1$ extrinsic pieces of information. Let $r_{i,1}$, $r_{i,2}$, ..., $r_{i,d}$ be the d repetitions of the current coded bit. In the probability domain, the extrinsic information can be expressed as:

$$Ext(r_{i,j}) = \prod_{k=1,k\neq j}^{d} Ext(r_{i,k})$$

The decoding process continues until convergence is reached or a maximum number of iterations have been performed. For irregular turbo codes, the number of iterations needed for convergence is significantly greater than the one we have for regular turbo codes. An explanation to this phenomenon can be found by analyzing the convergence of iterative decoding in both cases. For irregular turbo codes, the number of iterations can be increased in order to get closer to the theoretical limit. This was largely discussed in [94], where Sawaya studied an irregular TC with 100 decoding iterations instead of 20 iterations for the regular TC. Nevertheless, about ten iterations are enough to run simulations and one major advantage of irregular TCs is that better performance can be reached for both finite and infinite code length.

4.2 Selecting the degree profile of irregular turbo codes

The convergence threshold as well as the asymptotic performance of an irregular turbo code strongly depends on the degree profile. The degree profile depends on the interleaver and the generator polynomials of the RSC code. For codes with very large block lengths, the optimization of the previous parameters can be done using the density evolution method developed by Richardson and Urbranke [87, 88]. Using this approach, irregular LDPC codes with performance at 0.0045 dB from the capacity were obtained [29]. The Gaussian approximation can be used to speed up the search for good parameters. This sub-optimal method leads to quite accurate results and was defined in several different ways [30, 49, 93, 104]. A Gaussian approximation based on extrinsic information transfer functions was first introduced by Ten Brink [104]. Gamal and Hammons, the authors in [49], introduced a similar Gaussian approximation based on signal-to-noise ratio matching (SNRM-GA). To analyze the convergence of iterative decoding, Sawaya *et al.* [93] introduced the Gaussian approximation with error probablity matching (EPM-GA).

Although the density evolution method and the Gaussian approximation approach can be used to select a good degree profile for codes with large block sizes, Monte Carlo

simulations of the bit error probability are usually carried on for finite length. The method consists in fixing a degree d_{Irreg} and varying its fraction f_{Irreg}. A fraction that achieves the best performance can be found. The next step involves changing the degree d_{Irreg}, while the fraction f_{Irreg} is fixed to the value already selected. We can then find optimal values for both d_{Irreg} and f_{Irreg}. However, this profile is not automatically the best one, since the optimization does not take into account all the possible combinations (d_{Irreg} , f_{Irreg}). Also, better performance may be attained when the profile is not restricted to two non-zero fractions. This method was used for irregular turbo codes [52]. The main drawback is that Monte Carlo simulations are time consuming. An innovative method, based on the EXIT diagrams to select a good profile without resorting to extensive and long simulations, is discussed in subsection 4.2.1.

In our approach we separate the problems. First, we search for the best degree profile using a random interleaver. Afterwards, we optimize the interleaver. In addition, the number of degrees and fractions of a profile is equal to $2\,(d_{max} - 1)$. The main difficulty consists in having only two equations to optimize all these parameters:

$$\sum_{j=2}^{max} f_j = 1 \tag{4.5}$$

$$d_{Average} = \sum_{j=2}^{max} d_j\, f_j \tag{4.6}$$

Therefore, we choose few non-zero fractions. In the sequel, we just choose two non-zero fractions: one for the degree $d = 2$ and another one for $d_{Irreg} > 2$ in order to boost the performance of the iterative decoder by the insertion of pilot bits [105]. Since we focus on the optimization of irregular turbo codes with $d_{Average} = 3$, equations (4.5) and (4.6) become:

$$f_2 + f_{Irreg} = 1$$

$$2\, f_2 + d_{Irreg}\, f_{Irreg} = d_{Average} = 3$$

The number of parameters is then reduced to only three parameters: f_2, f_{Irreg} and d_{Irreg}. Also, only degree profiles without any puncturing are selected within our analysis.

4.2.1 Determination of the degree profile using hierarchical EXIT charts

For TCs, EXIT charts are usually used to find the best component codes. Like the density evolution method for LDPC codes, we decided to use the EXIT tool for irregular TCs in order to select a good degree profile without resorting to long and extensive simulations. For very large block lengths, we have observed that all the curves merge with one another and we cannot distinguish between the different degree profiles. However, we can distinguish between all these curves in practice when simulations of the error rate performance are carried out. Therefore, an idea was to use the EXIT method for finite

block sizes. In this case, we have observed that the EXIT tool provides a hierarchy between the different degree profiles. The only difference with a classical EXIT chart is that the interleaver has a finite length. We keep the hypothesis that the extrinsic information messages are independent and identically distributed. Indeed, this hypothesis does not raise any problem since we aim at comparing different degree profiles, not at computing accurate convergence thresholds. This new method using the EXIT diagrams is simple and allows many degree profiles to be compared at the same time. Besides, we have plotted the EXIT diagrams for different values of E_b/N_0. We have observed that the curves get closer to each other as E_b/N_0 increases. However, the same hierarchy between these curves is obtained as the value of E_b/N_0 varies. In order to identify easily the different curves and analyze their behaviour, it is preferable to choose a low value of E_b/N_0. Thus, comparisons between different degree profiles in terms of convergence behaviour as well as asymptotic behaviour are made and the selection of the best degree profile can be progressively refined.

Let be a rate-1/4 irregular TC, where the constituent code is an RSC code of unity rate with octal generators $g = (13, 15)$. We consider different degree profiles with two non-zero fractions and we plot the corresponding EXIT diagrams of an irregular TC in Fig. 4.5 for a given block length. First, the convergence threshold of an irregular turbo code is lower when the output mutual information (OMI) is high for a zero input mutual information (IMI). We notice that the regular turbo code has the lowest output mutual information. Thus, it will perform worse than all the other irregular turbo codes in terms of convergence threshold. Besides, the OMI increases with the IMI. However, the faster the EXIT curve reaches the point of coordinates (1,1), the better the behaviour of the code at high SNRs is. This point materializes the perfect knowledge of the received message.

Fig. 4.5 shows that some curves are lower than the others for high values of IMI. This observation is related to an important correlation effect when the pilot bits are of very high degree. In this case, the repeated bits are not spread enough after the interleaver: this has an impact on the minimum distance and predicts a floor at high error rates. The EXIT diagrams of the irregular turbo code where the degree profile is (f_2, f_{15}) or (f_2, f_{20}) highlight the phenomenon of the correlation effect and predict a bad behaviour at high SNRs. As usual, there is a convergence/distance trade-off to choose a degree profile.

If we compare two different degree profiles such as (f_2, f_8) and (f_2, f_4). Fig. 4.5 shows that the irregular TC with degree profile (f_2, f_4) reaches the point of coordinates (1,1) for lower values of IMI than those for irregular TC with degree profile (f_2, f_8), which predicts a better behaviour in the error floor. From a convergence point of view, the irregular TC with degree profile (f_2, f_8) produces a higher OMI than the irregular TC with degree profile (f_2, f_4) for a zero IMI , *i.e.* OMI $= 0.7$ compared to OMI $= 0.35$ respectively which predicts a gain in the convergence threshold. In the next subsection 4.2.2, the behaviour of these irregular turbo codes and the various observations through EXIT diagrams are validated by simulations.

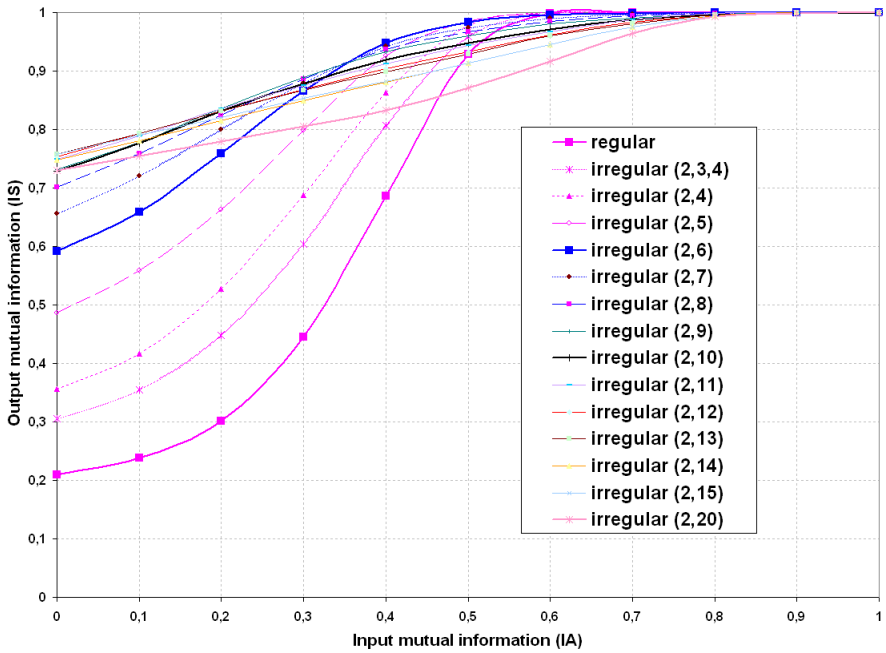


Figure 4.5: EXIT diagrams of irregular turbo codes with different degree profiles for blocks of 1146 bits, at coding rate $R = 1/4$ and $E_b/N_0 = 0.2$ dB. The interleaver length is 3438.

Note

The hierarchical EXIT charts provide a compromise between asymptotic performance and convergence for a range of degree profiles with two non-zero fractions. It is possible to refine the choice of the degree profile if we want to have an intermediary behaviour. In fact, Fig. 4.5 shows a set of EXIT curves concentrated around OMI = 0.75 for IMI = 0 (*i.e.*, degree profiles (f_2, f_{10}), (f_2, f_{11}), (f_2, f_{12}), (f_2, f_{13}), (f_2, f_{15}) and (f_2, f_{20})). Among them, the irregular TC with a degree profile (f_2, f_{10}) reaches the point of coordinates (1,1) for low values of IMI. Besides, the code with profile (f_2, f_6) has an OMI around OMI = 0.6 for IMI = 0 and is expected to perform the best at high SNRs among all the degree profiles presented in Fig. 4.5, since it reaches high values of OMI for the lowest IMI compared with the other degree profiles. The idea is to mix these two selected configurations (f_2, f_{10}) and (f_2, f_6) in order to benefit from the expected better performance of the latter in the error floor and that of the former in terms of convergence. Then, the new degree profile can be (f_2, f_6, f_{10}) and it is possible to test many other combinations based on the same reasoning.

4.2.2 Performance example of irregular turbo codes

Simulations have been carried out in order to see the effects of the irregularity and the choice of degree profiles on the performance in the waterfall and the error floor regions. In Fig. 4.6, the BER of regular and irregular turbo codes has been simulated for blocks of 1146 bits under random interleaving. As the average degree is 3, the interleaver length is equal to 3438. The use of random interleaving in the simulations indicates the average

performance of the code [13]. Nevertheless, there are then means to find better interleavers than this "average" one. Moreover, two degree profiles are considered in Fig. 4.6: (f_2, f_4) and (f_2, f_8). For profile (f_2, f_4), we have two classes of irregularity: $d = 2$ and $d = 4$. The corresponding fractions are $f_2 = f_4 = 1/2$. The other profile (f_2, f_8) consists of two classes of irregularity: $d = 2$ and $d = 8$. The corresponding fractions are $f_2 = 5/6$ and $f_8 = 1/6$.

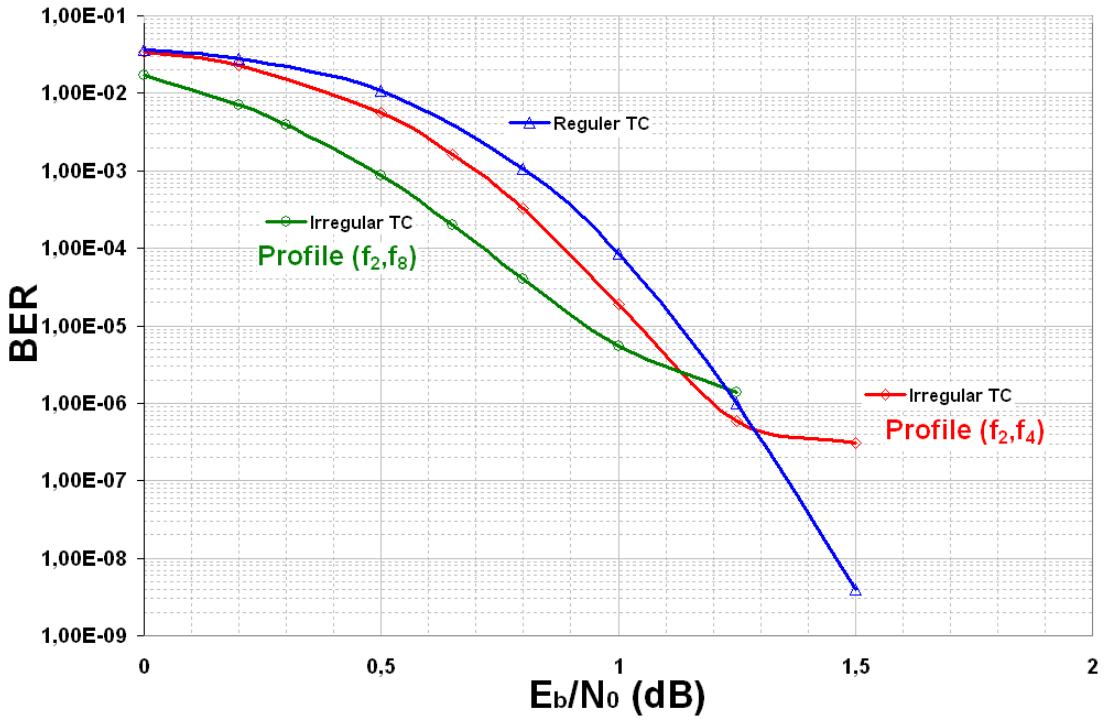


Figure 4.6: BER performance of irregular turbo codes and comparison with regular turbo code for block size $k = 1146$ bits and $R = 1/4$. All simulations use the MAP algorithm with 8 decoding iterations.

In general, we observe a significant gain in the waterfall region for irregular turbo codes. For example, the irregular turbo code using profile (f_2, f_4) gives a gain in the convergence threshold of 0.1 dB compared with regular turbo code. When a higher repetition degree is employed, *i.e.* for profile (f_2, f_8), the gain in convergence is more significant: 0.3 dB at low SNR. However, the error floor for code (f_2, f_8) is higher than the floor for code (f_2, f_4). These simulations confirm what we previously analysed through the EXIT diagrams in subsection 4.2.1.

These simulations confirm also the observation in [51]. Although irregular TCs can achieve performance closer to capacity, their asymptotic performance is very poor. Once the degree profile is defined, it is then necessary to take a great deal of interest in the permutation which plays an important role and influences the performance of the irregular TC.

4.3 Design of suitable permutations for irregular turbo codes

To our knowledge, only one reference [92] deals with the problem of lowering the error floor of irregular TCs: Sawaya *et al.* introduced symbol-based iterative decoding. Their results show that the error floor in the performance of the rate-1/2 irregular TC proposed by Frey *et al.* in [51] is lowered significantly and appears at a bit error probability equal to 6×10^{-6} instead of 4×10^{-4} with code length $n = 131,072$ and 100 decoding iterations. This is mainly due to the use of a rate-2/3 instead of a rate-1/2 RSC constituent code, like a double binary TC. However, no previous work focused on optimizing the interleaver, except in [67] where Kraidy *et al.* use a Progressive Edge Growth (PEG)-based interleaver for irregular TCs to lower the floor in the case of binary erasure channels. Moreover, we are interested not only in large block lengths but also medium and short blocks. Afterward, the proposed solutions in [67, 92] do not seem concrete especially if only few iterations are required during the decoding process. We explain in this section why a random interleaver is not adapted to be used for an irregular TC and we propose a construction of more sophisticated permutations, following the spirit of the PEG algorithm [63].

It is attractive to introduce a restricted number of bits with a very high degree, called pilot bits, whose forward and backward metrics in the decoding process are reliable. Thus, they propagate on both directions and influence the other bits with degree $d = 2$. This phenomenon is behind the gain in the convergence threshold of the irregular TC. However, a major problem is the auto-correlation introduced by these bits. If the pilot bits are uniformly distributed, the auto-correlation effect can be reduced. Nevertheless, the correlation between the different groups of pilot bits is still present unless they represent only very few bits. For this reason, the interleaver should spread these pilot bits along the frame and a random interleaver does not take into account this important condition. Therefore, the design of suitable permutations for irregular turbo codes is a big issue to limit the auto-correlation to the minimum while having high MHDs.

The application of the EXIT technique leads to the choice of two degree values $d = 2$ and $d = 8$ with the following degree profile ($f_2 = 5/6$, $f_8 = 1/6$). In the sequel, we adopt this profile as working assumption. Note that the method explained below is general and can be applied to any other degree profile even using more than two non-zero fractions.

4.3.1 Permutations with uniform distribution of the pilot bits

The first intuitive idea was to design an interleaver where all the groups of eight bits are uniformly distributed. The objective is that they spread their reliable forward and backward metrics along the frame. If we represent the interleaver with a circle, then the bits inside one pilot group are separated with an angle equal to $\frac{2\times\Pi}{d_{max}} = \frac{\Pi}{4}$[1]. Each time we place another group of eight bits, we move it with an angle of $\frac{\Pi}{4\times\eta}$, where η represents the total number of the pilot groups. Let us assume for example that we have only three

[1] As the circumference is $2 \times \Pi$, we obtain: $\frac{2\times\Pi}{d_{max}} = \frac{2\times\Pi}{8\,values} = \frac{\Pi}{4}$

groups of 8 bits ($\eta = 3$). Fig. 4.7 shows that the second group was placed with an angle of $\frac{\Pi}{12}$ after the first group. The same rotation was applied to the last group.

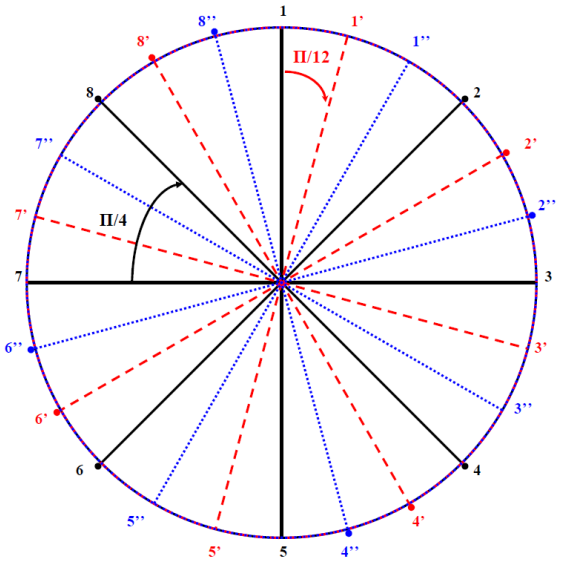


Figure 4.7: Three pilot groups uniformly distributed along the frame.

The next step of the design involves finding an optimal configuration for the groups of two bits that produces high minimum distances. We have performed both a random and S-random permutations for the whole groups of two bits. We have also tested some interleavers that appear to be promising in terms of minimum Hamming distance among the numerous permutation models suggested in the literature, such as ARP interleavers. It was observed that the minimum distance is poor (equal to 6 in the example of Fig. 4.8) whatever the configuration employed for the groups of two bits. As depicted in Fig. 4.8, the spread between the pilot groups is lower than or equal to two (≤ 2). This leads to a high correlation between the groups of eight bits even if we have solved the problem of the correlation between the 8 bits in one pilot group since they are uniformly distributed. Note that this spread also depends on the fraction of the pilot groups which is high ($= \frac{1}{6}$) in our example.

To conclude, the first intuitive idea investigated was to design an interleaver where all the groups of eight bits are uniformly distributed. The objective is that they spread their reliable forward and backward metrics along the frame. On the other hand, the spread between the pilot groups should be large enough to avoid correlation between them. High correlation may dramatically degrade error correction performance and even ruin the possible gain due to large minimum distance [61]. An empirical value for the spread is to be at least equal to $2 \times (\nu + 1)$, where ν is the memory length of the simulated code. The condition on the spread between the pilot groups imposes a constraint on the fraction f_8 (f_{max} in general) of the pilot bits:

$$f_{max} \leq \frac{d_{av}}{2 d_{max} (\nu + 1)} + \frac{1}{k} \tag{4.7}$$

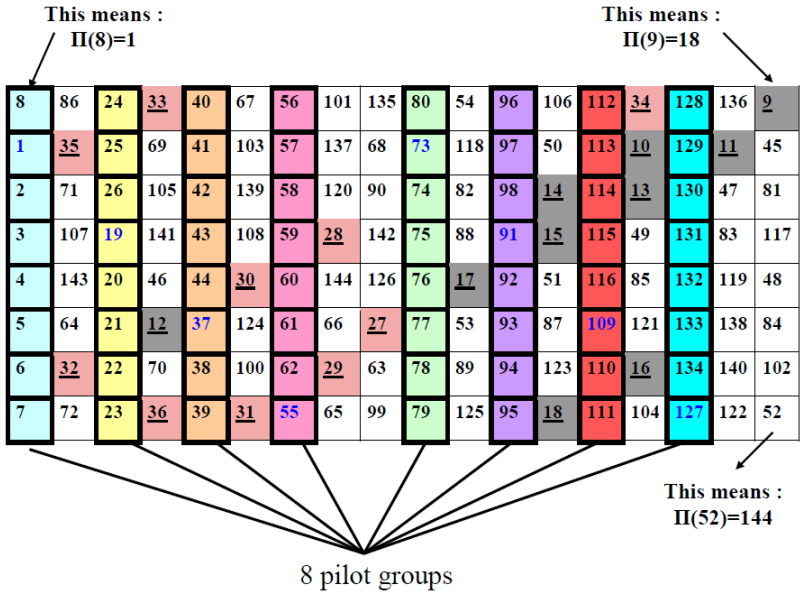


Figure 4.8: Visualization of an interleaver for blocks of 48 bits. We read the output of the interleaver line by line.

where d_{av} is the average information bits degree and k is the total number of information bits. For usual block size values, the term $\frac{1}{k}$ is negligible and the constraint (4.7) above is a relation between f_{max}, d_{max}, and ν that can be expressed as follows:

$$f_{max} \leq \frac{d_{av}}{2d_{max}\,(\nu+1)} \tag{4.8}$$

For $f_8 = \frac{1}{6}$, $\nu = 3$ and $d_{av} = 3$, this condition (4.8) can never be satisfied.

4.3.2 Designing permutations using the Dijkstra's algorithm

In the cases where this condition is not satisfied, we propose an algorithm in order to jointly spread the groups of eight bits along the frame and maximize the MHD. This idea was inspired by a procedure described in [90] where the author focuses on the optimization of TC permutation design with the so-called ARP model.

The Dijkstra's algorithm:

The Dijkstra's algorithm [41], discovered by the pioneering mathematician and programmer E.W.Dijkstra, finds the shortest path from a source vertex to all other vertices in a weighted directed graph without negative edge weights. The algorithm can be described in the following way: we pour some water in the starting (or initial) node and we visit the nodes of the graph in the order where they receive the water. The algorithm keeps the shortest distance of any vertex from the source in an array, *sDist*. The shortest distance of the source to itself is zero. *sDist* for all other vertices is initially set to infinity to indicate that those vertices are not yet processed. The algorithm operates step by step,

where at each step it updates locally the distances of certain nodes in the graph. Here is the algorithm for a graph G with vertices $V = \{v_1, ...v_n\}$ and edge weights w_{ij} for an edge connecting vertex v_i with vertex v_j. Let the source be v_1.

- ◇ Let S be a set used to keep track of all vertices that we have already computed the shortest distance to from the source. Initialize the set $S = \emptyset$.
- ◇ Initialize the array $sDist$ of estimates of shortest distances. $sDist\,[1] = 0$, while $sDist\,[i] = \infty$, for all other i. This means that our estimate from v_1 to v_1 is 0, and all of our other estimates from v_1 are infinity.
- ◇ While $S \neq V$ do the following:

  1. Find the vertex v_i, not belonging to S, that corresponds to the minimal estimate of shortest distances in array $sDist$.
  2. Add this vertex, v_i into S.
  3. For each vertex v_j connected with v_i, compute $sDist\,[i] + w_{ij}$. If this quantity is less than $sDist\,[j]$, then set $sDist\,[j] = sDist\,[i] + w_{ij}$.

The proposed algorithm:

To apply the Dijkstra's algorithm in our specific context, we reduce the space of search by building a graph. Similarly to the procedure described in [90], the purpose is maximize the correlation girth which is the length of a shortest cycle contained in the graph. This criterion of selection was a priority for the author of [90]. His first purpose was to maximize the correlation girth while keeping an acceptable minimum Hamming distance. However, the minimum Hamming distance is our most important criterion of selection in order to improve the distance properties of irregular TCs.

At the beginning, the graph exists and is empty. As we consider a tail-biting code, the graph has the form of a ring. The nodes, empty at first, are connected two by two in the ring. Thus, each node v_i is connected to only one predecessor and one successor v_j in the graph. The weight of each connection w_{ij} is equal to one. The purpose of the algorithm is to put addresses in the nodes. Before interleaving, an appropriate repetition is applied to each bit among the k bits stemming from the source. For the k bits stemming from the source, we assume that j is the address of the first copy before interleaving. As the bit is repeated d times, $j + i$ corresponds to the address of the $(i + 1)^{th}$ copy for $1 \leq i \leq d - 1$. We denote by max the maximum estimate of distance in the array $sDist$. The addresses that appear progressively in the graph are the interleaved addresses corresponding to the output of the interleaver. The algorithm used to build the interleaver is the following:

- ◇ /* For each bit stemming from the source, we want to find the interleaved addresses for the d copies:*/

1. For i from 0 to $d - 1$

   a) If $i = 0$

- ◇ /* The algorithm searches an interleaved address for the first copy of the bit:*/
- ◇ We choose $\Pi(j)$ at random among the available interleaved addresses.
- ◇ /* The node $v_{\Pi(j)}$ is no more empty and corresponds to the interleaved address of j.*/

b) Else

- ◇ /* The algorithm searches an interleaved address for the $(i+1)^{th}$ copy of the bit:*/
- ◇ Initialize max to 0.
- ◇ /* We must take into account the interleaved addresses of all the copies of the same bit already handled:*/
- ◇ For t from 0 to $i-1$
  - Compute the distances from $v_{\Pi(t)}$ to all the neighbouring vertexes in the actual graph by Dijkstra's algorithm.
  - Find the highest distance d_{high} in the array $sDist$.
  - If the best possible value d_{high} is greater than max, replace max by d_{high}.
  - Increment t.
- ◇ Choose an address $\Pi(j+i)$ at random such as the scores for all the $\Pi(t)$ are greater than α times the best possible value max, where α is a real number satisfying $0 < \alpha \leq 1$.
- ◇ /* The node $v_{\Pi(j+i)}$ is no more empty and corresponds to the interleaved address of $j+i$.*/
- ◇ Add i crossbars, of weight equal to zero, to the graph. In fact, these crossbars connect $v_{\Pi(j+i)}$ with $v_{\Pi(j)}$, $v_{\Pi(j+1)}$, ... $v_{\Pi(j+i-1)}$ and we have $w_{\Pi(j+i),\Pi(j)} = w_{\Pi(j+i),\Pi(j+1)} = ... = w_{\Pi(j+i),\Pi(j+i-1)} = 0$ since we connect different interleaved data of the same information bit. These crossbars update the graph and are necessary to apply correctly the Dijkstra's algorithm in the next step.

c) Increment i.

2. Every time an interleaver is found, we estimate the minimum Hamming distance of the irregular TC using the all-zero iterative decoding algorithm (see 1.4.3). Only interleavers that improve the asymptotic performance of irregular TCs are memorized.

The parameter $0 < \alpha \leq 1$ is used to implement a random variation in the selection. The value $\alpha = 1$ corresponds to the original Dijkstra's algorithm. We noticed that if we fix $\alpha = 1$ in our algorithm, the obtained interleavers produce high values of girths but sometimes unacceptable values of minimum distances. Therefore, different values were

tested for the parameter α and we finally set α to 0.85, giving a reasonable space of search.

Compared with the approach in [90] where the graph is almost cubic, the graph used in our algorithm is not the same since we consider a self-concatenated code. In [90], all the vertices $\{x_1, x_2, ..., x_k, y_1, y_2, ..., y_k\}$ have exactly three neighbours apart from x_1, x_k, y_1 and y_k. In our graph, the vertices have $d+1$ neighbours: the predecessor and the successor in the ring as well as $d-1$ neighbours because each information bit has a repetition degree d.

Example:

To apply the algorithm for the first information bit stemming from the source ($j = 1$) with repetition degree $d = 8$, we choose a random interleaved address $\Pi(1)$ for the first copy: $\Pi(1) = 565$. Then, we compute the distances from $\Pi(1)$ for any vertex in the actual graph by Dijkstra's algorithm. $\Pi(2)$, the interleaved address of the second copy, is chosen at random such as the score for $\Pi(1)$ is greater than α times the best possible value. We obtain $\Pi(2) = 273$ and we add a connection in the graph between $v_{\Pi(1)}$ and $v_{\Pi(2)}$, represented by a crossbar of weight equal to zero: $w_{\Pi(1),\Pi(2)} = 0$.

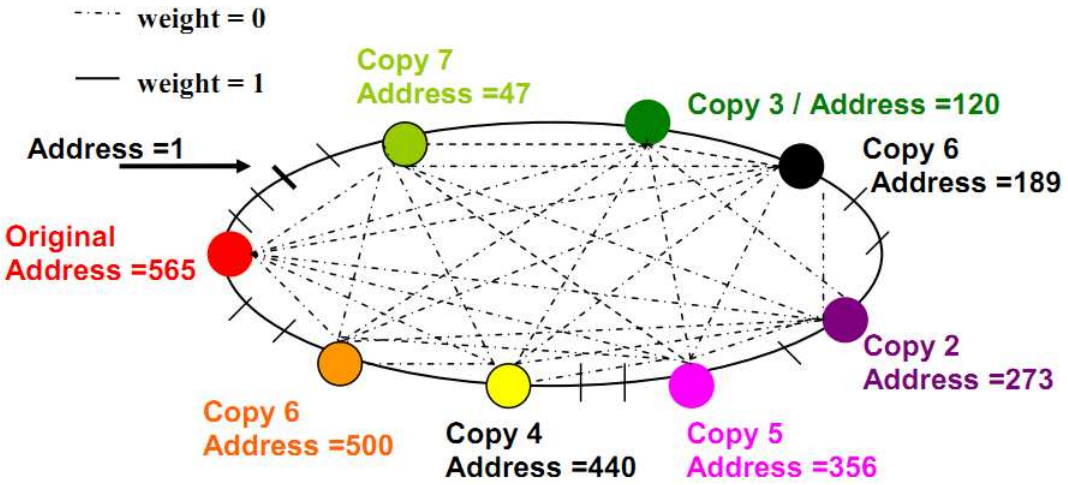


Figure 4.9: Example of the placement of a pilot group with degree $d = 8$ in the graph using the proposed algorithm.

Afterwards, we compute the distances from $\Pi(1)$ and also from $\Pi(2)$ for any vertex in the actual graph by Dijkstra's algorithm. $\Pi(3)$, the interleaved address of the third copy, is chosen at random such as the scores for both $\Pi(1)$ and $\Pi(2)$ are greater than α times the best possible value. We obtain $\Pi(3) = 120$ and we add two connections in the graph between represented by crossbars of weight equal to zero: $w_{\Pi(1),\Pi(3)} = 0$ and $w_{\Pi(2),\Pi(3)} = 0$. We continue to run the algorithm for all the other copies...

The algorithm constructs the graph progressively. At the end, crossbars are added to the graph as shown in Fig. 4.9. In general, for each information bit with repetition degree d, $\frac{d\times(d-1)}{2}$ connections of weight equal to zero appear in the graph (28 connections in the example).

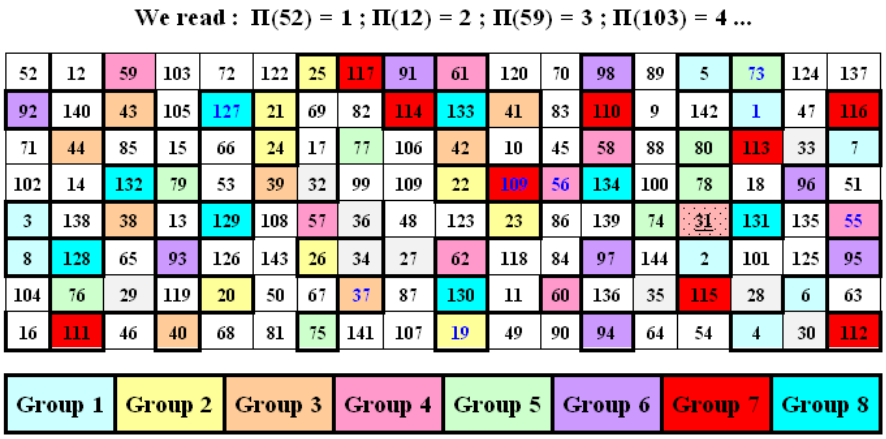


Figure 4.10: Visualization of an interleaver for blocks of 48 bits.We read the output of the interleaver line by line.

The proposed algorithm allows suitable permutations to be designed that lower the error floor of irregular TCs. As an example, this algorithm was run for blocks of 48 bits with degree profile (f_2, f_8). As the average degree is 3, the interleaver length is equal to 144. Fig. 4.10 shows that the correlation effect between the pilot groups is reduced since the pilot bits are distributed along the frame in a way that they guarantee a good spread between the different groups. To compare with the previous example of Fig. 4.8, the minimum distance of the code is now 19 instead of 6. Another interleaver allowing the irregular code to have a higher distance, *i.e.* 21, was found and the simulation results are available in Fig. 4.11.

4.3.3 Error rate performance of irregular turbo codes with an optimized interleaver

In Fig. 4.11, the FER performance of irregular turbo codes has been simulated for blocks of 48 bits and 192 bits, under both random and optimized interleaving. The degree profile consists of two degrees: $d = 2$ and $d = 8$. The corresponding fractions are $f_2 = 5/6$ and $f_8 = 1/6$. As the average degree is 3, the interleaver length is equal to 144 and 576 respectively. The parameter α is set to 0.85. When the optimized interleaver given by the Dijkstra's algorithm is employed instead of a random permutation, we observe a significant gain in the error floor: about 2.5 decades for 48 bits and 3.5 decades for 192 bits.

For medium sizes and large blocks, the algorithm may take an unacceptable computational time in order to detect interleavers that improve significantly the distance properties of irregular TCs and lower the observed floor. This computational cost grows exponentially with the block size. It does not mean that we are not able to find a better interleaver. It just means that we can improve the distance properties of irregular TCs, without any certainty that the best possible case is reached and the optimal interleaver is detected.

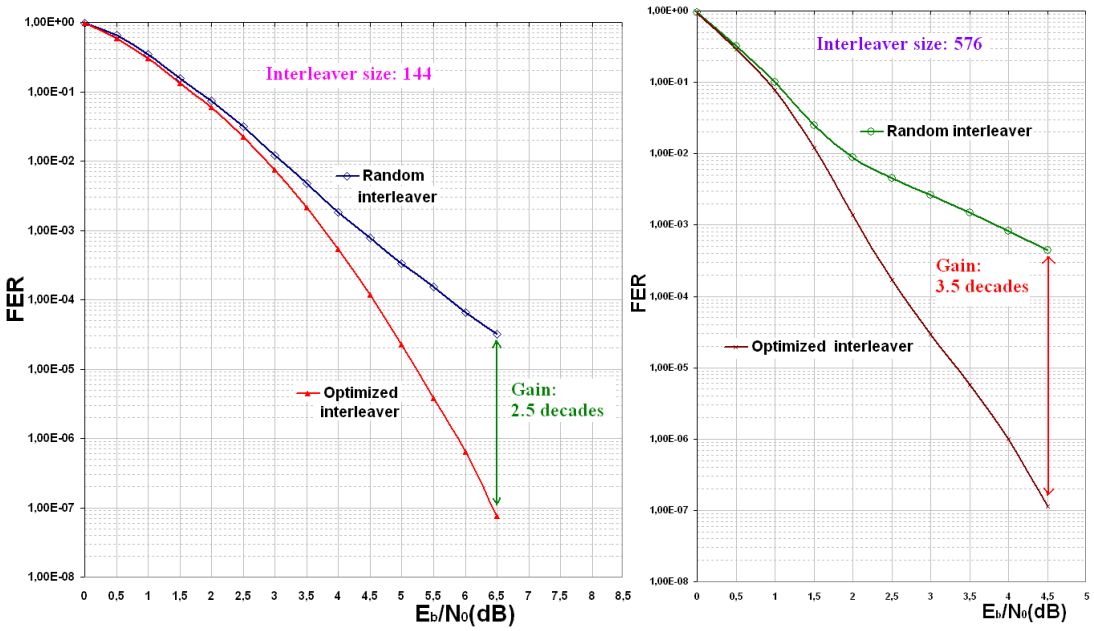


Figure 4.11: FER performance of irregular turbo codes under both random and optimized interleaving for short block sizes at coding rate $R = 1/4$. All simulations use the MAP algorithm with 10 decoding iterations.

For instance, the algorithm was run for blocks of 1146 bits with degree profile (f_2, f_8) and FER performance of irregular TCs has been simulated. As the average degree is 3, the interleaver length is equal to 3438. Fig. 4.12 compares the FER performance of the code under both random and optimized interleaving. A gain of two decades in the error floor is observed, in favour of the optimal interleaver. This may not be the best interleaver that has to be found. However, performance of irregular TCs at high SNRs is also improved in this case.

Although the proposed algorithm is very fast for short block sizes, it may take unacceptable computational time for medium sizes and large blocks to find good interleavers and we cannot be sure of detecting all the possible cases. Like for random interleavers, one additional drawback of this family of interleavers is the necessity to store the interleaved addresses as no equations are available for the permutation.

4.4 Adding a post-encoder to irregular turbo codes

As previously explained, devising good interleavers for irregular TCs proves to be a difficult task. In order to ensure large asymptotic gain at very low error rates, even with non-optimized internal permutation, we propose an irregular TC inspired by our work about 3D TCs in order to improve the distance properties of these codes.

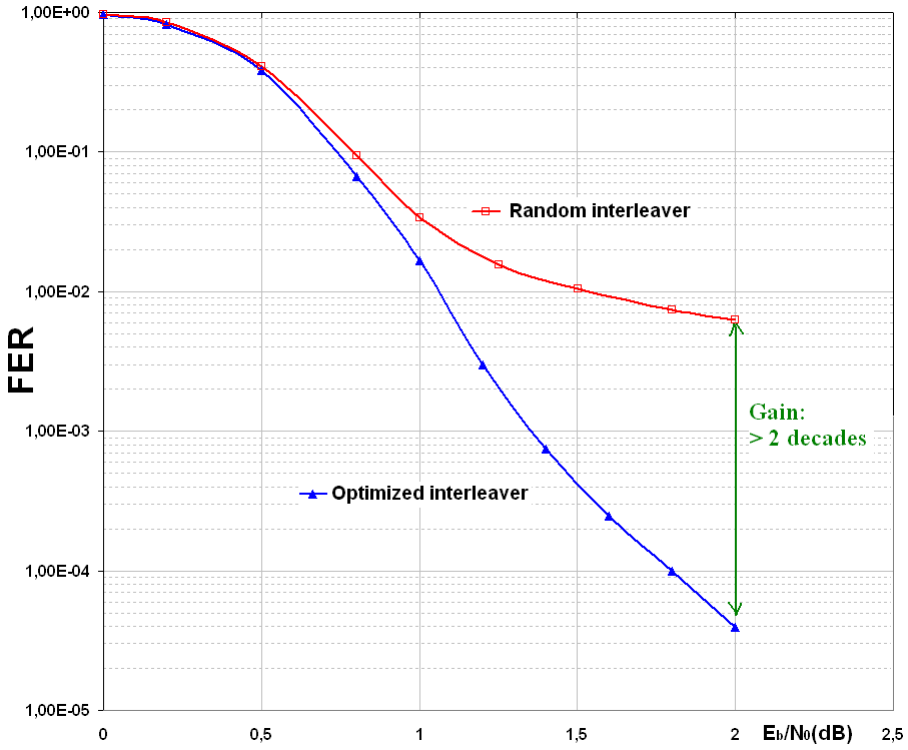


Figure 4.12: FER performance of irregular turbo codes under both random and optimized interleaving for blocks of 1146 bits at coding rate $R = 1/4$. All simulations use the MAP algorithm with 8 decoding iterations.

4.4.1 Proposed modified encoding procedure

A block diagram of the proposed irregular turbo code is depicted in Fig. 4.13. A fraction λ ($0 \leq \lambda \leq 1$) of the parity bits are post-encoded by a rate-1 post-encoder. For the 3D TC, the increase in minimum distance is significant at the expense of a loss in convergence threshold and an increase in complexity. Based on our analysis in the previous chapters, the same behaviour is expected for irregular TCs. Here also, the permeability pattern is considered to be regular. For instance, if $\lambda = \frac{1}{5}$ the bits to be post-encoded are chosen in a regular basis $\{10000\}$.

4.4.2 Performance of irregular TCs with post-encoding

We have investigated the distance gain for different block sizes and permeability rates. The values of the the minimum Hamming distance are obtained using the all-zero iterative decoding algorithm. Similarly to the case of 3D TCs, we have observed that the addition of the post-encoder improves the asymptotical behaviour of irregular TCs.

For example, the addition of the post-encoder results in an increase in the minimum Hamming distance of the irregular TC by more than 50% from $d_{min} = 33$ to $d_{min} = 50$ for code rate $R = 1/4$, $\lambda = 1/8$ and $k = 4094$ bits. This value has to be compared to $d_{min} = 44$ which is the distance of the regular TC in this case. Then, the irregular TC concatenated

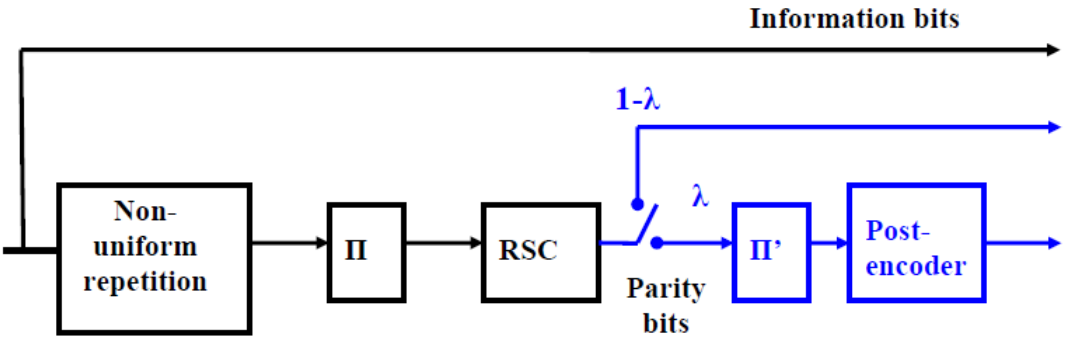


Figure 4.13: An irregular turbo code concatenated with a rate-1 post-encoder at its output.

with a post-encoder outperforms the regular TC. These results have been validated by simulations.

Fig. 4.14 shows the BER and FER performance of regular and irregular TCs for blocks of 4094 bits under 3GPP2 interleaving. Thus, the interleaver length is equal to 12282. In our simulation, a regular TC corresponds to a TC where all the bits are repeated three times each. For the irregular TC, the degree profile consists of two non-zero fractions: one for the degree $d = 2$ ($f_2 = \frac{5}{6}$) and another one $d = 8$ ($f_8 = \frac{1}{6}$). Like with 3D TCs, the post-encoder improves performance at low error rates. Compared with irregular TCs, Fig. 4.14 shows that the gain at high SNRs is nearly 2.5 decades when the post-encoding is performed. Note that compared to the search for permutations in section 4.3 which has limitations for medium and large block sizes, it is possible and easier here to simulate irregular TCs for large blocks. In other words, there is no limitation on the block size. The great advantage is that irregular TCs with post-encoding perform better than the regular TCs at low but also at high error rates. Other simulations for different bloc sizes were carried out to confirm these results.

Transmission over fading channels

Similar simulations have been carried out over both Gaussian and Rayleigh fading channels. In Fig. 4.15, the BER and FER performance of regular and irregular TCs have been simulated for blocks of 2046 bits. Thus, the interleaver length is equal to 6138. Again, it is observed that irregular TCs with post-encoding perform better than the regular TCs in both the waterfall and error floor regions. For a transmission on the Rayleigh channel, the gain at high SNRs exceeds one decade, whereas for a transmission over the Gaussian channel, the gain at high SNRs is nearly 2 decades. It is possible to increase even more the minimum distance by the search of adapted pattern of post-encoding. Here, the post-encoding is regular. However, it is possible to postcode only the pilot bits, or only the bits with the lowest degree, or to find a balance between both of them. This perspective is expected to give better results and can be investigated in a future work.

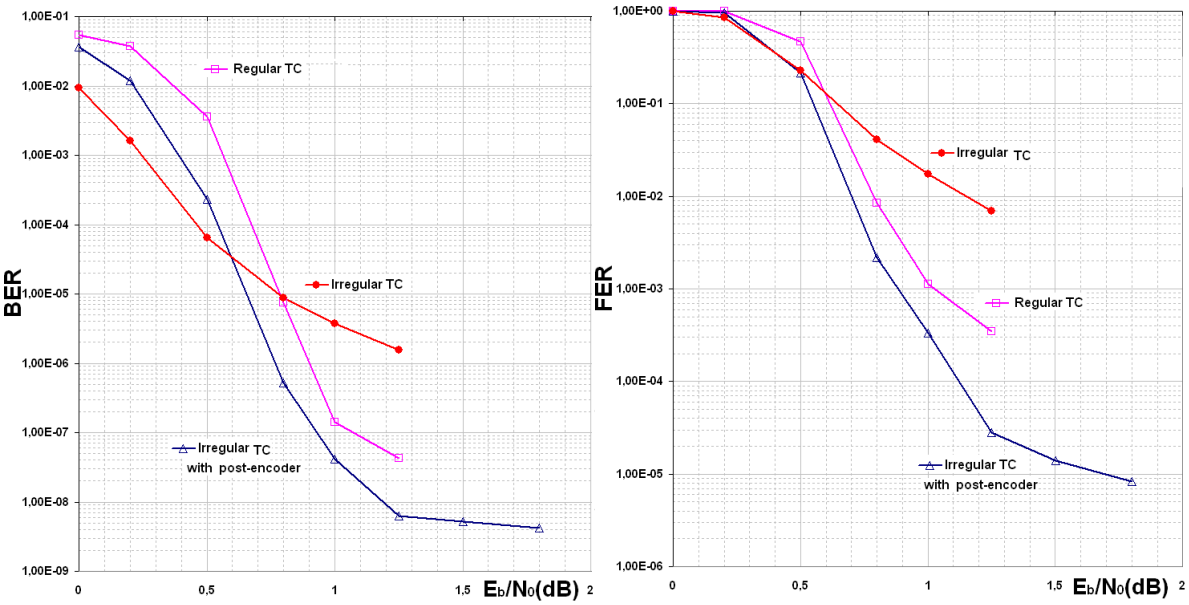
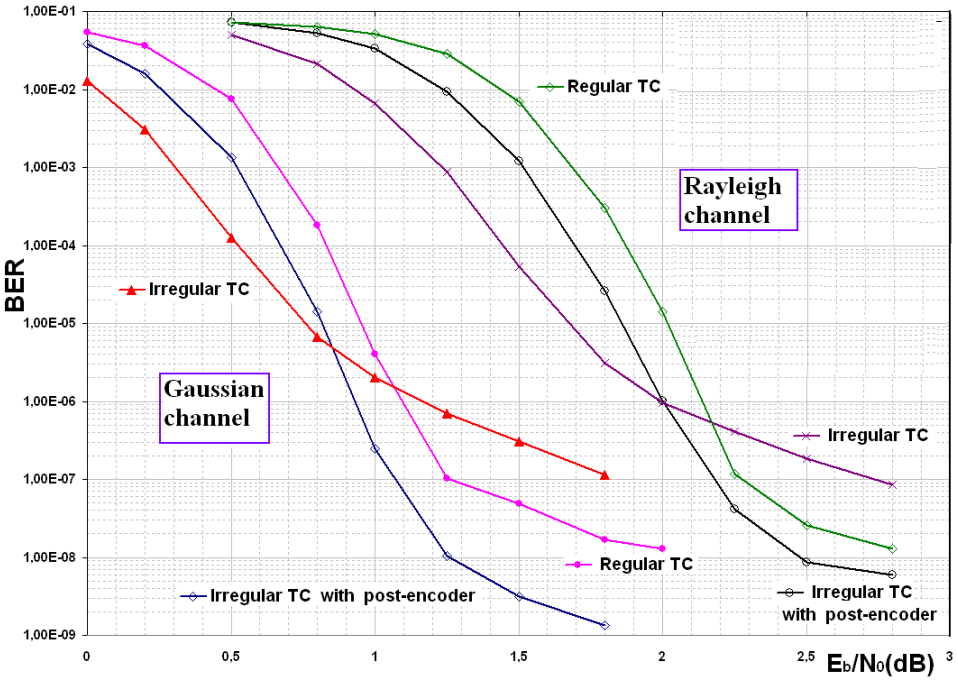


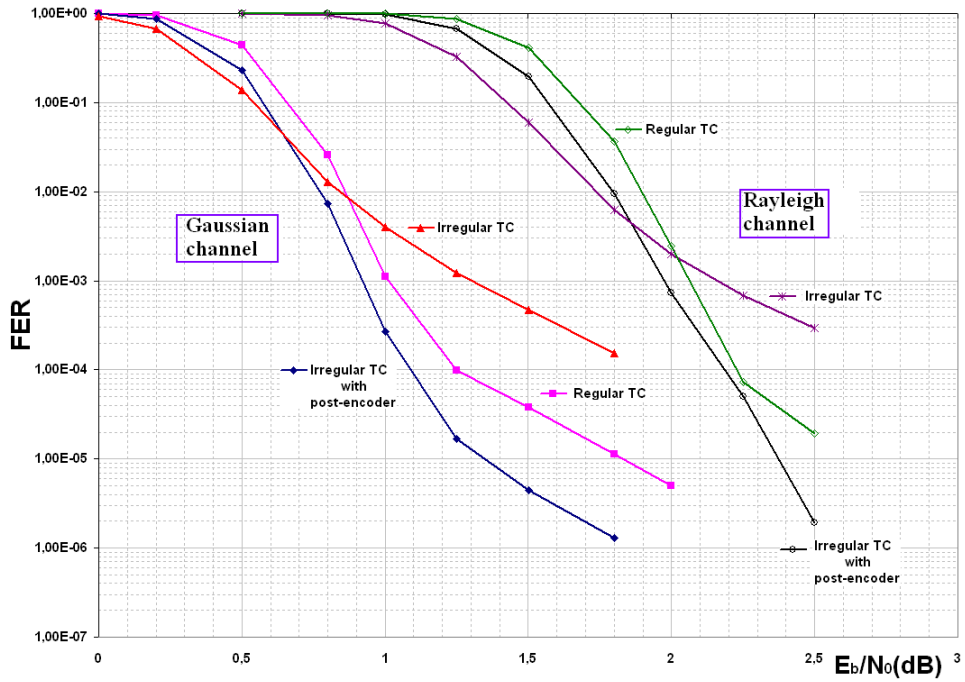
Figure 4.14: BER and FER performance of 3GPP2 irregular TCs and comparison with the corresponding regular TCs for blocks of 4094 bits. All simulations use the MAP algorithm with 10 decoding iterations. Transmission is over the Gaussian channel.

4.5 Conclusion

In this chapter, we investigated irregular turbo codes. In order to select a good profile of degrees, we used hierarchical EXIT charts instead of long Monte Carlo simulations. This is one of the contributions of the thesis. Another contribution was to focus later on the design of suitable permutations in order to lower the floor. Graph-based permutations built from a combination of the Dijkstra's algorithm with an estimation of the minimum distance improve significantly the distance properties of irregular TCs. However, it is only practicable for short to medium blocks. It is possible to investigate the interleavers provided by the proposed algorithm and explore them in details in order to find structured interleavers, having similar properties, which can be described in an analytical way. This perspective is interesting since the addresses do not need to be stored if it is the case. However, it is commonly known that devising permutations for turbo codes in not an easy task. Finally, an interesting idea was to reuse the previous analysis about 3D TCs and to apply it in the case of irregular turbo codes. This important contribution shows that irregular turbo codes with post-encoding perform better than regular turbo codes in both the waterfall and error floor regions.



(a) BER



(b) FER

Figure 4.15: BER and FER performance of 3GPP2 irregular TCs and comparison with the corresponding regular TCs for blocks of 2046 bits. All simulations use the MAP algorithm with 10 decoding iterations. Transmissions over Gaussian and Rayleigh fading channels are considered.

Conclusions and perspectives

In the early nineties, the invention of TCs was a revival for the channel coding research community. Their near-capacity performance and their suitability for practical implementation explain the adoption of TCs in various communication standards as early as the late nineties. However, TCs suffer from a flattening effect when the error rate reaches a limit and stops improving even if the number of iterations is increased. In future system generations, low error rates will be required to open the way to real time and more demanding applications, such as television broadcasting or videoconferencing. The minimum Hamming distance may not be sufficient to ensure large asymptotic gains at very low error rates. Therefore, more powerful coding schemes are required. At the same time, a reasonable complexity should be preserved.

The first aim of the thesis was to explore a new hybrid concatenation structure combining both parallel and serial concatenation based on a 3-dimensional code, simply derived from the classical TC by concatenating a rate-1 post-encoder at its output. Performance of the 3D TC depends on some key parameters. The interleaving law Π' (which permutes the parity bits before feeding them to the post-encoder) and the permeability rate λ have been properly optimized. Besides, I discussed the different requirements that the post-encoder has to meet and how it is possible to choose a post-encoder by means of an EXIT analysis. The most interesting property of the 3D TC is that it significantly improves performance in the error floor region with respect to the classical turbo code. Several union bounds on the minimum distance of 3D binary TCs with 8-state upper and lower constituent encoders and 3GPP2 interleavers were presented. Besides, the different stages were illustrated with simulation results and asymptotical bounds.

In the case of the 3GPP2 interleaving, I observed packets of errors at the output of the decoder. This is one drawback of the 3D TC, as the gain in terms of BER is not as much as the improvement in FER.

A thorough complexity analysis of the 3D decoder has been carried out in order to estimate the additional complexity. When high throughputs are required for a given application, several SISO processors can be placed in parallel while only one 3D predecoder is required in most cases; which decreases the relative additional complexity of the 3D coding scheme.

Later, I was interested in improving 3D TCs. An optimization method was introduced in order to increase even more the minimum Hamming distance of the 3D TC and an algorithm was proposed. Applying an irregular pattern of post-encoding allows the asymptotic performance of the codes having low multiplicities at the beginning of the spectrum to be improved. Then, convergence issues were discussed. In fact, one additional drawback of

the 3D TC structure is that the loss of convergence is more significant for fading channels compared with Gaussian channels. Therefore, it seems more necessary to find a solution to this problem. The use of a time varying post-encoder reduced the observable loss of convergence threshold for 3D TCs by 10% to 50% of the value expressed in dB. However, this technique requires finding an optimal period L, in order to replace properly some redundancies by the others. And this is not an easy task, since it is a matter of convergence/distance trade-off. Many values have to be tested, depending on the block size and the coding rate. Another alternative was considered when the code is associated with high order modulations. In this case, there is no need to use a time varying trellis and a specific mapping allows obtaining even a gain in the waterfall region. Therefore, the 3D TC is adapted to be used in high spectral efficiency transmission schemes. Thus, it is possible to build 3D TCs which have good performance in both regions.

I noticed that the notion of irregularity plays a major role to build 3D TCs having good performance at low and high SNRs. In fact, the optimization method is based on the use on a non regular pattern of post-encoding and allows the minimum distance to be increased. Besides, the time varying post-encoder (5, 4:7) with a little irregularity and also the association of the 3D TC with high order modulations, where both the systematic bits and the post-encoded parity bits are more protected than the other parity bits create a sort of irregularity in the Gray mapping. Both represent a success in reducing or even eliminating the problem of convergence loss. The next step of the study concerned the investigation of irregular TCs. The aim is to obtain an irregular TC which performs well in both the waterfall and the error floor regions.

Work on irregular LDPC codes was encouraging to study different aspects of the problem. To start, I used EXIT charts instead of long Monte Carlo simulations in order to select a good profile of degrees. Afterwards, I was interested in the design of powerful permutations suited for such code structures. Graph-based permutations built from a combination of the Dijkstra's algorithm with an estimation of the minimum distance improve significantly the distance properties of irregular TCs. However, it is only practicable for short to medium blocks. It is possible to investigate the interleavers provided by the proposed algorithm and explore them in details in order to find structured interleavers, having similar properties, which can be described in an analytical way. However, it is commonly known that devising permutations for turbo codes is not an easy task. An interesting idea was to reuse the previous analysis about 3D TCs and to apply it in the case of irregular TCs. The association of irregular TCs with the same post-encoder used for 3D TCs results in irregular turbo coding schemes which perform better than regular TCs in both the waterfall and error floor regions.

Of course, the techniques studied can be improved. The suggested directions for future research work are summarized below.

- ◇ For example, it seems interesting to think about the most effective way to post-encode the parity bits. The authors in [13] proposed different puncturing patterns that maximize the minimum distance of TCs under random interleaving. It is really encouraging to apply their results for the post-encoding pattern and analyze the impact on the performance of 3D TCs.

- ◇ While improving the convergence threshold of 3D TCs, I imagined new structures that can be applied in the future. What about using an 8-state time-varying post-encoder? This structure is obviously more complicated, but it is certainly more powerful. The loss in convergence can be significant. However, this does not represent a major problem for high coding rates and I expect that increased minimum distances of 3D TCs can be attained in this case.

- ◇ Also, what about the association of 3D TCs with rotated constellations, like those used for the second generation of terrestrial digital TV? Since the rotated constellations technique is particularly effective in bad transmission conditions, the use of the specific Gray mapping explained in the manuscript may compensate the loss in the convergence threshold and could increase even more the gain in convergence threshold observed over fading channels. I think that the search in this direction would give optimistic observations and interesting results.

- ◇ Moreover, all the analysis of 3D TCs was dedicated to the case of binary TCs. What about double binary (and m-binary in general) 3D TCs? It is known that double binary TCs perform better than classical TCs at both high and low error rates. It is attractive to apply the optimization method as well as the time-varying technique in the case of double binary 3D TCs.

Last but not least, the design of suitable permutations for irregular TCs is an important future research work. In fact, the proposed algorithm is very promising. However, it is necessary to find techniques that eliminate the interleavers producing low minimum distances early in the search process. In this way, the space of search would be reduced and the algorithm would become promising even for large blocks.

Contributions to the literature

Conference papers

1. KBAIER BEN ISMAIL Dhouha, DOUILLARD Catherine and KEROUÉDAN Sylvie, "Improving 3-dimensional turbo codes using 3GPP2 interleavers", *ComNet'09: 1st International Conference on Communications and Networking*, Hammamet, Tunisia, 03-06 november 2009.

2. KBAIER BEN ISMAIL Dhouha, DOUILLARD Catherine and KEROUÉDAN Sylvie, "Reducing the convergence loss of 3-dimensional turbo codes", *6th International Symposium on Turbo Codes & Iterative Information Processing*, Brest, France, 06-10 september 2010, pp. 146-150.

Journal papers

1. KBAIER BEN ISMAIL Dhouha, DOUILLARD Catherine and KEROUÉDAN Sylvie, "Analysis of three-dimensional turbo codes", to appear in *Annals of Telecommunications*, available online at `http://www.springerlink.com/content/1r8785617q48n106/`.

2. KBAIER BEN ISMAIL Dhouha, DOUILLARD Catherine and KEROUÉDAN Sylvie, "Design of suitable permutations for irregular turbo codes", *Electronics Letters*, vol. 47, no. 11, June 2011, pp. 748-749.

3. KBAIER BEN ISMAIL Dhouha, DOUILLARD Catherine and KEROUÉDAN Sylvie, "Improving irregular turbo codes", *Electronics Letters*, to appear.

Submitted journal paper

1. KBAIER BEN ISMAIL Dhouha, DOUILLARD Catherine and KEROUÉDAN Sylvie, "Design of a new generation turbo codes", submitted to *IEEE Transactions on Wireless Communications*.

Appendix A

Cdma2000 interleaver

THE cdma2000 interleaver is based on the principle of generating the interleaving positions through a counter that generates addresses which are modified through a preset table and a function that reverses the order of the bits. The resulting address vectors determine the permutation of the input data. In cdma2000, the input to the interleaver and the output data from the interleaver are defined as arrays (vectors) of length N_{turbo}. The values that the N_{turbo} variable can take are defined by the standard. Fig. A.1 shows the flow diagram of the interleaver used by the cdma2000 turbo encoder, which has as input the *packet size* variable which is used to determine from Table A.1 both the n and N_{turbo} parameters. The value of n is an interleaving parameter defined as an integer in the range $3 \leq n \leq 7$. N_{turbo} is the actual number of information bits in the interleaving block and must satisfy the relationship $N_{turbo} \leq 2^{n+5}$. The *packet size* is six bit longer than N_{turbo} because the six tail bits are used to force the turbo encoder to the initial state.

The parameters in Table A.1 are defined for reverse link channels *i.e.*, channels going from the mobile station to the base station. For forward link channels (base station to mobile), n is in the range $5 \leq n \leq 7$ and the values of *packet size* and N_{turbo} are different. The interleaving algorithm is the same for the reverse link as well as the direct link.

$Packet\_size$	n	N_{turbo}
256	3	250
512	4	506
1024	5	1018
2048	6	2042
4096	7	4090

Table A.1: Turbo interleaver parameter.

The sequence of the interleaver output addresses is generated by the procedure illustrated in Fig. A.1 and described below:

1. First, the *MSB* address is calculated by taking the n least significant bits of the value of the address counter n most significant bits plus one.

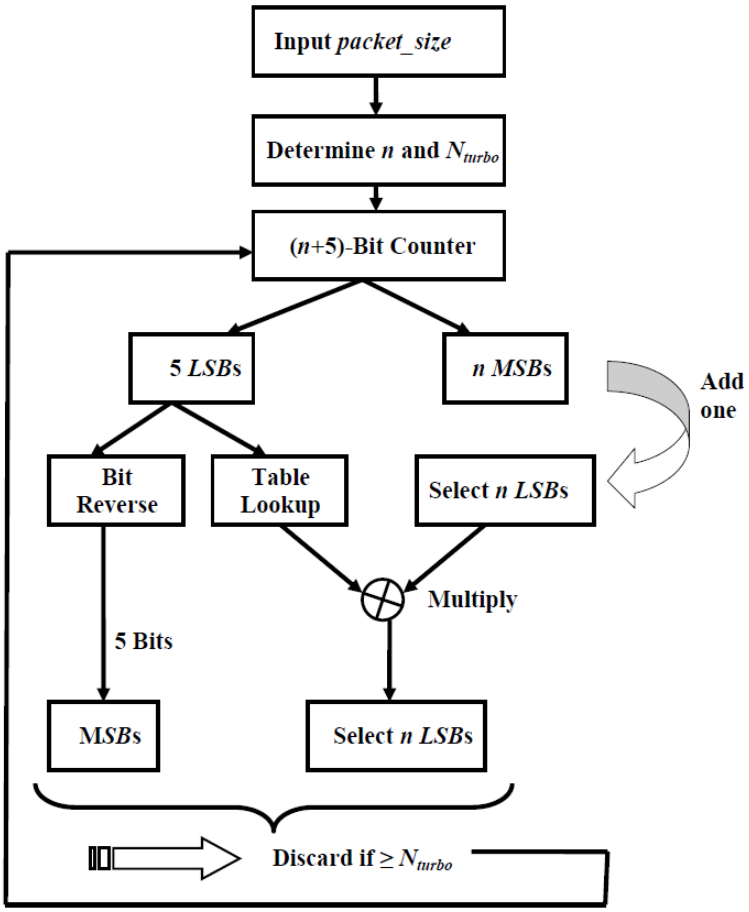


Figure A.1: Flow diagram for the cdma2000 standard's turbo interleaver algorithm.

2. Then, Table A.2 is indexed using the counter's five least significant bits. This lookup table indexing provides an LSB address of n bits.

3. The next step is to multiply the values obtained in steps 1 and 2, and to discard all except the n least significant bits. This will constitute the lower part (or LSB) of the final address.

4. The higher part (or MSB) of the final address is obtained by bit-reversing the five least significant bits of the counter.

5. The tentative of the output address is accepted only if it is less than N_{turbo}, otherwise the address is discarded.

The counter is increased by one and steps 1 through 5 are repeated until all N_{turbo} interleaving addresses are obtained.

Index	$n = 3$	$n = 4$	$n = 5$	$n = 6$	$n = 7$
0	1	5	27	3	15
1	1	15	3	27	127
2	3	5	1	15	89
3	5	15	15	13	1
4	1	1	13	29	31
5	5	9	17	5	15
6	1	9	23	1	61
7	5	15	13	31	47
8	3	13	9	3	127
9	5	15	3	9	17
10	3	7	15	15	119
11	5	11	3	31	15
12	3	15	13	17	57
13	5	3	1	5	123
14	5	15	13	39	95
15	1	5	29	1	5
16	3	13	21	19	85
17	5	15	19	27	17
18	3	9	1	15	55
19	5	3	3	13	57
20	3	1	29	45	15
21	5	3	17	5	41
22	5	15	25	33	93
23	5	1	29	15	87
24	1	13	9	13	63
25	5	1	13	9	15
26	1	9	23	15	13
27	5	15	13	31	15
28	3	11	13	17	81
29	5	3	1	5	57
30	5	15	13	15	31
31	3	5	13	33	69

Table A.2: Cdma2000 turbo interleaver lookup table.

Example of operation

In this example a packet size of 512 bits is used. Following the algorithm described in Fig. A.1 we have:

1. From Table A.1 $n = 4$ and $N_{turbo} = 506$.

2. Start the counter of $n + 5 = 9$ bits at zero, *i.e.* counter = 000000000.

3. Save the four most significant bits of counter in the *MSB* variable.
4. Save the five least significant bits of counter in the *LSB* variable.
5. Add one to *MSB*, *i.e.* $MSB = 0001$ for the first iteration.
6. Use Table A.2 with the five *LSB*s of counter and column n ($index = 00000$ for the first iteration) and store the n-bit result in *LSB*, *i.e.*, $LSB = 0001$ for the first iteration.
7. Take the n least significant bits of the $MSB \times LSB$ product and store it.
8. Bit reverse the five least significant bits of counter to obtain the higher part, *i.e.*, the *MSB*.
9. Form a tentative output address that has its *MSB*s equal to the value obtained in step 8 and its *LSB*s equal to the value obtained in step 7.
10. Accept the output address if it is less than N_{turbo}.
11. Add one to counter.
12. Go to Step 3.

For this example, the input vector has 506 data, whose values for simplicity are numbered from one to five hundred and six. As the input vector is relatively large in size, only some positions with their values are shown in Table A.3. Fig. A.2 shows the interleaved output data vector.

1	2	3	...	505	506

Table A.3: Cdma2000 example input vector.

{6 272 134 400 66 330 202 464 46 304 168 428 112 356 240 486 30 288 154 404 82 340 224 466 62 306 186 448 124 372 256 502 11 271 139 399 67 323 195 463 43 303 175 423 111 359 239 491 27 287 147 407 83 343 223 467 59 307 179 447 119 375 255 16 270 144 398 **68** 332 204 462 40 302 166 418 110 362 238 496 24 286 156 410 **84** 346 222 468 56 308 188 446 114 378 254 5 269 133 397 69 325 197 461 37 301 173 429 109 365 237 485 21 285 149 413 85 349 221 469 53 309 181 445 125 381 253 501 10 268 138 396 70 3 34 206 460 34 300 164 424 108 368 236 490 18 284 158 416 86 352 220 470 50 310 190 444 120 384 252 506 15 267 143 395 71 327 199 459 47 299 171 419 107 355 235 495 31 283 **151** 403 87 339 219 471 63 311 183 443 115 371 251 4 266 132 394 72 336 208 458 44 298 162 430 106 358 234 484 28 282 160 406 88 342 218 472 60 312 192 442 126 374 250 500 9 265 137 393 73 329 201 457 41 297 169 425 105 361 233 489 25 281 153 409 89 345 217 473 57 313 185 441 121377 249 505 14 264 142 392 74 322 194 456 38 296 176 420 104 364 232 494 22 280 146 412 90 348 216 474 54 314 178 440 116 380 248 3 263 131 391 75 331 203 455 35 295 167 431 103 367 231 483 19 279 155 415 91 351 215 475 51 315 187 439 127 383 247 49 9 8 262 136 390 76 324 196 454 48 294 174 426 102 354 230 488 32 278 148 402 92 **338** 214 476 64 316 180 438 122 370 246 504 13 261 141 389 77 333 205 453 45 293 165 421 101 357 229 493 29 277 157 405 93 341 213 477 61 317 189 437 117 373 245 2 260 1 30 388 78 326 198 452 42 292 172 432 100 360 228 482 26 276 150 408 94 344 212 478 58 318 182 436 128 376 244 498 7 259 135 387 79 335 207 451 39 291 163 427 99 363 227 487 23 275 159 411 95 347 211 479 55 319 191 435 123 379 243 503 12 258 140 386 80 328 200 450 36 290 170 422 98 366 226 492 20 274 152 414 96 350 210 480 52 320 184 434 118 382 242 1 257 129 385 65 321 193 449 33 289 161 417 97 353 225 481 17 273 145 401 81 337 209 465 49 305 **177** 433 113 369 241 497}

Figure A.2: Cdma2000 output data after interleaving.

As depicted in Fig.A.2, the input data has been totally interleaved from their original positions. For instance at position 501 is the element 177 and at position 177 is the element 151, whereas at the same positions in the input vector in Table A.3 are the data 501 and 177 respectively.

Although the output produced by the interleaver seems to have elements of randomness, it should be noticed that the the transformation between input and output positions is both deterministic and bijective. Also, the interleaving process does not mean all positions must change at the output. In the previous example of Fig. A.2, positions 68, 84 and 338 appear at the same positions at the interleaver's output.

Annexe B

State mapping encoding

FOR the 3D turbo code, the choice of the post-encoder has to meet several requirements detailed in chapter 3. We have chosen the code (5,4) as a convenient post-encoder (from the convergence point of view) in different simulations. Unfortunately, this code cannot be made circular directly in order to properly deal with blocks of data. This appendix explains how the state mapping encoding allows to solve easily the problem.

B.1 Circular (tail-biting) encoding

Let $\mathbf{s}_i$ and $\mathbf{d}_i$ be the state of the encoder register, and the encoder data input, respectively, at discrete time i. The encoder state at time $i+1$ is given by the following equation :

$$\mathbf{s}_{i+1} = \mathbf{G}\,\mathbf{s}_i + \mathbf{d}_i$$

where $\mathbf{G}$ is the state matrix of the linear feedback register (LFR). For instance, considering the LFR in Fig. B.1, whose binary input is denoted $\mathbf{d}_i$, we have :

$$\mathbf{s}_i = \begin{bmatrix} \mathbf{s}_{1,i} \\ \mathbf{s}_{2,i} \end{bmatrix}, \mathbf{d}_i = \begin{bmatrix} \mathbf{d}_i \\ 0 \end{bmatrix}, \mathbf{G} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

where $\mathbf{s}_{1,i}$ and $\mathbf{s}_{2,i}$ are the content of the first and the second memory cell of the LFR, respectively.

More generally, for a memory v register, vectors $\mathbf{s}_i$ and $\mathbf{d}_i$ have v components and $\mathbf{G}$ is of size $v \times v$. After the encoding of the data sequence $\{\mathbf{d}_i\}$, of length P, the final state $\mathbf{s}_P$ can be expressed as a function of initial state $\mathbf{s}_0$ and $\{\mathbf{d}_i\}$:

$$\mathbf{s}_P = \mathbf{G}^P\,\mathbf{s}_0 + \sum_{j=1}^{P} \mathbf{G}^{P-j}\,\mathbf{d}_{j-1} \tag{B.1}$$

If it is possible to find a circulation state, denoted $\mathbf{s}^c$, such that $\mathbf{s}^c = \mathbf{s}_0 = \mathbf{s}_p$, this is given by :

$$\mathbf{s}^c = \left[\mathbf{I} + \mathbf{G}^P\right]^{-1} \sum_{j=1}^{P} \mathbf{G}^{P-j}\,\mathbf{d}_{j-1} \tag{B.2}$$

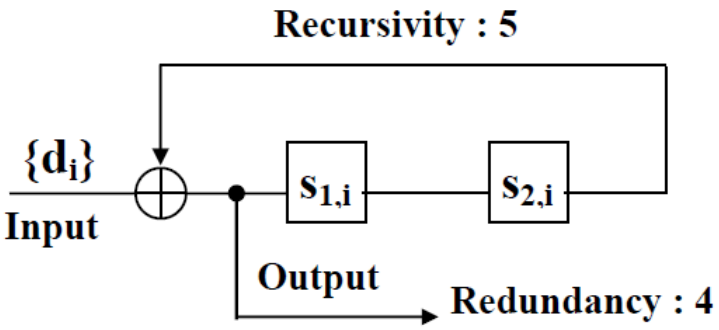


Figure B.1: Selected post-encoder: code (5,4) with memory 2. The recursivity polynomial is 5 and the redundancy polynomial is 4.

where $\mathbf{I}$ is the $v \times v$ identity matrix.

Note that $\mathbf{s}^c$ exists if and only if $\mathbf{I} + \mathbf{G}^P$ is invertible. This condition is never satisfied for some matrixes $\mathbf{G}$, whatever P. For this reason, the LFR of Fig. B.1 cannot directly be made circular.

Before the encoding of $\{\mathbf{d}_i\}$, the knowledge of $\mathbf{s}^c$ requires a preliminary step. The encoder is first set up in the all-zero state, and then fed by the data sequence $\{\mathbf{d}_i\}$. The final state is denoted $\mathbf{s}_P^0$. From equation (B.2), we have

$$\mathbf{s}_P^0 = \sum_{j=1}^{P} \mathbf{G}^{P-j}\, \mathbf{d}_{j-1}$$

and $\mathbf{s}^c$ can then be related to $\mathbf{s}_P^0$ by :

$$\mathbf{s}^c = \left[\mathbf{I} + \mathbf{G}^P\right]^{-1} \mathbf{s}_P^0$$

Finally, the encoder being initialized to the circulation state, the encoding process can really start to provide the redundancy sequence.

B.2 State mapping encoding

State mapping encoding is introduced for cases where standard circular (tail-biting) encoding is not possible. The core of this encoding is a mapping that maps the final state $\mathbf{s}_P$ to the state $\mathbf{s'}_P = \mathbf{A}\,\mathbf{s}_P$ using a state mapping matrix $\mathbf{A}$ [98]. Mapping the final state yields the equation :

$$\mathbf{s'}_P = \mathbf{A}\,\mathbf{G}^P\,\mathbf{s}_0 + \mathbf{A} \sum_{j=1}^{P} \mathbf{G}^{P-j}\,\mathbf{d}_{j-1}$$

Under these conditions, a mapping state $\mathbf{s}^m$ with $(\mathbf{s}^m = \mathbf{s}_P^0 = \mathbf{s'}_P)$ always exists, and it is given by :

$$\mathbf{s}^m = \mathbf{B} \sum_{j=1}^{P} \mathbf{G}^{P-j}\, \mathbf{d}_{j-1} = \mathbf{B}\, \mathbf{s}_P^0 \tag{B.3}$$

with

$$\mathbf{B} = \left[\mathbf{I} + \mathbf{A}\, \mathbf{G}^P\right]^{-1} \mathbf{A} \tag{B.4}$$

In other words, if the encoding starts in the state $\mathbf{s}^m$, the encoding will end in the state $\mathbf{s}^e$ with $\mathbf{s}^m = A\, \mathbf{s}^e$.

Because $\mathbf{G} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, and thanks to the values assigned to $\mathbf{A}$, $\mathbf{I} + \mathbf{A}\, \mathbf{G}^P$ is always invertible. The encoding procedure can be summarized in the following steps :

- ◇ set up the encoder at the all-zero state. Feed it with $\{\mathbf{d}_i\}$ and compute the final state $\mathbf{s}_P^0$,
- ◇ calculate $\mathbf{s}^m$ through equations (B.3) and (B.4),
- ◇ encode $\{\mathbf{d}_i\}$ starting from $\mathbf{s}^m$. If needed, map the final state $\mathbf{s}^e$ using $\mathbf{A}$, in order to verify that the result is $\mathbf{s}^m$ (*i.e.*, $\mathbf{s}^m = \mathbf{A}\, \mathbf{s}^e$).

The post-encoder of Fig. B.1, with generator polynomial 5, can be encoded using $\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ if P is odd, and $\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ if P is even.

The decoding process has to take into account the mapping described above. This is done by an exchange of metrics after having processed last address during the forward recursion, and after having processed first address during the backward recursion, when the MAP algorithm or a simplified version is employed. Table B.1 provides the values of $\mathbf{s'}_P$ obtained through the mapping of $\mathbf{s}_P$. The table also provides the values of $\mathbf{s}^m$ for each $\mathbf{s}_P^0$ using equations (B.3) and (B.4). We can observe that only two (if P is odd) or three (if P is even) metrics need to be swapped during the decoding process, at the extremity of the block, which represents a very small additional complexity for the 4-state decoder.

P odd		P **even**		P odd		P **even**	
$\mathbf{s}_P$	$\mathbf{s'}_P$	$\mathbf{s}_P$	$\mathbf{s'}_P$	$\mathbf{s}_P^0$	$\mathbf{s}^m$	$\mathbf{s}_P^0$	$\mathbf{s}^m$
0	0	0	0	0	0	0	0
1	3	**1**	**2**	1	1	**1**	**3**
2	2	**2**	**3**	2	3	2	1
3	1	**3**	**1**	3	2	3	2

Table B.1: Corresponding values $\mathbf{s}_P$ of and $\mathbf{s'}_P$, and of $\mathbf{s}_P^0$ and $\mathbf{s}^m$.

B.3 Example

Let us assume that we have $P = 384$ bits to postcode. The post-encoder is then initialized to the all-zero state. For reasons of simplification, we assume also that all the data to be post-encoded are equal to one ($d_i = 1$). The data $\{d_i\}$ in the entry of the pre-decoder allow to calculate the final state $\mathbf{s}_P^0$. Here, we obtain $\mathbf{s}_P^0 = 1$. As P is even, the circulation state is $\mathbf{s}^m = 3$ according to Table B.1. Therefore, the data $\{d_i\}$ are coded starting from the state 3, and we check that the final state is state 2.

At the end of the backward recursion of the pre-decoding process (*i.e.* after after having processed first address $i = 1$), three metrics have to be exchanged. The minimum metric of state 2 is replaced by that of state 3, the minimum metric of state 1 is replaced by that of state 2 and the minimum metric of state 2 is replaced by that of state 1 as follows :

$$\begin{cases} 2 \leftarrow 3 \\ 1 \leftarrow 2 \\ 3 \leftarrow 1 \end{cases}$$

By this way, the smallest metric is transposed from state 3 into state 2. And it is as if the code, having started the backward recursion from state 2, reached at the end the same state (2 instead of the state 3).

Besides, at the end of the forward recursion of the pre-decoding process (*i.e.* after after having processed last address $i = P$), three metrics have to be exchanged. The minimum metric of state 2 is replaced by that of state 1, the minimum metric of state 1 is replaced by that of state 3 and the minimum metric of state 3 is replaced by that of state 2 :

$$\begin{cases} 2 \leftarrow 1 \\ 1 \leftarrow 3 \\ 3 \leftarrow 2 \end{cases}$$

Now, the smallest metric is transposed from state 2 into state 3. And it is as if the code, having started the forward recursion from state 3, reached at the end the same state (3 instead of the state 2).

Note that the metrics need to be swapped during the decoding process, at the extremity of the block, for each iteration. So, the pre-decoder is updated in a convenient state to begin the following iteration.

Appendix C

About 16-QAM and 8-PSK modulations

LET us consider a 16-QAM scenario. Each constellation point can represent $\log_2(16) = 4$ bits, with two bits on the I axis and two on the Q axis. For 16-QAM, the values taken by the I and Q axes are $\{-3, -1, +1, +3\}$. The two bits on the I and Q arms can be Gray coded as shown in the table below.

I	$b_0\, b_1$	Q	$b_3\, b_2$
-3	01	-3	01
-1	00	-1	00
$+1$	10	$+1$	10
$+3$	11	$+3$	11

Table C.1: Gray coded constellation mapping for 16-QAM.

The corresponding constellation diagram with the bit mapping is shown in Fig. C.1. As can be observed from the figure, the adjacent constellation symbols differ by only one bit. Consequently, if the noise causes the constellation to cross the decision threshold, only one out of four bits will be in error. Then, the relation between the bit error rate P_b and symbol error rate P_s can be expressed in the following way : $P_b \approx \frac{P_s}{4}$.

Let us consider a transmission over a zero-mean Gaussian channel with variance σ^2. The probability density function is:

$$p(x) = \frac{1}{\sqrt{2\Pi}\sigma}\, e^{-\frac{x^2}{2}}$$

We are interested in the error probability of each bit in the symbol. As the axes I and Q are independently coded, the results obtained for an axis can be directly applied to the other one. In the sequel, we only consider the I axis represented in Fig C.2.

In fact, the easiest way of obtaining the approximate BER is to view 16-QAM as two orthogonal independent 4-PAM schemes with 4 levels each. Note that this analysis is only valid for square QAM modulation schemes such as the 16-QAM example of Fig. C.1.

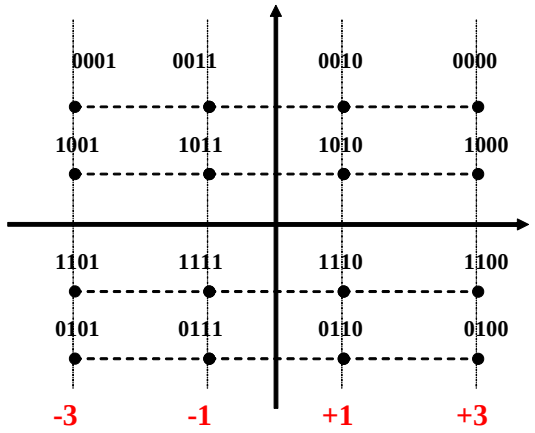


Figure C.1: 16-QAM constellation with Gray coded mapping.

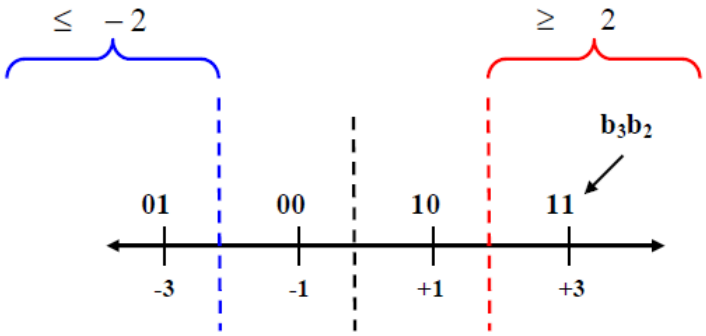


Figure C.2: In-phase axis of a 16-QAM modulation with Gray mapping.

Let x_I be the coordinate of the emitted symbol on the I axis. The probabilities to be considered in the different positions are expressed according to the Marcum function $Q(x)$[1] :

$$P_e\left(b_3 \diagdown x_I = \pm 3\right) = \int_0^{+\infty} \frac{1}{\sqrt{2\Pi}\sigma} e^{-\frac{(x+3/\sigma)^2}{2\times\sigma^2}} dx = \frac{1}{\sqrt{2\Pi}\sigma} \int_{3/\sigma}^{+\infty} e^{-\frac{x^2}{2\times\sigma^2}} = Q\left(\frac{3}{\sigma}\right)$$

$$P_e\left(b_3 \diagdown x_I = \pm 1\right) = \int_0^{+\infty} \frac{1}{\sqrt{2\Pi}\sigma} e^{-\frac{(x+1/\sigma)^2}{2\times\sigma^2}} dx = \frac{1}{\sqrt{2\Pi}\sigma} \int_{1/\sigma}^{+\infty} e^{-\frac{x^2}{2\times\sigma^2}} dx = Q\left(\frac{1}{\sigma}\right)$$

$$P_e\left(b_2 \diagdown x_I = \pm 3\right) = \int_{-2/\sigma}^{+2/\sigma} \frac{1}{\sqrt{2\Pi}\sigma} e^{-\frac{(x+3/\sigma)^2}{2\times\sigma^2}} dx = \frac{1}{\sqrt{2\Pi}\sigma} \left(\int_{-2/\sigma}^{+\infty} e^{-\frac{(x+3/\sigma)^2}{2\times\sigma^2}} dx - \int_{+2/\sigma}^{+\infty} e^{-\frac{(x+3/\sigma)^2}{2\times\sigma^2}} dx \right)$$

[1] $Q(x) = \frac{1}{\sqrt{2\Pi}} \int_x^{+\infty} e^{-\frac{u^2}{2}} du = \frac{1}{\sqrt{2\Pi}\sigma} \int_{x/\sigma}^{+\infty} e^{-\frac{x^2}{2\times\sigma^2}} dx = \frac{1}{2} erfc\left(\frac{x}{\sqrt{2}\sigma}\right)$

$$= \frac{1}{\sqrt{2\Pi}\sigma} \left(\int_{+1/\sigma}^{+\infty} e^{-\frac{x^2}{2\times\sigma^2}} dx - \int_{+5/\sigma}^{+\infty} e^{-\frac{x^2}{2\times\sigma^2}} dx \right) = Q\left(\frac{1}{\sigma}\right) - Q\left(\frac{5}{\sigma}\right)$$

$$P_e\left(b_2 \diagdown x_I = \pm 1\right) = \frac{1}{\sqrt{2\Pi}\sigma} \left(\int_{-\infty}^{-2/\sigma} e^{-\frac{(x+1/\sigma)^2}{2\times\sigma^2}} dx + \int_{+2/\sigma}^{+\infty} e^{-\frac{(x+1/\sigma)^2}{2\times\sigma^2}} dx \right)$$

$$= \frac{1}{\sqrt{2\Pi}\sigma} \left(\int_{-\infty}^{-1/\sigma} e^{-\frac{x^2}{2\times\sigma^2}} dx + \int_{+3/\sigma}^{+\infty} e^{-\frac{x^2}{2\times\sigma^2}} dx \right)$$

$$= \frac{1}{\sqrt{2\Pi}\sigma} \left(\int_{+1/\sigma}^{+\infty} e^{-\frac{x^2}{2\times\sigma^2}} dx + \int_{+3/\sigma}^{+\infty} e^{-\frac{x^2}{2\times\sigma^2}} dx \right) = Q\left(\frac{1}{\sigma}\right) + Q\left(\frac{3}{\sigma}\right)$$

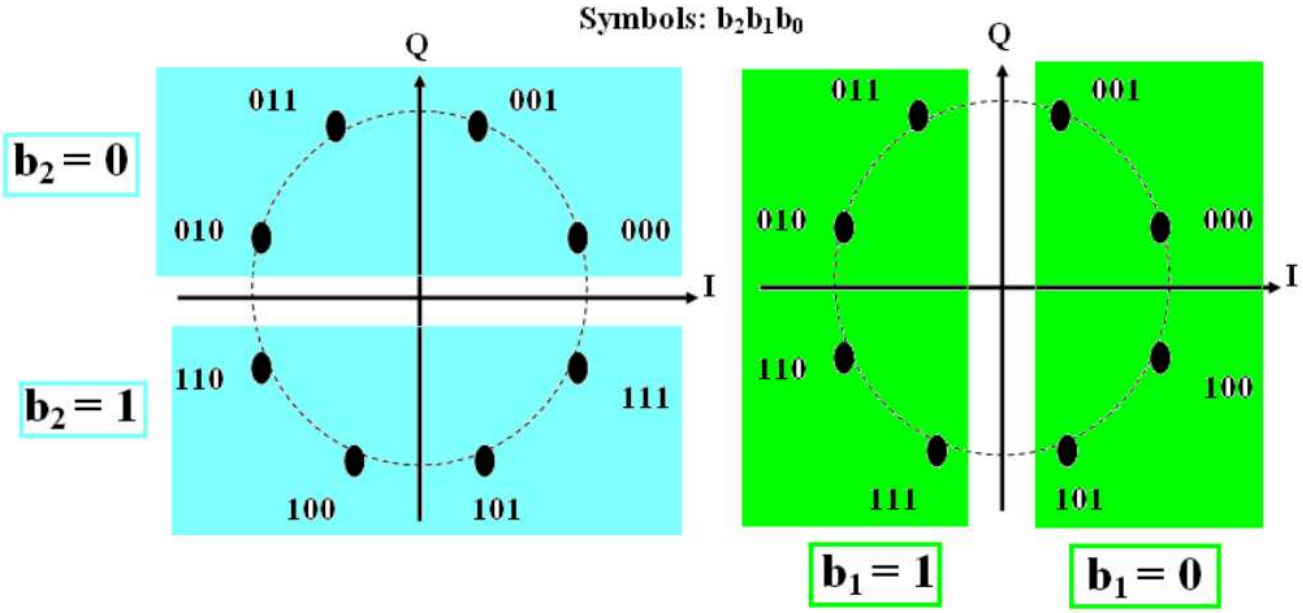
By noticing that $Q\left(\frac{1}{\sigma}\right) \gg Q\left(\frac{3}{\sigma}\right) \gg Q\left(\frac{5}{\sigma}\right)$, these probabilities can be approximated as follows :

$$P_e\left(b_3 \diagdown x_I = \pm 3\right) = Q\left(\frac{3}{\sigma}\right)$$

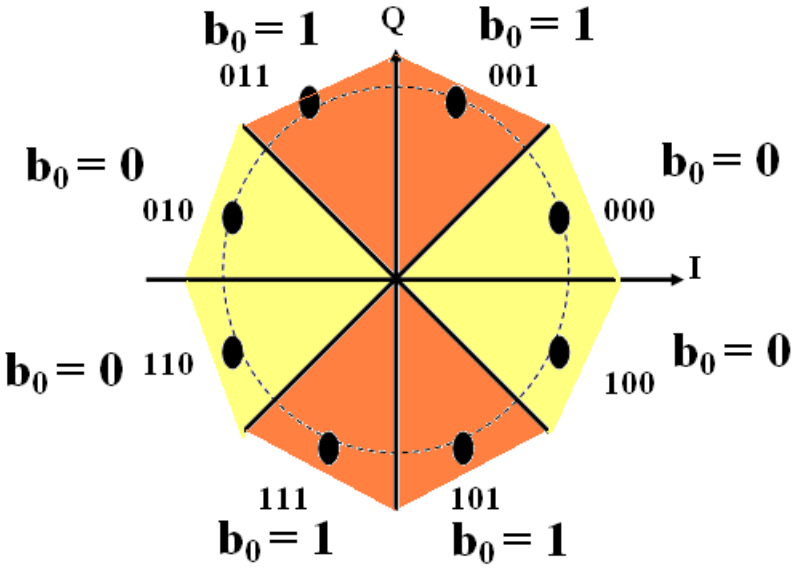
$$P_e\left(b_3 \diagdown x_I = \pm 1\right) = Q\left(\frac{1}{\sigma}\right) \simeq P_e\left(b_2 \diagdown x_I = \pm 3\right) \simeq P_e\left(b_2 \diagdown x_I = \pm 1\right)$$

Then, we can conclude that in the 16-QAM modulation the binary positions b_3 and b_1 are better protected than the binary positions b_2 and b_0. We took advantage of this property to protect the systematic bits as well as the post-encoded parity bits as a priority. And this configuration reduced the loss of convergence of 3D turbo codes.

The reasoning is similar in the case of an 8-PSK modulation.We can also show that among the three bits forming a symbol in 8-PSK, the average probability of error is smaller for the first and the second bits than for the third bit. Nevertheless, this properties can be directly deduced from the constellation in Fig. C.3. The represented decision zones already predict that the average probability of error is smaller for bits b_2 and b_1 than for the bit b_0.



(a) Decision zones for bits b_2 and b_1.



(b) Decision zones for the LSB bit b_0.

Figure C.3: Decision zones for an 8-PSK constellation with Gray coded mapping.

Appendix D

Basics and properties of LDPC codes

Low density parity check (LDPC) codes were originally invented by Robert Gallager in 1963 [54]. Although they perform near the Shannon limit for standard additive white Gaussian noise channels, LDPC codes were ignored for a long time due to the requirement of high complexity computation and the introduction of Reed-Solomon codes [84]. They were rediscovered in the mid-1990's by MacKay and Neal[72].

D.1 Encoding and iterative decoding of LDPC codes

Like all linear block codes, the structure of LDPC codes can be described by the generator matrix G or the parity check matrix H with dimension $n \times m$. H is sparse and contains only a few 1's in each row and column (in comparison to the amount of 0's). Let w_r be the number of 1's in each row and w_c for the columns. For a low-density matrix, two conditions must be satisfied: $w_c \ll m$ and $w_r \ll n$. In this case large minimum distance is expected, as it represents the least number of columns in H that sum up to zero.

An alternative approach to simplified encoding is to design the LDPC codes via algebric or geometric methods. Tanner introduced an effective graphical representation for LDPC codes [102]. Tanner graphs are bipartite graphs. That means that the nodes of the graph are separated into two classes, where edges only connect two nodes of different classes. The two types of nodes in a Tanner graph are called variable nodes (v-nodes) and check nodes (c-nodes). In other words, nodes of the same type cannot be connected (*e.g.* a c-node cannot be connected to another c-node).

Let us look at an example for a low-density parity-check matrix H defined in equation (D.1) for a (8,4) product code. Fig. D.1 represents the corresponding Tanner graph. It consists of $m = 4$ check nodes (the number of parity bits) and $n = 8$ variable nodes (the number of bits in a codeword). Check node f_i is connected to variable node c_j if the element h_{ij} of H is equal to 1.

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{D.1}$$

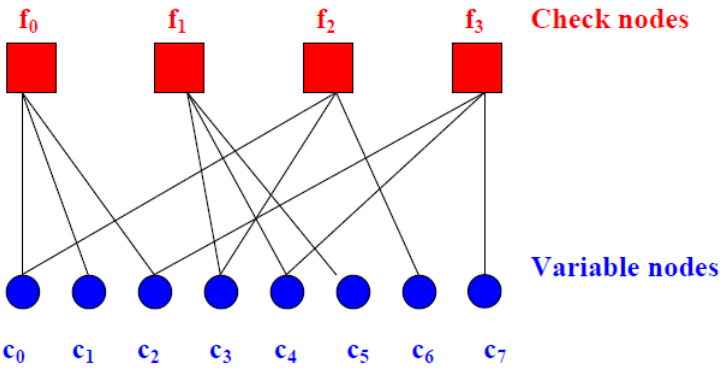


Figure D.1: Tanner graph corresponding to the parity check matrix of equation (D.1).

Like turbo codes, LDPC can be decoded iteratively. Instead of a trellis, the decoding takes place on a Tanner graph where messages are exchanged between the v-nodes and c-nodes. The edges of the graph act as information pathways. Many graph-based algorithms are used in the decoding process. The most common ones are the sum-product algorithm used for general graph-based codes, the MAP (BCJR) algorithm for trellis graph-based codes and the message passing algorithm for bipartite graph-based codes.

D.2 Irregular LDPC codes

A LDPC code is called regular if w_c is constant for every column and H contains exactly $w_r = w_c\,(n/m)$ 1's per row. Otherwise, the code is irregular (*i.e.* H is low density but the rows and columns have non-uniform weight).

It's also possible to see the regularity of this code while looking at the graphical representation. There is the same number of incoming edges for every v-node and also for all the c-nodes.

Although regular codes have impressive performance, they are still about 1 dB from capacity and generally perform worse than turbo codes. Irregular LDPC codes tend to outperform turbo codes for large block lengths (of more than 10^5 bits).

Bibliography

[1] *DVB, Interaction channel for satellite distribution systems*, 2000, ETSI EN 301 790, v. 1.2.2.

[2] *DVB, Interaction channel for digital terrestrial television*, 2001, ETSI EN 301 958, v. 1.1.1.

[3] *Third Generation Partnership Project 2 (3GPP2), Physical layer standard for cdma2000 spread spectrum systems, Release D*, February 2004, 3GPP2 C.S0002-D, Version 1.0.

[4] *Third Generation Partnership Project (3GPP) Technical SpecificationGroup, Multiplexing and channel coding (FDD)*, June 1999, TS 25.212, v2.0.0.

[5] *IEEE standard for local and metropolitan area networks. Part 16: air interface for fixed broadband wireless access systems*, November 2004, IEEE 802.16-2004.

[6] Error correcting coding diversity for rayleigh fading channels. *San Jose State University*, EE252, Spring 2004.

[7] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate (Corresp.). *IEEE Transactions on Information Theory*, 20(2):284–287, March 1974.

[8] A. Barbulescu and S. Pietrobon. Interleaver design for turbo codes. *Electronics Letters*, 30(25):2107–2108, December 1994.

[9] G. Battail. A conceptual framework for understanding turbo codes. *IEEE Journal on Selected Areas in Communications*, 16(2):245–254, February 1998.

[10] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara. Soft-output decoding algorithms in iterative decoding of turbo codes. *JPL TDA Progress Report*, 42:63–87, February 1996.

[11] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara. Serial concatenation of interleaved codes: Performance analysis, design, and iterative decoding. *IEEE Transactions on Information Theory*, 44(3):909–926, May 1998.

[12] S. Benedetto and G. Montorsi. Iterative decoding of serially concatenated convolutional codes. *Electronics letters*, 32(13):1186–1188, June 1996.

[13] S. Benedetto and G. Montorsi. Unveiling turbo codes: Some results on parallel concatenated coding schemes. *IEEE Transactions on Information Theory*, 42(2):409–428, March 1996.

[14] S. Benedetto and G. Montorsi. Serial concatenation of block and convolutional codes. *Electronics Letters*, 32(10):887–888, May 1996.

[15] C. Berrou. The ten-year-old turbo codes are entering into service. *IEEE Communications Magazine*, 41(8):110–116, August 2003.

[16] C. Berrou and A. Glavieux. Reflections on the Prize Paper: Near optimum error correcting coding and decoding: Turbo-Codes. *IEEE Information Theory Society Newsletter*, 48(2):24–31, June 1998.

[17] C. Berrou and A. Glavieux. Near optimum error correcting coding and decoding: Turbo-codes. *IEEE Transactions on Communications*, 44(10):1261–1271, October 1996.

[18] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo-codes. In *Proc. ICC'93, Geneva, Switzerland*, volume 2, pages 1064–1070, May 1993.

[19] C. Berrou, A. Graell i Amat, and Y. Ould-Cheikh-Mouhamedou. About rate-1 codes as inner codes. In *Proc. 5th International Symposium on Turbo Codes and Related Topics*, September 2008.

[20] C. Berrou, G. i Amat, O. Mouhamedou, C. Douillard, and Y. Saouter. Adding a rate-1 third dimension to turbo codes. In *Proc. ITW 2007 : IEEE Information Theory Workshop, September 2-6, Lake Tahoe, CA, USA*, pages 156–161, 2007.

[21] C. Berrou, G. i Amat, and Y. Saouter. Improving the distance properties of turbo codes using a third component code: 3D turbo codes. *IEEE Transactions on Communications*, 57(9):2505–2509, September 2009.

[22] C. Berrou, Y. Saouter, C. Douillard, S. Kerouédan, and M. Jézéquel. Designing good permutations for turbo codes: towards a single model. In *Proc. ICC'04*, volume 1, pages 341–345, June 2004.

[23] C. Berrou, S. Vaton, M. Jézéquel, and C. Douillard. Computing the minimum distances of linear codes by the error impulse method. In *Proc. IEEE Global Communication Conference (Globecom'2002), Taipei, Taiwan*, pages 1017–1020, November 2002.

[24] E. Boutillon and D. Gnaedig. Maximum spread of D-dimensional multiple turbo codes. *IEEE Transactions on Communications*, 53(8):1237–1242, August 2005.

[25] J. Boutros. Asymptotic behavior study of irregular turbo codes. *7th International Workshop on DSP Tech. for Space Communications, Sésimbra, Portugal*, October 2001.

[26] J. Boutros, G. Caire, E. Viterbo, H. Sawaya, and S. Vialle. Turbo code at 0.03 dB from capacity limit. In *Proc. IEEE International Symposium on Information Theory*, page 56, July 2002.

[27] F. Brannstrom, L. Rasmussen, and A. Grant. Convergence analysis and optimal scheduling for multiple concatenated codes. *IEEE Transactions on Information Theory*, 51(9):3354–3364, September 2005.

[28] G. Caire, G. Taricco, and E. Biglieri. Bit-interleaved coded modulation. *IEEE Transactions on Information Theory*, 44(3):927–946, 1998.

[29] S. Chung, G. Forney Jr, T. Richardson, and R. Urbanke. On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit. *IEEE Communications Letters*, 5(2):58–60, February 2001.

[30] S. Chung, T. Richardson, and R. Urbanke. Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation. *IEEE Transactions on Information Theory*, 47(2):657–670, February 2001.

[31] H. Claussen, H. Karimi, and B. Mulgrew. Improved max-log map turbo decoding using maximum mutual information combining. In *Proc. 14th IEEE on Personal Indoor and Mobile Radio Communications, (PIMRC)*, volume 1, pages 424–428, 2003.

[32] D. Costello and G. Forney. Channel coding: The road to channel capacity. *Proceedings of the IEEE*, 95(6):1150–1177, June 2007.

[33] D. Costello Jr. Free distance bounds for convolutional codes. *IEEE Transactions on Information Theory*, 20(3):356–365, May 1974.

[34] S. Crozier. New high-spread high-distance interleavers for turbo-codes. In *Proc. 20th Biennial Symposium on Communications, Kingston, Ontario, Canada*, pages 3–7, May 2000.

[35] S. Crozier and P. Guinand. Distance upper bounds and true minimum distance results for turbo-codes designed with DRP interleavers. In *Annals of telecommunications*, volume 60, pages 10–28. Springer, February 2005.

[36] S. Crozier and P. Guinand. High-performance low-memory interleaver banks for turbo-codes. In *Proc. IEEE 54th Vehicular Technology Conference (VTC 2001 Fall), Atlantic City, NJ*, volume 4, pages 2394 –2398, October 2001. US Patent App. 10/165,122.

[37] S. Crozier, P. Guinand, and A. Hunt. Estimating the minimum distance of turbo-codes using double and triple impulse methods. *IEEE Communications Letters*, 9(7):631–633, July 2005.

[38] S. Crozier, P. Guinand, and A. Hunt. Computing the minimum distance of turbo-codes using iterative decoding techniques. In *Proc. 22th Biennial Symposium on Communications, Kingston, Ontorio, Canada*, pages 306–308, June 2004.

[39] S. Crozier, J. Lodge, P. Guinand, and A. Hunt. Performance of Turbo-codes with relative prime and golden interleaving strategies. In *Proc. 6th International Mobile Satellite Conference (IMSC'99), Ottawa, Canada*, pages 268–275, June 1999.

[40] R. Deshmukh and S. Ladhake. Analysis of Various Puncturing Patterns and Code Rates: Turbo Code. *International Journal of Electronics Engineering Research*, 1(2):79–88, 2009.

[41] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[42] D. Divsalar, H. Jin, and R. McEliece. Coding theorems for" turbo-like" codes. In *Proc. 36th Allerton Conference on Communications, Control, and Computing*, pages 201–210, 1998.

[43] D. Divsalar and F. Pollara. Multiple turbo codes for deep-space communications. *TDA progress report*, 42:121, 1995.

[44] D. Divsalar and F. Pollara. Hybrid concatenated codes and iterative decoding. In *Proc. ISIT'97, Ulm, Germany*, page 10, July 1997.

[45] D. Divsalar and F. Pollara. Multiple turbo codes. In *Proc. IEEE Military Communications Conference (MILCOM'95), San Diego, California*, volume 1, pages 279–285, November 1995.

[46] D. Divsalar and F. Pollara. Serial and hybrid concatenated codes with applications. In *Proc. International Symposium on Turbo Codes and Related Topics, Brest, France*, pages 80–87, September 1997.

[47] S. Dolinar and D. Divsalar. Weight distributions for turbo codes using random and nonrandom permutations. *TDA Progress report*, 42(122):56–65, 1995.

[48] C. Douillard and C. Berrou. Turbo codes with rate-m/(m+ 1) constituent convolutional codes. *IEEE Transactions on Communications*, 53(10):1630–1638, 2005.

[49] H. El Gamal and J. Hammons, A.R. Analyzing the turbo decoder using the Gaussian approximation. *IEEE Transactions on Information Theory*, 47(2):671–686, February 2001.

[50] J. Forney G.D. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, March 1973.

[51] B. Frey and D. MacKay. Irregular turbocodes. In *Proc. 37th Allerton Conference on Communication, Control and Computing, Illinois*, page 121, September 1999.

[52] B. Frey and D. MacKay. Irregular turbo-like codes. In *Proc. 2nd International Symposium on Turbo Codes and Related Topics, Brest, France*, pages 67–72, September 2000.

[53] R. Gallager. Low-density parity-check codes. *IRE Transactions on Information Theory*, 8(1):21–28, January 1962.

[54] R. Gallager and L. Codes. Cambridge, MA. *MIT Press*, 5:619–637, 1963.

[55] R. Garello and V. Casado. The all-zero iterative decoding algorithm for turbo code minimum distance computation. In *Proc. ICC'04, Paris, France*, pages 361–364, June 2004.

[56] H. Gonzalez, C. Berrou, and S. Kerouédan. Serial/Parallel Turbo Codes for Low Error Rates. In *Proc. ICC'04, Paris, France*, volume 1, pages 346–350, June 2004.

[57] J. Hagenauer. The EXIT chart-introduction to extrinsic information transfer in iterative processing. In *Proc. 12th European Signal Processing Conference (EUSIPCO)*, pages 1541–1548, September 2004.

[58] J. Hagenauer, E. Offer, and L. Papke. Iterative decoding of binary block and convolutional codes. *IEEE Transactions on Information Theory*, 42(2):429–445, March 1996.

[59] M. Ho, S. Pietrobon, and T. Giles. Interleavers for punctured turbo codes. *IEEE Asia-Pacific Conf. Commun. and Singapore Int. Conf. Commun. Syst. (APCC/ICC),Singapore, Malaysia*, 2:520–524, November 1998.

[60] J. Hokfelt, O. Edfors, and T. Maseng. Interleaver design for turbo codes based on the performance of iterative decoding. In *Proc. ICC'99*, volume 1, pages 93–97, 1999.

[61] J. Hokfelt, O. Edfors, and T. Maseng. A turbo code interleaver design criterion based on the performance of iterative decoding. *IEEE Communications Letters*, 5(2):52–54, February 2001.

[62] Q. Hu and L. Perez. Some periodic time-varying convolutional codes with free distance achieving the Heller bound. In *IEEE International Symposium on Information Theory*, page 247, 2001.

[63] X. Hu, E. Eleftheriou, and D. Arnold. Regular and irregular progressive edge-growth Tanner graphs. *IEEE Transactions on Information Theory*, 51(1):386–398, January 2005.

[64] G. i Amat and E. Rosnes. Analysis of 3-dimensional turbo code ensembles. In *Proc. IEEE Information Theory Workshop (ITW)*, pages 46–50, January 2010.

[65] G. i Amat and E. Rosnes. Stopping set analysis of 3-dimensional turbo code ensembles. In *Proceedings of IEEE International Symposium on Information Theory (ISIT)*, pages 1814–1818, July 2009.

[66] S. Jamali and T. Le-Ngoc. *Coded-modulation techniques for fading channels*. Springer, 1994.

[67] G. Kraidy and V. Savin. Capacity-approaching irregular turbo codes for the binary erasure channel. *IEEE Transactions on Communications*, 58(9):2516–2524, September 2010.

[68] P. Lee. There are many good periodically time-varying convolutional codes. *IEEE Transactions on Information Theory*, 35(2):460–463, March 1989.

[69] T. Lehngik-Emeden, M. Alles, and N. Wehn. 3d duo binary turbo decoder hardware implementation. 2009.

[70] J. Li, K. Narayanan, and C. Georghiades. Product accumulate codes: a class of codes with near-capacity performance and low decoding complexity. *IEEE Transactions on Information Theory*, 50(1):31–46, January 2004.

[71] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman. Improved low-density parity-check codes using irregular graphs. *IEEE Transactions on Information Theory*, 47:671–686, 2000.

[72] D. MacKay. Good error-correcting codes based on very sparse matrices. *IEEE Transactions on Information Theory*, 45:399–431, 1999.

[73] D. MacKay and R. Neal. Near Shannon limit performance of low density parity check codes. *Electronics letters*, 32(18):1645–1646, 1996.

[74] D. MacKay, S. Wilson, and M. Davey. Comparison of constructions of irregular Gallager codes. *IEEE Transactions on Communications*, 47(10):1449–1454, 1999.

[75] J. Massey. Reversible codes. *Information and Control*, 7(3):369–380, 1964.

[76] M. Mooser. Some periodic convolutional codes better than any fixed code. *IEEE Transactions on Information Theory*, 29(5):750–751, September 1983.

[77] O. Muller, A. Baghdadi, and M. Jézéquel. On the parallelism of convolutional turbo decoding and interleaving interference. In *Proc. IEEE Global Conference on Communications (GLOBECOM), San Francisco, California, USA*, November 2006.

[78] A. Nimbalker, Y. Blankenship, B. Classon, and T. Blankenship. ARP and QPP interleavers for LTE turbo coding. In *Proc. IEEE Wireless Communications and Networking Conference (WCNC 2008), Las Vegas, NV, USA*, pages 1032–1037, March 2008.

[79] R. Palazzo. A time-varying convolutional encoder better than the best time-invariant encoder. *IEEE Transactions on Information Theory*, 39(3):1109–1110, May 1993.

[80] L. Perez, J. Seghers, and D. Costello Jr. A distance spectrum interpretation of turbo codes. *IEEE Transactions on Information Theory*, 42(6):1698–1709, 1996.

[81] S. Pietrobon. *Interleaver Address Generator*. Small World Communications, Version 1.01, October 1998.

[82] J. Proakis and M. Salehi. *Digital communications*, volume 5. McGraw-hill, New York, 2004.

[83] R. Pyndiah. Near-optimum decoding of product codes: Block turbo codes. *IEEE Transactions on Communications*, 46(8):1003–1010, 1998.

[84] I. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics (SIAM)*, 8(2):300–304, June 1960.

[85] T. Richardson, A. Shokrollahi, and R. Urbanke. Design of provably good low-density parity check codes. In *Proc. IEEE International Symposium on Information Theory*, page 199, 2000.

[86] T. Richardson, M. Shokrollahi, and R. Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *IEEE Transactions on Information Theory*, 47(2):619–637, February 2001.

[87] T. Richardson and R. Urbanke. Efficient encoding of low-density parity-check codes. *IEEE Transactions on Information Theory*, 47(2):638–656, February 2001.

[88] T. Richardson and R. Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Transactions on Information Theory*, 47(2):599–618, February 2001.

[89] P. Robertson, E. Villebrun, and P. Hoeher. A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain. In *Proc. ICC'95, Seattle,'Gateway to Globalization'*, volume 2, pages 1009–1013, June 1995.

[90] Y. Saouter. Selection procedure of turbocode parameters by combinatorial optimization. In *Proc. 6th International Symposium on Turbo Codes and Iterative Information Processing (ISTC), Brest, France*, pages 156–160, September 2010.

[91] P. Sauvé. Multibit decoding of turbo codes. Master's thesis, University of Toronto, Canada, October 1998.

[92] H. Sawaya and J. Boutros. Irregular turbo-codes with symbol-based iterative decoding. In *Proc. 3rd International Symposium on Turbo-codes, Brest, France*, pages 407–410, September 2003.

[93] H. Sawaya, S. Vialle, J. Boutros, and G. Zémor. Performance Limits of Compound Codes with Symbol-Based Iterative Decoding. *Electronic Notes in Discrete Mathematics*, 6:433–443, 2001.

[94] H. E. Sawaya. *Performance optimization for capacity-approaching channel coding schemes*. PhD thesis, ENST Paris, France, March 2002.

[95] C. Schlegel and L. Perez. *Trellis and turbo coding*. IEEE/Wiley, Piscataway, NJ, USA, 2004.

[96] C. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, January 2001.

[97] J. Sun and O. Takeshita. Interleavers for turbo codes using permutation polynomials over integer rings. *IEEE Transactions on Information Theory*, 51(1):101–119, January 2005.

[98] J. Sun and O. Takeshita. Extended tail-biting schemes for turbo codes. *IEEE Communications Letters*, 9(3):252–254, March 2005.

[99] T. Synchronization and C. Coding. Recommendation for Space Data System Standards. Technical report, CCSDS 131.0-B-1. Blue Book.

[100] O. Takeshita, O. Collins, P. Massey, and D. Costello Jr. A note on asymmetric turbo-codes. *IEEE Communications Letters*, 3(3):69–71, 1999.

[101] B. Talibart and C. Berrou. Notice Preliminaire du Circuit Turbo-Codeur/Decodeur TURBO4. *Version 0.0, June*, 1995.

[102] R. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, September 1981.

[103] S. ten Brink. Convergence of iterative decoding. *Electronics Letters*, 35(10):806–808, 1999.

[104] S. ten Brink. Iterative decoding trajectories of parallel concatenated codes. *3rd IEEE/ITG Conf. Source and Channel Coding*, pages 75–80, 2000.

[105] S. ten Brink. Code doping for triggering iterative decoding convergence. In *Proc. IEEE International Symposium on Information Theory*, page 235, 2001.

[106] S. ten Brink. Convergence behavior of iteratively decoded parallel concatenated codes. *IEEE Transactions on Communications*, 49(10):1727–1737, 2001.

[107] S. ten Brink. Convergence of multidimensional iterative decoding schemes. In *Proceedings of IEEE 35th Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 270–274, 2001.

[108] A. Viterbi. Convolutional codes and their performance in communication systems. *IEEE Transactions on Communication Technology*, 19(5):751–772, 1971.

[109] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, April 1967.

[110] J. Vogt and A. Finger. Improving the max-log-MAP turbo decoder. *Electronics letters*, 36(23):1937–1939, November 2000.

[111] B. Vucetic and J. Yuan. *Turbo codes: principles and applications*. Springer Netherlands, 2000.

[112] C. Weiss, C. Bettstetter, and S. Riedel. Code construction and decoding of parallel concatenated tail-biting codes. *IEEE Transactions on Information Theory*, 47(1):366–386, 2001.

[113] J. Yuan, B. Vucetic, and W. Feng. Combined turbo codes and interleaver design. *IEEE Transactions on Communications*, 47(4):484–487, 1999.

[114] J. Zhang and M. Fossorier. Shuffled iterative decoding. *IEEE Transactions on Communications*, 53(2):209–213, February 2005.