

THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Adaptive Concatenated Coding for Wireless Real-Time Communications

ELISABETH UHLEMANN

School of Information Science,
Computer and Electrical Engineering
HALMSTAD UNIVERSITY

Department of Computer Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2004

Adaptive Concatenated Coding for Wireless Real-Time Communications

Elisabeth Uhlemann

ISBN 91-7291-516-1

Copyright © Elisabeth Uhlemann, 2004.

All rights reserved.

Doktorsavhandlingar vid Chalmers tekniska högskola

Ny serie nr. 2198

ISSN 0346-718X

School of Computer Science and Engineering

Chalmers University of Technology

Technical Report No. 29D

ISSN 1651-4971

Contact Information:

Department of Computer Engineering

Chalmers University of Technology

SE-412 96 Göteborg, Sweden

Telephone: +46 (0)31 772 1000

Fax: +46 (0)31 772 3663

URL: <http://www.ce.chalmers.se>

School of Information Science,

Computer and Electrical Engineering

Halmstad University

Box 823

SE-301 18 Halmstad, Sweden

Telephone: +46 (0)35 16 71 00

Fax: +46 (0)35 12 03 48

URL: <http://www.hh.se/ide>

Printed by Chalmers Reproservice

Göteborg, Sweden, September 2004.

Adaptive Concatenated Coding for Wireless Real-Time Communications

ELISABETH UHLEMANN

Department of Computer Engineering, Chalmers University of Technology

Abstract

The objective of this thesis is to improve the performance of real-time communication over a wireless channel, by means of specifically tailored channel coding. The deadline dependent coding (DDC) communication protocol presented here lets the timeliness and the reliability of the delivered information constitute quality of service (QoS) parameters requested by the application. The values of these QoS parameters are transformed into actions taken by the link layer protocol in terms of adaptive coding strategies.

Incremental redundancy hybrid automatic repeat request (IR-HARQ) schemes using rate compatible punctured codes are appealing since no repetition of previously transmitted bits is made. Typically, IR-HARQ schemes treat the packet lengths as fixed and maximize the throughput by optimizing the puncturing pattern, i.e. the order in which the coded bits are transmitted. In contrast, we define an IR strategy as the maximum number of allowed transmissions and the number of code bits to include in each transmission. An approach is then suggested to find the optimal IR strategy that maximizes the average code rate, i.e., the optimal partitioning of $n - k$ parity bits over at most M transmissions, assuming a given puncturing pattern. Concatenated coding used in IR-HARQ schemes provides a new array of possibilities for adaptability in terms of decoding complexity and communication time versus reliability. Hence, critical reliability and timing constraints can be readily evaluated as a function of available system resources. This in turn enables quantifiable QoS and thus negotiable QoS. Multiple concatenated single parity check codes are chosen as example codes due to their very low decoding complexity. Specific puncturing patterns for these component codes are obtained using union bounds based on uniform interleavers. The puncturing pattern that has the best performance in terms of frame error rate (FER) at a low signal-to-noise ratio (SNR) is chosen. Further, using extrinsic information transfer (EXIT) analysis, rate compatible puncturing ratios for the constituent component code are found. The puncturing ratios are chosen to minimize the SNR required for convergence.

The applications targeted in this thesis are not necessarily replacement of cables in existing wired systems. Instead the motivation lies in the new services that wireless real-time communication enables. Hence, communication within and between cooperating embedded systems is typically the focus. The resulting IR-HARQ-DDC protocol presented here is an efficient and fault tolerant link layer protocol foundation using adaptive concatenated coding intended specifically for wireless real-time communications.

Keywords: Incremental redundancy hybrid ARQ, multiple concatenated codes, iterative decoding, rate compatible punctured codes, union bounds, EXIT charts, convergence analysis, wireless real-time communication, quality of service.

List of Publications

This thesis is partly based on the publications listed below.

Elisabeth Uhlemann, Lars K. Rasmussen, Alex J. Grant and Per-Arne Wiberg, "Frame length optimisation for type-II hybrid ARQ," submitted to *Electronics Letters*, July 2004.

Elisabeth Uhlemann, Lars K. Rasmussen, Alex J. Grant and Per-Arne Wiberg, "Optimal type-II concatenated hybrid ARQ using single parity check codes," in *Proc. International Symposium on Turbo Codes & Related Topics*, Brest, France, Sept. 2003, pp. 587-590.

Elisabeth Uhlemann, Lars K. Rasmussen, Alex J. Grant and Per-Arne Wiberg, "Optimal incremental-redundancy strategy for type-II hybrid ARQ," in *Proc. IEEE International Symposium on Information Theory*, Yokohama, Japan, June 2003, p. 448.

Elisabeth Uhlemann, Tor M. Aulin, Lars K. Rasmussen and Per-Arne Wiberg, "Packet combining and doping in concatenated hybrid ARQ schemes using iterative decoding," in *Proc. IEEE Wireless Communications and Networking Conference*, New Orleans, LA, March 2003, pp. 849-854.

Elisabeth Uhlemann, Tor M. Aulin, Lars K. Rasmussen and Per-Arne Wiberg, "Concatenated hybrid ARQ – a flexible scheme for wireless real-time communication," in *Proc. IEEE Real-Time Embedded Technology and Applications Symposium*, San Jose, CA, September 2002, pp. 35-44.

Elisabeth Uhlemann, Tor M. Aulin, Lars K. Rasmussen and Per-Arne Wiberg, "Hybrid ARQ based on serially concatenated block codes using iterative decoding for real-time communication," in *Proc. Radiovetenskap och Kommunikation*, Stockholm, Sweden, June 2002, pp. 517-521.

Elisabeth Uhlemann, "Hybrid ARQ Using Serially Concatenated Block Codes for Real-Time Communication – An Iterative Decoding Approach," *Licentiate Thesis*, Chalmers University of Technology, Göteborg, Sweden, October 2001.

Elisabeth Uhlemann, Tor M. Aulin, Lars K. Rasmussen and Per-Arne Wiberg, "Deadline dependent coding – a framework for wireless real-time communication," in *Proc. International Conference on Real-Time Computing Systems and Applications*, Cheju Island, South Korea, December 2000, pp. 135-142.

Contents

Acknowledgements	ix
1. Introduction	1
1.1 Real-Time Systems.....	2
1.2 Real-Time Communications.....	5
1.3 Challenges with Wireless Real-Time Communication	7
1.4 Problem Formulation.....	8
1.5 Deadline Dependent Coding	9
1.5.1 Hybrid Automatic Repeat Request.....	10
1.5.2 Concatenated Codes with Iterative Decoding	12
1.6 Contributions	13
1.7 Outline of the Thesis	16
2. System Model	19
3. Incremental Redundancy Hybrid Automatic Repeat Request	31
3.1 Pure ARQ Schemes	31
3.2 Hybrid ARQ Schemes	32
3.2.1 Packet Combining	33
3.3 Hybrid ARQ Schemes with Concatenated Codes	35
3.4 Throughput, BER and Average Code Rate	36
3.4.1 Maximizing Average Code Rate	39
4. Multiple Concatenated Single Parity Check Codes	43
4.1 Encoding.....	44
4.2 Decoding	46
4.3 Interleaver Design	48
4.4 Rate Compatible Puncturing	52
5. Performance Analysis Using Union Bounds	59
5.1 Union Bounds on Concatenated Block Codes	59
5.1.1 Multiple Parallel Concatenated Block Codes.....	63
5.1.2 Multiple Serially Concatenated Block Codes	65
5.1.3 Punctured Multiple Concatenated SPC Codes	67
5.2 Selection of Good Puncturing Patterns	73

6. Performance Analysis Using EXIT Charts	81
6.1 Entropy and Mutual Information.....	82
6.2 Channel Capacity	83
6.2.1 Capacity for the Binary Input AWGN Channel	85
6.3 Extrinsic Information Transfer Characteristics	86
6.3.1 Extrinsic Information Transfer Functions	89
6.3.2 Extrinsic Information Transfer Charts	93
6.3.3 Puncturing	109
7. Performance Results	129
7.1 Parallel Concatenated Scheme	129
7.2 Serially Concatenated Scheme	140
8. Conclusions	149
8.1 Objectives Achieved.....	149
8.2 Contributions and Impact	150
8.2.1 Optimization of Packet Lengths	151
8.2.2 Multiple Concatenated SPC Codes	152
8.2.3 Performance Bounds	152
8.2.4 EXIT Charts Analysis	153
8.3 Perspectives	154
8.4 Future Work	154
A. Multiple Parallel Concatenated Zigzag Codes	157
A.1 Encoding.....	157
A.2 Decoding	158
A.3 Puncturing	160
A.4 Extrinsic Information Transfer Functions	161
A.5 Extrinsic Information Transfer Charts	161
A.6 Puncturing Ratios Obtained Using EXIT Charts	166
References	169

Acknowledgements

First and foremost I would like to thank my research supervisor Professor Lars K. Rasmussen, Chalmers University of Technology and University of South Australia. His scientific mind, his truly contagious enthusiasm and his indefatigable encouragement cannot be overestimated in the genesis of this thesis. His mode of guidance makes one grow and I hope some of his zeal is reflected in this work. For all of this and for good friendship I owe him my deepest gratitude.

I am also greatly indebted to:

My colleagues Dr. Fredrik Brännström and Peng Hui Tan, Chalmers University of Technology, who have constituted my academic brotherhood over many years and during travels in many countries. They have provided active help by discussing, proof-reading, reassuring as well as being excellent travel companions.

My project leader/initiator Per-Arne Wiberg, Halmstad University and Free2move, who originally lit my interest in scientific work and who has provided profitable aspects during its process.

Professor Alex Grant, University of South Australia, for welcoming me to spend a number of months as a visiting researcher in Adelaide and for valuable ideas and helpful criticism on my papers.

Professor Tor M. Aulin, Chalmers University of Technology, for introducing me to the field of telecommunication.

Professor Bertil Svensson, Halmstad University, for constant encouragement, guidance and support in matters of scientific as well as practical nature.

My project group member Urban Bilstrup, Halmstad University, for practical and mental support.

My friends and colleagues at Halmstad University, Chalmers University of Technology and University of South Australia, for providing fruitful and pleasant research environments.

Last, but not least I would like to thank my family and friends who make it all worth while. My parents Christer and Margareta for believing in me (if I ever become half as good as you believe I am – it will be purely a result of good genes). My sister Hélène, Malin Svensson and Henrik Bengtsson for being friends out of the ordinary.

This work was mainly funded by the national Swedish Real-Time Systems research initiative ARTES, supported by the Swedish Foundation for Strategic Research, but also in part by Personal Computing and Communication (PCC++) under Grant PCC-0201-09 and by the Center for Research on Embedded Systems (CERES), Halmstad University.

Chapter 1

Introduction

The recent development in wireless communication has resulted in enhanced services and products being introduced into the market at an ever-increasing rate. This wireless evolution offers improvements for industrial applications, where traditional wired solutions have prohibitive problems in terms of cost and feasibility. Wired implementations are for example not cost efficient for large, temporary production lines, and may not be feasible at all for systems including rotating or high mobility machinery such as measurement and control of moving objects. In this context, when considering the opportunities in related fields and not just as replacement for existing cables, wireless communication has the scope for considerable growth. There is even reason to talk about a wireless revolution when considering cooperative embedded systems in home equipment, automotive components, logistics services, and entertainment devices wanting to communicate information as well as entertainment.

The growing evolution of wireless communication, and all the new applications this enables, also rapidly increases our demands on the performance of communication networks. As the transmission speed increases, new wireless applications and services, like for example wireless video streaming, suddenly becomes interesting. The expectations of the general user with respect to performance of wireless applications are guided by the current quality of traditional wired systems. Many of these new wireless applications are subject to time-critical constraints, so called real-time constraints.

In a typical wireless communication system, the channel conditions vary with time, and thus the quality of frames transmitted over the channel is not constant. The inherent consequence is a relatively high average error rate, making the wireless channel significantly less reliable in comparison to copper wire local loop channels or optical channels. This has limited the extensive use of wireless access in systems with real-time constraints.

The purpose of the work in this thesis is to improve the performance of real-time communication over a wireless channel, by means of specifically tailored channel coding. The applications that are targeted in this thesis are not necessarily replacement of cables in

existing wired systems. Instead the motivation lies in the new services that wireless real-time communication enables – services that may not yet exist. Examples of this could be communication between cars to improve security in a collision avoidance system. It may be sensors along the road informing passing cars about hazards ahead, e.g., icy road conditions, a moose crossing or traffic jams ahead. Therefore, embedded systems and communication within and between cooperating embedded systems are typically the focus.

1.1 Real-Time Systems

A real-time system depends on real time in the sense that the result of its execution needs to be presented in a timely fashion. This implies that it is not only the result itself that is of importance but also when in time it is presented. Therefore a real-time task has a deadline to meet. What happens if the deadline is missed varies with application, much the same way as presenting a timely but incorrect result does. In some applications a missed deadline may have severe consequences. If, for example, a real-time system is used to control the airbag in a car, it does not matter if a correct result is presented, i.e., the airbag is correctly triggered – if it is not presented in time, i.e., if the airbag is not triggered until after the car has crashed, the consequences can be severe. In other cases, a missed deadline will only imply reduced quality. For example, in a video conference a missed deadline will only result in a temporary lowered quality and any picture or sound frames that arrive too late will simply be thrown away in order to quickly return to normal behavior. This implies that the application is not terminated by a missed deadline, but the quality is reduced. The consequences of a missed deadline, and hence a reduced quality of the service provided, are not severe but can still be damaging in some way. The video conference may possibly be tele-medicine, i.e., surgical procedures which are monitored by a remote expert, and a continuing bad quality may jeopardize the procedure. It may also make the users of the application choose another service provider for the next session.

In order to grade the importance of a deadline, real-time tasks are often classified as being critical, essential or non-essential. If a critical task misses its deadline the consequences can be catastrophic and in most cases the system activity is terminated. Therefore, when critical tasks are present in the system, resources are often kept in reserve since the analysis of the required service for critical tasks is made on worst-case values rather than average behavior. Essential tasks will not cause a catastrophe or a system halt if they miss their deadlines, but will lead to a system malfunction for a short or long period of time. Most real-time tasks fall into this category. Finally, non-essential tasks often have no deadline at all (non-real-time tasks) or, if they do have deadlines, nothing critical or essential will happen if these are missed. Maintenance tasks are typical examples of non-essential tasks.

Real-time tasks are also classified as having hard or soft real-time constraints. A task in a hard real-time system becomes useless when the deadline has passed, whereas in a soft real-time system the importance of the result degrades with time after the deadline has

passed. Hence, if a task with soft real-time constraints misses its deadline, its execution is often continued, since its result will still be of some reduced value for a short period of time. Consequently, its deadline is said to be softer.

Often when hard real-time is discussed in the literature, it is also implied that the tasks are critical. Similarly, soft real-time tasks and essential tasks are often connected. The example with the airbag above can be classified as being a critical task, since a missed deadline may have critical effects such as personal injury. The task can potentially also be classified as a hard real-time system, since the airbag needs to be inflated before the crash. It is sometimes difficult to classify a task as being a true hard real-time task, even in the case with the airbag. Assume that the deadline for triggering the airbag is before impact. If this deadline is missed the airbag could still be triggered after the car has started crashing – since crashing can be expected to take a non-zero time. Hence, the deadline is soft, even if it is clear that inflating the airbag once the car has come to a complete halt is useless. It is easier to determine that the video conference example above can be classified as an essential, soft real-time system.

Somehow this classification into critical and essential, hard and soft real-time tasks is an attempt made by the application or the user of the system to convey the importance of different tasks in a system and their deadlines. The reason for doing this is that the available resources in the system often are limited and hence we need to use them as best we can. This is when scheduling becomes important. Consider a processing unit, which is a limited resource that typically has several tasks of different importance to run. We then need to find a suitable procedure for determining the order in which these tasks should be executed. If all the tasks in the system are scheduled such that they all meet their respective deadlines, the corresponding schedule is called feasible. A scheduling algorithm is said to be optimal if it can always find a feasible schedule whenever any other scheduling algorithm is able to do so.

Sometimes additional constraints, such as precedence constraints and preemption or non-preemption, can be encountered. If task A is dependent on the result of task B , it has to start execution following completion of task B . Consequently, even if the tasks have the same release time, i.e., time when the tasks are made available for execution, the dependent task A must be delayed until task B has been completed. This is referred to as a precedence constraint. Some tasks can be interrupted during execution to give way for more critical tasks, whereas others cannot be stopped once they have started. This is termed preemption and non-preemption respectively, as illustrated in Figure 1.1.

There are a number of scheduling algorithms available in the literature, e.g., earliest deadline first (EDF), [1]. EDF looks only at the deadline of each individual task and executes the task which has its deadline closest in time. Hence, priority is given to the most urgent task. EDF is optimal for uniprocessors [1]. Recent research on scheduling is focusing on, e.g., finding optimal scheduling techniques using constraint programming, [2]. Many times, scheduling is done offline for systems that include critical tasks, but in some cases offline scheduling is not an option.

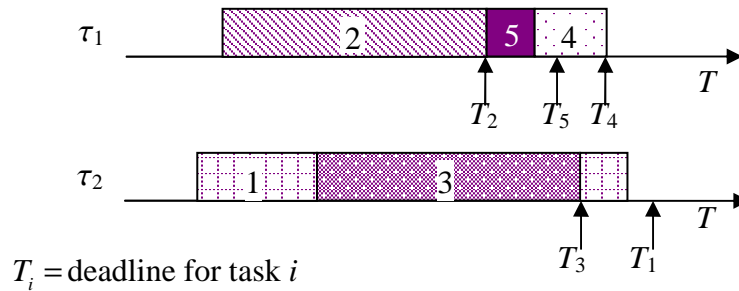


Figure 1.1. Scheduling tasks on two processors. The tasks are numbered according to their respective release times. Further, there is a precedence constraint so that task 5 should be executed before task 4. Also note that task 1 is preempted by task 3 on processor τ_2 .

Scheduling with priorities is one way of controlling the order in which tasks are to be executed. Admission control is another option. Admission control is most often used for online scheduling. The scheduler then makes an estimate of the remaining available resources and only allows a new task into the system if its inclusion does not jeopardize previously guaranteed tasks. If the system includes critical tasks the worst-case values are used in the analysis, otherwise average values are most common. When determining the worst-case completion time for a task, we do not only consider the execution time of the task, but also execution interference from other higher priority tasks and any potential preemption. If a task is accepted in an admission control system, it is typically guaranteed a certain quality of service (QoS). Usually, there are different levels of QoS available in the system. Using different priorities is one way of ensuring different levels of quality. Best-effort is a particular QoS level often provided. It does not, however, give any actual guarantees, since only best effort will be made to ensure execution before deadline. Hence, best-effort is mostly used for non-essential soft real-time or non-real-time tasks. In admission control systems a task often seeks admission to a system requesting a specific QoS level and if accepted it will be so with the requested QoS level. If the task cannot be accepted, there is often a possibility to reduce the requested level of QoS and seek admission anew. Consequently, a negotiation may take place when using admission control systems.

One reason for having several QoS levels is that most systems need to be able to support both real-time and non-real-time tasks concurrently. The relative deadline of a task is defined as the difference between its deadline and its release time. If the execution time of a real-time task is smaller than its relative deadline, the remaining time is called slack time. In order to support both real-time and non-real-time tasks in a system it is important to use the slack time to run non-real-time tasks. Obviously, if preemption is allowed the risk of completely starving non-real-time tasks will be reduced.

1.2 Real-Time Communications

Communication is an important part of most real-time systems and can take place both between systems and within systems. For example, communication is needed in and between most embedded and distributed systems so that sensors can report their readings and actuators be given directions. In multiprocessor systems the different processors need to communicate with each other, and multimedia applications such as video conferencing and voice-over-IP implies communication.

The communication medium can be quite different depending on the application. It could be, e.g., optical wireline channels, copper wire local loops, a ring or a bus, wireless, or the Internet which is in a sense a mixture of all of the above. A local area network (LAN) is a relatively small network that shares a common medium that usually employs the same medium access control (MAC) method. The role of the MAC protocol is to ensure that all nodes connected to the LAN get access to the medium. The specific MAC method used generally depends on the medium in question, e.g., a ring networks may use Token ring, the Ethernet uses carrier sense multiple access with collision detection (CSMA/CD), and the GSM network uses time division multiple access (TDMA), [3]. Some of these MAC methods are said to be deterministic whereas others are not. TDMA is an example of a MAC method with a deterministic behavior. The time slots are usually allocated offline, and a node carrying real-time traffic knows when it will get access to the medium. Hence, a schedulability analysis is possible. CSMA/CD, on the other hand, is not deterministic. The reason for this is that collisions can occur, forcing the nodes to compete for access. Whenever a collision occurs between two nodes, they both wait a random time before making a second attempt. Hence, collisions may occur once more. Consequently, there are no deterministic guarantees on when a node can get access to the medium.

Several extensions to non-deterministic MAC methods attempting to make them more suitable for real-time traffic have been proposed. The Timed-token protocol [4] is a real-time extension to the Token ring protocol where the token rotation time is monitored and kept close to a target time. RETHER [5] is a real-time extension to the Ethernet, where a token based protocol is used on top of the normal MAC protocol. Recent research concerning how to use the Ethernet for real-time communication mainly focuses on avoiding collisions completely by using switches that support full duplex, [6].

Scheduling of tasks to different processors has many similarities with accessing the shared communication medium. The communication medium is a limited shared resource and the communication itself can be seen as a task that has a release time and a deadline. Consequently, processor scheduling algorithms can often be used to do communications scheduling as well. There are however a few differences. It is difficult to completely centralize the scheduling of communication tasks. Further, a large diverse network makes the scheduling problem significantly more complex. Specifically, if there are more than one MAC technique in use in the network and when routing is necessary.

There are two types of distinguished communication scheduling methods; integrated scheduling and separated scheduling. With integrated scheduling, the extra delay caused by communication is seen as a part of the overall execution time of the task. In separated scheduling, the communication is seen as a separate task with its own release time and its own deadline, as illustrated in Figure 1.2. The latter allows for different dispatching strategies, i.e., different MAC techniques, and is also more suitable when routing is required. Routing issues for real-time communication in *ad hoc* networks have been investigated in e.g., [7],[8].

The characteristics of real-time communication differ from the characteristics of non-real-time communication. For example, the traditional measure of throughput is of less importance. Instead we are interested in a message loss rate or a probability of a message arriving before its deadline. A lost message will result in an infinite delay. Traditional critical hard real-time communication systems often require an upper bound on the maximum end-to-end message delay. The real-time literature discusses two types of traffic in this context; guaranteed traffic and statistical traffic. When the traffic stream is guaranteed, it means that every frame in the stream is guaranteed to arrive before its deadline. Statistical traffic, on the other hand, refers to that no more than a certain percentage of the frames in the stream may miss their respective deadline. Hence, when dealing with statistical guarantees we need to know the task deadlines as well as the accepted deadline miss ratio. Statistical guarantees are often given to, e.g., traffic generated by an over-sampled sensor.

Alternatively, some real-time literature classifies guarantees as being deterministic or probabilistic. If the offered service is deterministic it is said to be predictable and suitable for hard real-time systems, since normally both a “guaranteed” minimum throughput and a bounded end-to-end delay is offered. A probabilistic guarantee is said to “only guarantee” to meet a specified QoS with a certain *probability*. This probability may however be equal to one and hence result in performance that is comparable to a deterministic system.

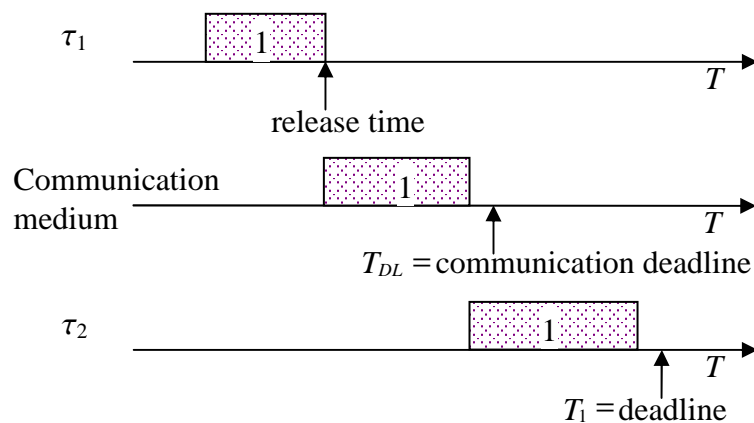


Figure 1.2. Separated scheduling of a communication task.

Probabilistic guarantees are often connected to average behavior, whereas deterministic guarantees are made on a worst-case analysis and hence having a probabilistic guarantee is said to yield a higher utilization of the system resources. When a real-time system includes communication, we inherently have probabilistic guarantees only since random noise prevents a deterministic description. It follows that throughput is a statistical measure and hence we can only guarantee a minimal *average* throughput. Similarly, the end-to-end delay cannot be deterministically bounded since that would imply a zero error rate. *Therefore, it may be meaningful to talk about deterministic guarantees in a real-time system that does not include communication, but as soon as communication is involved then the probability of an error free message arriving on time is always less than one.* For some communication channels the error rate may be relatively low, as is the case for some fiber optic networks. In these cases it may still be justified to talk about a behavior being more or less deterministic, but for e.g., a wireless channel we can only offer probabilistic guarantees.

1.3 Challenges with Wireless Real-Time Communication

In a typical wireless communication system, the channel conditions vary with time, and thus the quality of a stream of frames that have been transmitted over the channel is not constant. The inherent consequence is a relatively high intermittent error rate, making the wireless channel less reliable compared to copper wire local loops or optical wired channels. The concepts of channel coding [9] to cope with the high error rate must therefore be introduced in order to provide a more reliable channel.

Another difficulty with a wireless channel is that its bandwidth is limited since the radio spectrum is a limited natural resource. A fully utilized frequency band cannot easily be complemented by additional resources. It is not possible to add an extra fiber or an extra communication bus which is often done with wired real-time systems in order to increase the capacity or the fault tolerance. The radio spectrum is assigned according to strictly enforced rules and consequently additional bandwidth may be costly or may not exist at all.

Furthermore, wireless devices are often battery powered and therefore the transmitted signal power should be limited to prolong battery life. The battery also puts restrictions on the maximal computational complexity that can be used in each of the end nodes. Moreover, given that we have a certain bandwidth to use in our system, a limited transmit power also limits the inherent interference generated by other transmitters present in the local wireless multiple access system.

The channel capacity formulated by Shannon [10] incorporates into one composite parameter the effects of channel parameters such as thermal noise, constrained bandwidth, and limited signal power. The channel capacity is a fundamental upper limit for the achievable data rate over channels described by these parameters. The significance of the channel capacity is that as long as the communication rate is kept below the channel capacity, an arbitrarily low error rate can in principle be obtained if infinitely long signals

are used. From coding theory, we know that most finite-length codes are good, provided that they are sufficiently long. Decoding complexity, however, generally increases exponentially with the block length of the code and hence may prohibit the use of codes beyond a certain length. At the same time as a long code will improve the performance in terms of lowering the bit error rate (BER), it will also require more resources in terms of more bandwidth, more energy and above all, in this context, more time to transmit and decode. When a real-time communication system is considered, time is in some sense a limited resource and hence we are not only concerned with limiting the length of the code in order to limit the decoding complexity but also to limit the overall communication time.

1.4 Problem Formulation

Although the search for contention free MAC protocols and optimal communications scheduling algorithms are interesting and important problems, they are not treated here. *Instead this thesis treats the problem of suggesting good channel coding and decoding methods to be used in a wireless real-time communication system.* Hence, the aim is to find a deadline dependent coding protocol, where the channel code used is tailored to the real-time constraints.

We assume that we have a MAC protocol that enables instant access to a wireless LAN, perhaps by using code division multiple access (CDMA), [3]. Further, the focus is on point-to-point communication rather than routing. Separated scheduling is assumed so that each communication task has its own release time and deadline.

We exploit a probabilistic view of the real-time constraints that focuses on worst case behavior as far as possible¹ in order to provide a systematic approach for the development of efficient wireless real-time communication protocols. This means that we do not talk about hard or soft real-time requirements or critical and essential tasks. Instead we talk about a communication deadline and the probability of succeeding in delivering correct information before this deadline. Thus, we use two QoS parameters: deadline for delivery, T_{DL} , and the probability of correct delivery prior to reaching the deadline, P_d . Correct delivery implies that a certain target error rate, P_t , is met. This error rate can be in terms of average frame error rate (FER) or average BER within the frames. Note that the target error rate cannot be equal to zero due to the presence of channel noise. Nor can P_d be equal to one.

Using these QoS parameters it follows that a protocol layer can negotiate values of the parameters with an upper or a lower layer, thus enabling flexible admission control that provides a *trade-off between the delivery time and the quality of the delivered data*. The values of the QoS parameters are requested by the application using the communication system. The value of P_d controls how reliable the transfer must be in a real-time

¹ Note that the worst case behavior for a real-time communication system will always be that the deadline will be missed, since the BER and the throughput is based on the average behavior of the system.

perspective, i.e., a measure of how critical the task is, and consequently, it does not say anything about delivery of correct information after the designated deadline. The negotiation about the value of T_{DL} reflects how soft the real-time constraints on the deadline are.

One of the objectives of the wireless real-time communication protocol is to maximize the probability that the communication system will be able to accept the transmission request with the required values of the QoS parameters. Besides maximizing the probability of delivering the required information before a given deadline, the protocol should also attempt to minimize the required bandwidth, the transmitted energy and the average time required to successfully deliver the information.

It is worth noting that these QoS parameters are useful for real-time as well as non-real-time applications. An application sending emails may for example require P_d to be as high as possible but can relax the constraints on T_{DL} if negotiations require a lower QoS level in order to be able to accept the request. If a packet of image data intended for video streaming is to be sent, the deadline is relatively tight but P_d can be moderate, since an incorrect delivery (or no delivery) will appear as image noise. In contrast, a packet arriving too late will disturb the viewing more. For a control application, it is important that correct information reaches its destination before a firm deadline with a fairly low error probability. This implies that more bandwidth will be required for these kinds of applications.

1.5 Deadline Dependent Coding

Given the above framework, the main idea behind the concept of deadline dependent coding (DDC) is to make all components in the link control protocol, including the channel code, *deadline dependent*. Generally, by using longer code words, i.e., by adding more redundancy, we will achieve a higher P_d – but we will, in turn, be requiring more time, as illustrated in Figure 1.3. Traditionally, a fixed time has been used when scheduling the transmission over a communication medium. In contrast, we make this time variable depending of the required quality (error rate).

The protocol is further intended to minimize the required bandwidth, the multiple access interference and the transmitted energy. We therefore not only consider concatenated codes [9] to cope with high error rates, but also powerful retransmission schemes [11] to provide time diversity in order to obtain a more flexible and reliable scheme. Concatenated codes using iterative decoding is a way of providing long codes with manageable decoding complexity. The retransmission protocol plays the role of maximizing the probability of delivery with the required error rate before the deadline, using a minimum of resources. These two main components in the DDC protocol, the retransmissions scheme and the concatenated code, are described further below.

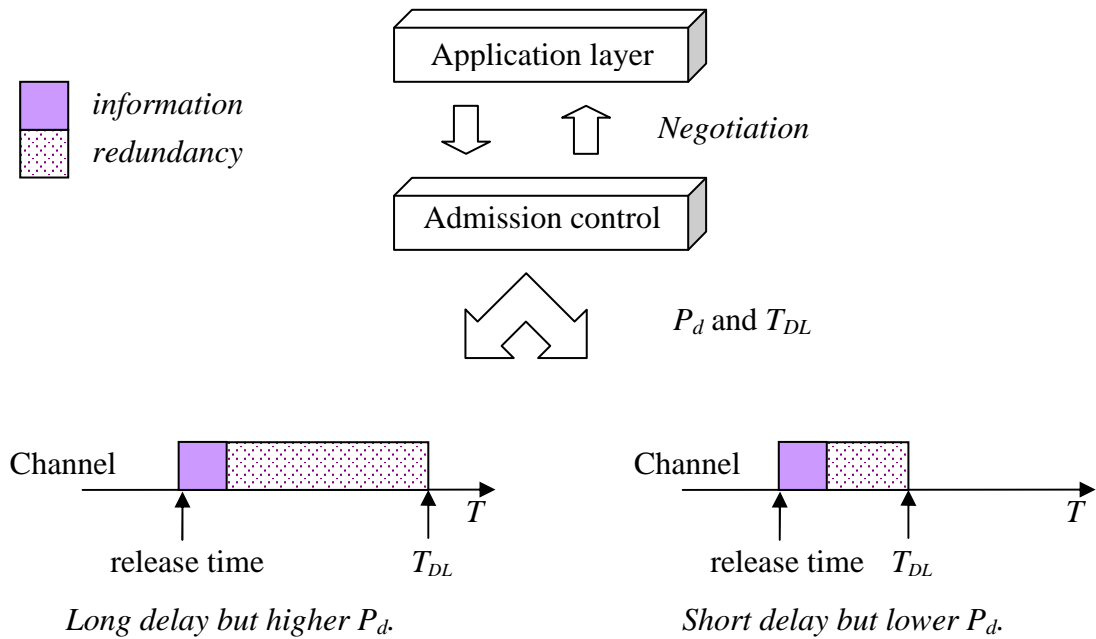


Figure 1.3. Quality of service negotiations leading to different code lengths.

1.5.1 Hybrid Automatic Repeat Request

In a packet-based system, an automatic repeat request (ARQ) scheme [12] can be used. Whenever a packet arrives, the receiver may choose to reject it, and instead send a retransmission request through a feedback channel. To determine whether or not a retransmission should be requested the receiver checks the quality of the received packet, usually by means of an error detection code. A hybrid ARQ (HARQ) scheme, first suggested in [13], uses an error control code in conjunction with the retransmission scheme. Consequently, the receiver first tries to decode the received codeword and only requests a retransmission if the quality of the decoded information is not acceptable. There are different methods of determining whether a decoder output is sufficiently reliable and hence different criteria for requesting a retransmission. The choice of method significantly affects the character of the retransmission scheme. The most common method is to apply an error detection code like a Cyclic Redundancy Check (CRC) code [11]. When concatenated codes are used in conjunction with the HARQ scheme, reliability information from the iterative decoding process may be used as a retransmission criterion. This is discussed further in the next sub-section.

In a pure HARQ scheme, rejected packets are discarded. Previously received packets may, however, be used for so-called packet combining, in order to improve performance. There are two major types of packet combining, diversity combining [14] and code combining [15]. The choice of packet combining technique is related to the choice of error control code, but the goal should always be to use all the received observables in the

decoding process. In [16, 17] different diversity combining techniques for HARQ protocols using concatenated coding were investigated.

Incremental redundancy (IR), first suggested in [18], implies that the HARQ system responds to a retransmission request by sending more redundancy. This redundancy is then used to form an increasingly longer code, by means of code combining. IR-HARQ schemes are appealing since no repetition of previously transmitted bits is made.

Rate compatible code families are commonly used in IR-HARQ schemes. Typically, a rate compatible code family is constructed by choosing a low rate *mother code*, which is punctured to provide several higher-rate codes. For example, a codeword of length n from the mother code may be divided into two parts, constituting two complementing punctured codewords of lengths n_1 and n_2 with $n = n_1 + n_2$. These codewords may then be sent one after the other over the channel. This splitting or puncturing of the mother code can be done in different ways. Generally, the IR packet lengths are assumed to be fixed, and the puncturing strategy, i.e. the order in which the coded bits are to be transmitted, is optimized [19]. Hence, the focus on IR schemes has been concentrated on finding optimal puncturing patterns, assuming a specific IR transmission strategy. The converse problem has so far been overlooked in the literature.

One of the main components in the DDC protocol is the retransmission scheme. However, since we specifically consider a time-limited channel, the maximum number of retransmissions must be limited, and hence the HARQ system becomes truncated [20]. The maximum number of retransmissions allowed is chosen according to the deadline for delivery, T_{DL} and will consequently provide an upper bound on the communication time.

The benefit of a retransmission scheme is that it is continuously adapting to instantaneous channel conditions. A series of retransmissions is initiated when the channel is bad, thus contributing to the robustness of the protocol, while only a negligible number of retransmissions are required when the channel is well behaved. We may therefore use only the required amount of redundancy suitable for the current channel condition and thereby save energy and bandwidth resources as well as reducing the amount of multiple access interference. Rather than designing the system based on the worst possible channel conditions, increasingly more channel resources are allocated as the deadline approaches, in order to meet the probabilistic requirements for delivery before the deadline.

It has been argued that a retransmission scheme is not to be used in a real-time system, e.g., [21] – however as long as there is an upper limit on the maximum number of retransmissions allowed and this limit is connected to the deadline, the delay is finite and controllable. Consequently, having a truncated ARQ protocol we know how many retransmissions will occur during worst-case conditions. Knowing the transmission rate and the approximate distance to the receiver we can get a worst-case transmission time for the communication task. We will also have an average transmission time and hence slack time. This slack time may be used to run non-real-time tasks, or alternatively just seen as a way to limit the interference to other nodes that are transmitting real-time traffic concurrently.

Recall that scheduling of the communication channel is usually non-preemptive, in the sense that once a task has started to be transmitted it cannot be preempted if another more

important task comes along. However, using the DDC scheme, we can preempt the ongoing task in the sense that we can stop further retransmissions and instead let the new task start.

1.5.2 Concatenated Codes with Iterative Decoding

The channel capacity unfortunately only states what data rate is theoretically possible to achieve, but it does not say what codes to use in order to achieve an arbitrary low BER for this data rate. Therefore, there has traditionally been a gap between the theoretical limit and the achievable data rates obtained using codes of a manageable decoding complexity. However, in 1993 a novel approach to error control coding revolutionized the area of coding theory. The so-called turbo codes [22] almost completely closed the gap between the theoretical limit and the data rate obtained using practical implementations. Turbo codes are based on concatenated codes separated by an interleaver. The concatenated code can be decoded using a low-complexity iterative decoding algorithm. Although this iterative decoding algorithm is sub-optimal, it has been shown to essentially avoid any performance loss, as compared to optimal decoding, [22].

Concatenated codes using iterative decoding is a way of providing long codes with manageable decoding complexity. Note that the term “long code” used here does not refer to the length of the information frame, but rather the length of the codeword, or the length of the code memory. A turbo code is basically a concatenated system of two simple component codes connected through an interleaver in order to create a very long code and thus also a very strong code [22, 23]. Optimal decoding of such a system is NP-hard and intractable due to the length of the code as determined by the size of the interleaver. However, the attractive characteristic of a concatenated system is that iterative *a posteriori* probability (APP) decoding based on exchanging soft reliability information provides a low complexity sub-optimal decoding algorithm. This implies that each component decoder is used several times in the decoding process, usually once for each iteration. Hence, the soft reliability information exchanged between the component decoders is iteratively refined and the interleaver is used as an integrated part of the code. Given certain conditions, the iterative decoding algorithm performs close to the fundamental Shannon capacity [22].

In general, concatenated coding provides longer codes yielding significant performance improvements at reasonable complexity investments. The overall decoding complexity of the iterative decoding algorithm for a concatenated code is lower than that required for a single code of the corresponding performance. The lower complexity is achieved by decoding each component code separately. Hence, the low-complexity decoders for the simple component codes are used and iteratively reused several times, instead of using one highly complex optimal decoder for the mother code. The complexity of the iterative decoder is increasing linearly with the interleaver size and number of iterations. Initially, there is much to be gained from iterating, but eventually the performance reaches a point of diminishing returns. The number of iterations needed for convergence varies between packets and is generally not known. A common approach for stopping the iterative decoding process is to allow for a fixed number of iterations. This may lead to unnecessary

iterations or to performance degradation if the process is terminated prematurely. Applying a performance-based stopping criterion as in, e.g., [24], these problems can be addressed. The stopping criterion is intended to stop the iterative process as soon as the required performance is reached – however, this may never occur in some cases due to excessive noise. As we are dealing with real-time communication we must have an upper limit on the time to decode a frame. The decoding complexity, and hence the time required to deliver a frame with sufficient quality, is directly related to the number of iterations. Hence, we still need to have an upper limit on the maximum number of iterations allowed. Examining the convergence behavior in a concatenated system, it is noticed that for a majority of packets, if convergence occurs, it is generally reached after a fixed number of iterations, [16]. Consequently, the maximum allowed number of iterations may be set accordingly. A non-negligible number of packets may, however, converge faster and iterations will cease as a result of the performance-based stopping criterion.

Using concatenated codes in an HARQ scheme also elevates the corresponding performance to levels close to fundamental limits. The same performance-based stopping criterion can also be used as a retransmission criterion so that if the stopping criterion is not fulfilled after the maximum allowed number of iterations a retransmission is requested, [16, 24]. The stopping criterion together with the upper limit on the number of iterations and the number of retransmissions in the truncated IR-HARQ scheme yield an upper bound on the decoding complexity thus also the time to decode the information.

Concatenated codes using iterative decoding also provides the opportunity to perform so-called erasure decoding. This means that if a specific bit is unknown, i.e., no APP information is received; the iterative decoder simply assigns an *a priori* probability of 0.5 and proceeds with the decoding process. This in turn implies that we can construct *rate compatible codes* simply by transmitting a subset of all available parity bits. Hence, we can puncture some bits from the mother codeword and instead include them in a potential retransmission. Rate compatible punctured codes used in a HARQ scheme results in efficient code combining techniques, [25].

A time and safety critical application benefits from the long powerful concatenated codes yielding reliable communication, while the processing time of the iterative decoder is kept low. The iterative decoding algorithm together with the retransmission scheme also give the opportunity to always deliver something to the receiver before the deadline, i.e., we can offer a fast tentative response and progressively provide iterative refinements (lower BER). This last-minute delivery can also be complemented by a measure of reliability or quality of the delivered information based on the current APP information.

1.6 Contributions

The objective of the work conducted in this Ph.D. study has been to develop the foundation for an efficient and reliable real-time communication protocol for critical deadline dependent communication over less reliable wireless channels. From results in the

literature, the principles of DDC tend to provide the most promising design approach for achieving this objective. The main idea behind the concept of DDC is to make the communication protocol deadline dependent. The deadline and the probability of correct delivery before the deadline are QoS parameters that are mapped onto a retransmission protocol.

Concatenated coding within ARQ protocols provides a new array of possibilities for adaptability in terms of decoding complexity and communication time versus reliability. Hence, critical reliability and timing constraints can be readily evaluated as a function of available system resources and complexity. This in turn enables quantifiable QoS and thus negotiable QoS. Service requests can therefore be accepted, rejected or re-negotiated depending on available resources.

The work reported in [16] by the author considered powerful diversity combining techniques as well as retransmission criteria especially adapted to concatenated coding. In contrast, the focus in this thesis is on incremental redundancy schemes rather than diversity combining schemes. IR-HARQ schemes are appealing since no repetition of previously transmitted bits is made, and thus redundancy can be exploited more efficiently. The work here is therefore directed towards the development of IR-HARQ schemes and corresponding low-complexity rate compatible code families based on punctured concatenated codes.

We define an IR strategy to be the maximum number of allowed transmissions and the number of code bits to be included in each transmitted packet. Based on this definition, this thesis work suggests an approach to find the optimal IR strategy that maximizes the average code rate, i.e., the optimal partitioning of the mother code over at most M transmissions, assuming a given puncturing strategy.

Rate compatible punctured concatenated codes used in a HARQ scheme do not only allow for efficient code combining, but also reduces the decoding complexity since a retransmission can simply be included in the ongoing iterative decoding process. In other words, we chose a long mother code and simply let each transmission constitute a new part of this mother code. The decoder can start iterating when the first transmission is delivered and when additional parts of the mother code arrive, they may simply be included in the ongoing iterative process. This way the iterative decoding procedure does not have to be restarted.

The constituent codes can be very simple since the total code is constructed by concatenating several component codes. Consequently, single parity check (SPC) codes [26] are chosen in this work. Simple component codes like these are especially relevant for low-complexity implementations and as such they are attractive for real-time applications. The SPC codes are concatenated both in parallel and in serial, typically using more than two components codes. The FER performance of concatenated SPC codes are analyzed both using Monte-Carlo simulations and using analytical upper bounds based on uniform interleavers [27, 28]. The bounds are modified to apply to punctured concatenated codes with more than two component codes. Further, the modified bounds are used to search for good rate compatible puncturing patterns.

The SPC component codes are also analyzed using semi-analytical extrinsic information transfer (EXIT) charts [29]. This tool provides insight into convergence behavior and, in some cases, also gives good estimates of the BER. In addition, the EXIT charts can be used to find good puncturing ratios [30] and even rate compatible puncturing ratios. For SPC codes this set of puncturing ratios obtained from EXIT charts is the same as a puncturing pattern when using random interleavers. Hence, the results obtained using bounds can be confirmed by the results achieved using EXIT charts. It is shown that a noteworthy improvement can be obtained by choosing a good puncturing pattern.

Finally, performance results using the concatenated SPC codes in IR-HARQ schemes are given. Real-time constraints are put upon the maximum number of transmissions and the required reliability in terms of FER for different QoS levels. Using the optimal packet lengths are shown to increase the average code rate, implying better use of available resources. The resulting protocol is termed incremental redundancy concatenated hybrid automatic repeat request deadline depending coding (IR-CHARQ-DDC).

As a final point concatenated zigzag (ZZ) codes [31] are considered as an alternative to enhance the performance, at the expense of a slightly higher decoding complexity. The ZZ codes are also analyzed using EXIT charts.

The aim of this thesis is to propose good channel coding and decoding methods to be used in a wireless real-time communication system. Consequently, a central contribution is the IR-CHARQ-DDC protocol and the attempt to bring together the areas of real-time communication and coding theory. However, this attempt has also resulted in some contributions to the area of communication theory. These are listed below in the order they appear in the thesis.

- Optimization of packet lengths to maximize the average code rate in an IR-HARQ scheme.
- Performance evaluation using Monte-Carlo simulations of multiple parallel and serially concatenated SPC codes and the effect of the interleaver.
- Performance bounds for punctured multiple parallel and serially concatenated SPC codes.
- Search for good rate compatible puncturing patterns for multiple concatenated SPC codes, using union bounds.
- Performance evaluation using EXIT charts of multiple parallel and serially concatenated SPC codes.
- Search for good rate compatible puncturing ratios for multiple parallel and serially concatenated SPC codes, using EXIT charts.
- Performance evaluation of parallel concatenated ZZ codes using EXIT charts.

The set of contributions are incorporated into a design process for QoS-based IR-CHARQ-DDC protocols. The QoS parameters are here in terms of a deadline and a target FER. Based on a target FER, a suitable rate compatible code family is selected where the mother code is able to provide the required reliability. As the delay in connection with a

retransmission is the most significant contribution to the overall delay in an ARQ scheme, the deadline determines the maximum number of retransmissions allowed. Given the selected code family and the maximum number of transmissions allowed, optimal packet lengths for each transmission and corresponding good puncturing patterns are then found to complete the design.

1.7 Outline of the Thesis

The thesis consists of eight chapters, which are briefly summarized below.

In Chapter 2 the system model adopted throughout the thesis is presented and discussed. A concatenated encoder and a mapper are used to transmit packets over a simple additive white Gaussian noise (AWGN) channel using binary phase shift keying (BPSK). An iterative APP decoder allowing erasure decoding is used to decode and retransmissions are requested through an error free feedback channel.

In Chapter 3 the concepts of ARQ and incremental redundancy are explained. Further, the average code rate and the FER of a truncated IR-HARQ system are derived. It is also shown how the average code rate can be maximized by choosing optimal packet lengths for each retransmission, given a particular puncturing pattern, i.e., a particular order in which to transmit the available redundancy.

Chapter 4 discusses the component codes employed by the concatenated codes used in this thesis, namely SPC codes. These codes are simple to decode and therefore ideal for use as component codes in a concatenated code. This way the concatenated code itself can be made more complex in the sense that we can concatenate more than two component codes. This is termed multiple concatenated codes. The performance of these codes is discussed with examples for both punctured and full-rate codes concatenated in parallel and in serial. The influence of the interleaver on the performance is exemplified.

The performance examples made using Monte-Carlo simulations in Chapter 4 are compared to analytical upper bounds based on a uniform interleaver in Chapter 5. The bounds are made for both punctured and full-rate multiple serially and parallel concatenated codes. For some cases, the union bounds can further be used to obtain good puncturing patterns, which are chosen to give a low FER for low signal-to-noise ratios (SNRs). Further, the FER for every code rate using the chosen puncturing pattern can be obtained for use in the optimization procedure of Chapter 3.

An EXIT chart is a powerful semi-analytical tool used to track the performance of an iterative decoder as a function of the number of iterations. EXIT charts can for example be used to analyze the convergence behavior of multiple concatenated codes and to obtain an estimate of the BER after convergence is reached. An EXIT chart approach for performance evaluation of SPC codes is detailed in Chapter 6. Most importantly in this context, the EXIT charts are also used to find good puncturing ratios for specific code rates that yield good performance in terms of convergence behavior. The concatenated SPC codes are analyzed using EXIT charts and good puncturing ratios are obtained. Finally, an attempt to

find rate compatible puncturing ratios is made. For concatenated SPC codes using random interleavers, puncturing ratio and puncturing pattern implies the same thing. Thus, the rate compatible puncturing pattern obtained using bounds can be confirmed by the rate compatible puncturing ratios obtained using EXIT charts.

In Chapter 7 numerical results on the average code rate in an IR-HARQ scheme using optimal packet lengths are given. The probability of a retransmission occurring is still needed. However, if we assume a genie-aided retransmission criterion resulting in perfect error detection (PED) the average code rate depends on the FER. Using PED results in a lower bound on what is possible to achieve with a good retransmission scheme. Hence, the FER obtained for the rate compatible code families from previous chapters can now be used in the optimization process together with the good puncturing patterns. Performance results are presented for both parallel and serial SPC codes for different QoS requirements.

Chapter 8 contains conclusions and suggestions for future research. Finally, Appendix A suggests using ZZ codes as an attractive alternative to SPC codes. The ZZ codes are analyzed using EXIT charts and yields notable performance improvements at reasonable complexity investments.

Chapter 2

System Model

The communication system model used throughout this thesis can be described by the block diagram in Figure 2.1. The *data source* generates a sequence of equally likely information bits. These bits are grouped into frames of length k bits, denoted $\mathbf{x} \in \{0,1\}^k$.

Error detection encoding is used when an error detection code is to be included. However, in this work a theoretical approach is taken so that retransmission requests are generated assuming a genie aided retransmission criterion resulting in perfect error detection. This will yield a lower bound on what is possible to achieve with a good retransmission criterion. This actually results in a tight lower bound in most cases, since the probability of undetected errors can be made much smaller than the probability of detected errors, [32]. Consequently, error detection encoding is not used in this work and hence the vector \mathbf{x} is fed directly into the concatenated encoder.

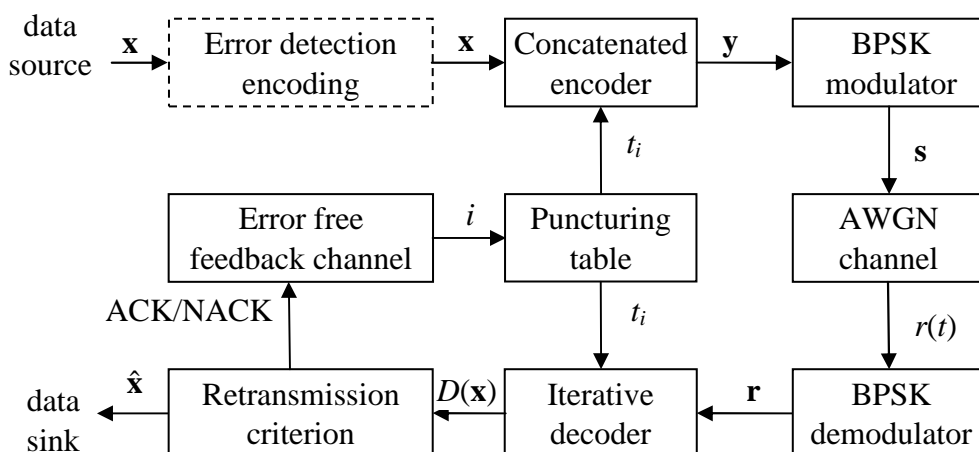


Figure 2.1. Block diagram of the communication system model used in this thesis.

The *concatenated encoder* adds a controlled amount of redundant bits to each of the frames, producing a longer codeword, n bits long, denoted $\mathbf{y} \in \{0,1\}^n$. This resulting code is called the *mother code* and it has a code rate, $r_c = k/n$. The encoder used in this work consists of multiple concatenated component codes, separated by random interleavers. The component codes are either concatenated in parallel or in serial and they are always systematic. Each encoder output as well as the systematic information output is sent through individual puncturers. Hence, only the unpunctured vector \mathbf{y} has length n . For simplicity the vectors in Figure 2.1 are denoted \mathbf{y} , \mathbf{s} and \mathbf{r} respectively, regardless of the specific puncturing ratio used. Further details of the concatenated encoder are discussed later on.

In the *BPSK modulator* each bit in the code block, \mathbf{y} , is associated with a corresponding signal waveform, $s_m(t)$, where $m = 0, 1, \dots, n-1$, for transmission over the *AWGN channel*. AWGN is a simple and frequently used mathematical model for the communication channel, which models thermal noise present in all electronic equipment. The channel corrupts the transmitted signal with additive white noise so that $r_m(t) = s_m(t) + w_m(t)$. The optimal *demodulator* for this channel samples a filter matched to the signal waveforms [33]. Consequently, the received sequence of observables, \mathbf{r} , is the sampled matched filter output, representing sufficient statistics for detection of the transmitted symbols. When using a sampled matched filter, the waveform channel can be reduced to a vector channel. This implies that each modulated code block instead can be represented by a discrete vector of symbols, \mathbf{s} , to which a noise vector of Gaussian random variables, \mathbf{w} , is added so that $\mathbf{r} = \mathbf{s} + \mathbf{w}$. The variance of each element in \mathbf{w} is $\sigma_w^2 = N_0/2$, where N_0 represents the single-sided noise spectral density. The AWGN vector channel is depicted in Figure 2.2. When a vector channel is used, the modulator and demodulator are sometimes referred to as mapper and demapper instead. In this thesis the following mapping for $\mathbf{y} \rightarrow \mathbf{s}$ is used

$$0 \rightarrow +\mu \quad (2.1)$$

$$1 \rightarrow -\mu, \quad (2.2)$$

where $\mu = \sqrt{E_s}$ and E_s denotes the symbol energy. Sometimes it is convenient to use ‘+1’ and ‘-1’ rather than zeros and ones to represent the elements in \mathbf{x} and \mathbf{y} . Consequently the

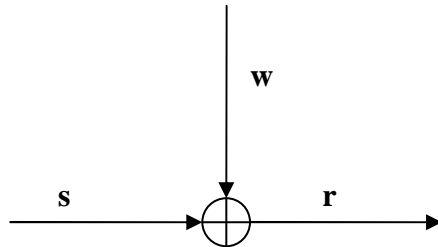


Figure 2.2. The AWGN vector channel.

following notation is used: $\bar{\mathbf{y}} \in \{+1, -1\}$, where $\bar{\mathbf{y}} = -2\mathbf{y} + 1$ and thus $\mathbf{r} = \mathbf{s} + \mathbf{w} = \mu\bar{\mathbf{y}} + \mathbf{w}$. Furthermore, an individual element in \mathbf{y} will be denoted $\mathbf{y}(m)$, while an arbitrary element with an unspecified index in \mathbf{y} will be denoted y . These notations are used for all the sequences in the system model.

A sub-optimal *iterative decoder* is used for decoding the concatenated code. It is constructed using APP decoders, one for each component code used in the encoder. Similar to the encoder, the decoder components are concatenated in parallel or in serial and are separated by random interleavers and deinterleavers. The individual puncturers are also matched in the decoder. An erasure is simply inserted whenever a bit has been punctured. This is explained in more detail below. Thereafter decoding continues normally, i.e., the constituent decoders are activated interchangeably, [22]. Since all of the component codes are systematic, any of the constituent decoders can be activated first and any arbitrary activation order can be chosen subsequently. Note that this is true regardless of whether the component codes are concatenated in parallel or in serial. Finally, the decision statistic for the transmitted sequence, $D(\mathbf{x})$, which is of length k elements, is delivered for a reliability check using a retransmission criterion. The decision is made according to

$$D(\bar{\mathbf{x}}(m)) \underset{\hat{\mathbf{x}}(m) = +1}{\overset{\hat{\mathbf{x}}(m) = -1}{\lesssim}} 0. \quad (2.3)$$

Based on the *retransmission criterion*, it is determined whether or not a retransmission is required, which in this work is done by means of perfect error detection. The request is sent through the *error free feedback channel* either by sending a negative acknowledgment signal or simply by *not* sending an acknowledgement signal. The feedback channel is depicted as a separate channel in Figure 2.1 since it is assumed to be noise free throughout this work. In addition, the actual information sequence in an acknowledgement message is usually much shorter than k bits.

Since the use of an IR-HARQ scheme implies that no repetition of previously transmitted bits is made, the following notation is chosen. If the transmitter receives a retransmission request for a particular *frame*, a new *packet* will be transmitted. An information frame or a data frame is a set of k information bits. Each data frame is passed through a rate k/n encoder, generating a codeword of length n code bits with $n - k$ redundant bits or parity bits. This codeword is then partitioned, by means of rate compatible puncturing, into M IR packets. The packets are transmitted according to an IR strategy defined as follows. Let t_i denote transmission i and c_i denote the number of parity bits sent in t_i . Note that t_1 will include k information bits as well as c_1 parity bits according to $t_1 : k + c_1$, whereas for $i > 1$, t_i includes only parity bits according to:

$$t_1 : k + c_1, \quad t_2 : c_2, \quad t_3 : c_3, \quad \dots, \quad t_M : c_M \quad (2.4)$$

such that

$$\sum_{i=1}^M c_i \leq n - k, \quad \text{where } c_i \in \{0, 1, 2, \dots, n - k\}. \quad (2.5)$$

Following transmission i , the decoder attempts to decode the corresponding codeword of length

$$n_i = k + \sum_{j=1}^i c_j. \quad (2.6)$$

If decoding is not successful, a retransmission is requested through the feedback channel and transmission t_{i+1} is made. Consequently, if a request for a retransmission is made on a particular frame – an additional IR packet is sent. A retransmission request makes the state machines of the encoder and the decoder change from t_i to t_{i+1} . This is what is done in the box marked *puncturing table* in Figure 2.1.

If systematic bits are allowed to be punctured t_1 will still include at least k bits², but now according to $t_1 : k_1 + c_1$, where $k_i \in \{0, 1, 2, \dots, k\}$. For $i \geq 1$, t_i will include $t_i : k_i + c_i$, where

$$\sum_{i=1}^M k_i \leq k. \quad (2.7)$$

Following transmission i , the received packet length is

$$n_i = \sum_{j=1}^i k_j + c_j. \quad (2.8)$$

In Figure 2.3, a parallel concatenated encoder with three component codes, C_1 , C_2 and C_3 , is depicted. This work considers parallel concatenated codes with between two and five component codes, but the notation is general and straightforward to modify accordingly. An interleaver, marked Π_i in Figure 2.3, indicates that the output vector is a scrambled version of the input vector. Random interleavers are used in this thesis work, which implies that the vector is scrambled in a random manner. Note that $\mathbf{x} = \mathbf{x}_0 = \mathbf{x}_1$ and consequently $\Pi_1(\mathbf{x}) = \mathbf{x}_2$ and $\Pi_2(\mathbf{x}) = \mathbf{x}_3$. A deinterleaver will descramble the vector to its original order as $\mathbf{x} = \mathbf{x}_0 = \mathbf{x}_1 = \Pi_1^{-1}(\mathbf{x}_2) = \Pi_2^{-1}(\mathbf{x}_3)$. Each component encoder thus has as its input the vector \mathbf{x} or an interleaved version of it. The output of each encoder is a parity vector of length $n_{C_l} - k_{C_l}$ denoted $\bar{\mathbf{y}}_l \in \{+1, -1\}^{n_{C_l} - k_{C_l}}$ for $l=1, 2, 3$, where the subscript C_l indicates the corresponding component code. The length of the output vector \mathbf{y}_l depends on the rate of the component code. If C_1 , C_2 and C_3 are all SPC codes, a single parity bit is generated for each input block of size k_{C_l} and hence each component code can be regarded as a rate-

² When systematic bits are punctured, the invertibility of the code has to be guaranteed, i.e., a one-to-one mapping between corresponding systematic and coded bits must exist. Therefore at least k bits needs to be transmitted before successful decoding is possible.

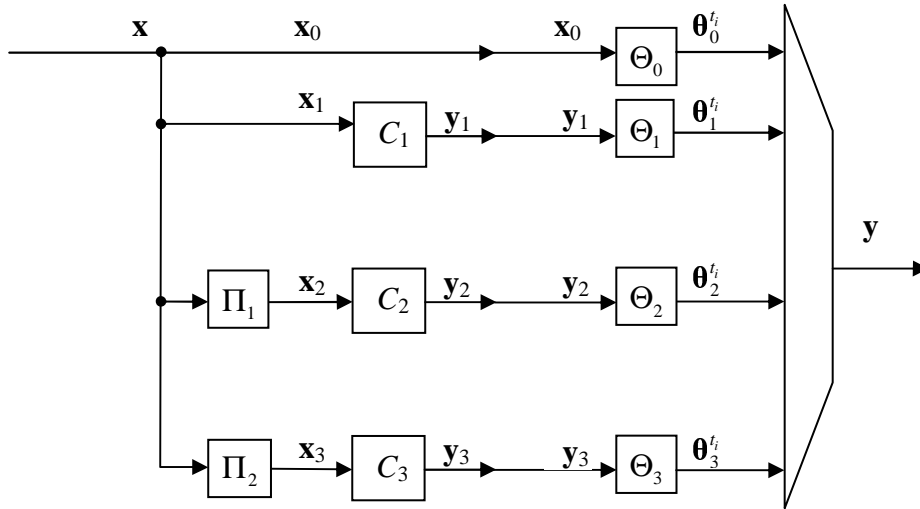


Figure 2.3. Parallel concatenated encoder with three component codes.

k_{C_i} code. For the concatenated code based on these three components, an input frame \mathbf{x} of length $k = k_{C_i}$ will result in an output packet \mathbf{y} of length $n = k + 3$, since $\mathbf{y} = [\mathbf{x}_0, \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3]$.

Alternatively, the same concatenated code could be constructed as in Figure 2.4. Here, C_1 is a systematic encoder with output $\bar{\mathbf{y}}_1 \in \{+1, -1\}^{n_{C_1}}$, whereas C_2 and C_3 are the same as in Figure 2.3. Consequently, C_1 is a rate- $k_{C_1}/(k_{C_1} + 1)$ code and C_2 and C_3 are still rate- $k_{C_i}/1$ codes. This yields the same output packet \mathbf{y} of length $n = k + 3$ for the concatenated code, since now instead $\mathbf{y} = [\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3]$. The encoder from Figure 2.3 will be used henceforth.

In Figure 2.3 it can be noted that before modulation, each component code output is sent through individual puncturers. This results in different IR packets for transmission, i.e., vectors of different lengths according to

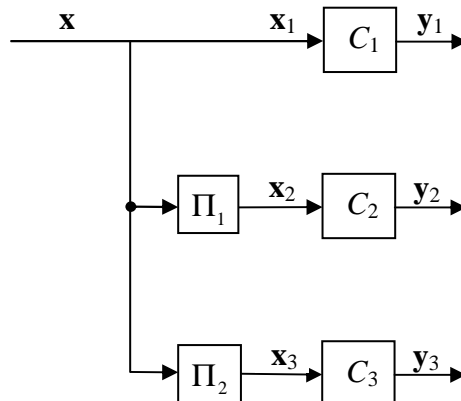


Figure 2.4. Parallel concatenated encoder with three component codes – equivalent to the version in Figure 2.3.

$$\boldsymbol{\theta}^{t_i} = [\boldsymbol{\theta}_0^{t_i}, \boldsymbol{\theta}_1^{t_i}, \boldsymbol{\theta}_2^{t_i}, \boldsymbol{\theta}_3^{t_i}] \quad (2.9)$$

for transmission t_i .

In Figure 2.5 the parallel concatenated decoder for three components, matched to the encoder used in Figure 2.3, is depicted. The *depuncturers*, denoted Θ_i^{-1} in Figure 2.5, will place the received bits, $C(\boldsymbol{\theta}_i^{t_i})$ at the correct places in the component code vectors. The number of non-zero elements in the vectors $C(\mathbf{x}_0)$, $C(\mathbf{y}_1)$, $C(\mathbf{y}_2)$ and $C(\mathbf{y}_3)$ will thus increase for each transmission. The notation $C(\cdot)$ is used to denote that these vectors are the received observables obtained from the channel. The reason for using $C(\cdot)$ will become apparent when the serially concatenated decoder is described.

Rather than working with probabilities, the decoders in this thesis work with log-likelihood ratios (LLRs), which are defined as follows. The LLR of the *a priori* probabilities of y is [34]

$$L(y) \triangleq \ln \frac{P(\bar{y} = +1)}{P(\bar{y} = -1)}, \quad (2.10)$$

where $P(\bar{y} = +1)$ denotes the probability that the discrete random variable y takes on the value $+1$. Note that we now have one value, $L(y)$ instead of two probabilities. Similarly, the LLR of y conditioned on the matched filter output r is [34]

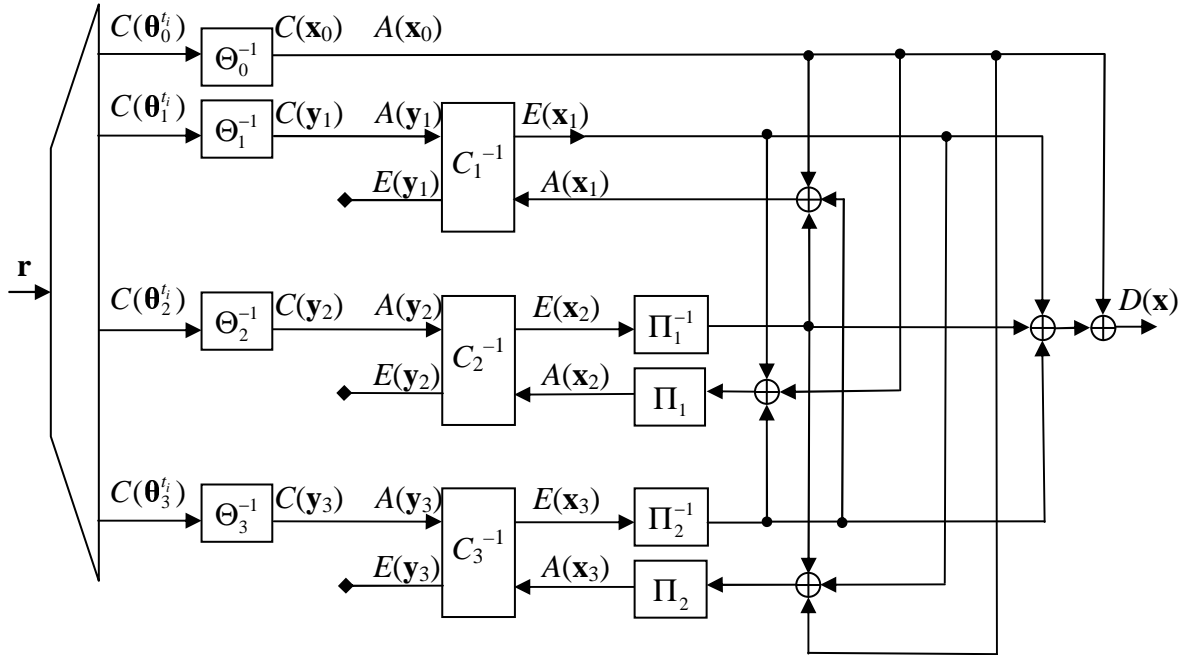


Figure 2.5. Decoder for a parallel concatenated code with three components – matching the encoder in Figure 2.3.

$$L(y|r) \triangleq \ln \left(\frac{P(\bar{y} = +1|r)}{P(\bar{y} = -1|r)} \right) = \ln \left(\frac{p_r(\beta|\bar{y} = +1)}{p_r(\beta|\bar{y} = -1)} \right) + \ln \left(\frac{P(\bar{y} = +1)}{P(\bar{y} = -1)} \right), \quad (2.11)$$

and thus

$$L(y|r) = L(r|y) + L(y). \quad (2.12)$$

For the AWGN channel with $r = s + w = \mu\bar{y} + w$, the probability density function (PDF) of the continuous random variable r conditioned on y is [33]

$$p_r(\beta|y) = p_w(\beta - \mu\bar{y}) = \frac{1}{\sqrt{2\pi\sigma_w^2}} e^{-(\beta - \mu\bar{y})^2 / 2\sigma_w^2}, \quad (2.13)$$

where the Gaussian noise has zero mean and variance $\sigma_w^2 = N_0/2$. Using (2.13) in (2.11) yields

$$L(y|r) = \ln \left(\frac{e^{-(r-\mu)^2 / 2\sigma_w^2}}{e^{-(r+\mu)^2 / 2\sigma_w^2}} \right) + L(y) = L_c r + L(y), \quad (2.14)$$

where $L_c = 2\mu/\sigma_w^2$. Before decoding of a new frame starts, $L(y) = 0$ since the binary values of y are equally likely. Consequently, the role of the depuncturers, Θ_l^{-1} , is to take each received bit from $C(\theta_l^t)$, multiply with L_c and place it in the correct place in the component code vectors.

The vectors $C(\mathbf{x}_0)$, $C(\mathbf{y}_1)$, $C(\mathbf{y}_2)$ and $C(\mathbf{y}_3)$ all contain LLR values of the observables received from the channel. If there are any positions in the vectors that are empty, because the corresponding bit has not been transmitted yet, a zero is simply inserted, implying that no prior information about this bit position has been obtained from the channels so far. If a vector for a particular component code contains all zeros, i.e., no bits pertaining to this particular component code have been received yet – the corresponding decoder need not be activated and hence complexity can be reduced.

The component decoders, marked C_l^{-1} in Figure 2.5, take as inputs all available prior information of \mathbf{x}_l and \mathbf{y}_l respectively, denoted $A(\mathbf{x}_l)$ and $A(\mathbf{y}_l)$. Note that for the parallel decoder $A(\mathbf{x}_0) = C(\mathbf{x}_0)$ and $A(\mathbf{y}_l) = C(\mathbf{y}_l)$, for $l=1,2,3$. Each decoder produces *extrinsic information* of the corresponding vectors, denoted $E(\mathbf{x}_l)$ and $E(\mathbf{y}_l)$. Extrinsic information on x represents the indirect information available about x obtained from the other elements in \mathbf{x} and is usually defined as

$$E(\mathbf{x}) \triangleq L(\mathbf{x}|\mathbf{r}) - L(\mathbf{x}). \quad (2.15)$$

The extrinsic information on \mathbf{x}_0 obtained from C_1^{-1} is the indirect information available about the elements in \mathbf{x}_0 obtained using the parity bits of C_1 . This information is of course relevant to the other decoders too, since they are also trying to decode \mathbf{x}_0 . Consequently, the

extrinsic information obtained from C_1^{-1} is used as prior information when C_2^{-1} and C_3^{-1} are activated, as shown in Figure 2.5. It follows that properly interleaved versions of $C(\mathbf{x}_0)$, $E(\mathbf{x}_1)$ and $E(\mathbf{x}_2)$ are used as prior information to C_3^{-1} according to

$$A(\mathbf{x}_3) = \Pi_2 \left(C(\mathbf{x}_0) + E(\mathbf{x}_1) + \Pi_1^{-1} (E(\mathbf{x}_2)) \right). \quad (2.16)$$

Note that these entities can now be added together since the decoders are working in the log-domain. All prior information available on \mathbf{y}_l is $C(\mathbf{y}_l)$ since the encoders only have the systematic bits in common. For the same reason the outputs $E(\mathbf{y}_l)$ are not connected in Figure 2.5. This sub-optimal iterative process of updating the extrinsic information and sharing it with the other constituent components decoders proceeds until convergence, subject to a stopping criterion being fulfilled. Following convergence a decision is made based on the APP of the transmitted frame \mathbf{x} , consisting of the extrinsic information and the *a priori* information, i.e., the extrinsic information obtained from the other component decoders and the intrinsic information obtained directly from the channel, according to

$$D(\mathbf{x}) = C(\mathbf{x}_0) + E(\mathbf{x}_1) + \Pi_1^{-1} (E(\mathbf{x}_2)) + \Pi_2^{-1} (E(\mathbf{x}_3)). \quad (2.17)$$

In Figure 2.6, an encoder for a serially concatenated code with three component codes, C_1 , C_2 and C_3 , is shown. Each component encoder still has as its input the vector \mathbf{x} or an interleaved version of it. Also, the output of each encoder is still a parity vector, $\bar{\mathbf{y}}_l \in \{+1, -1\}^{n_{C_l} - k_{C_l}}$. However, the difference is that the output parity bits of an encoder with a lower number are used as input to the next encoder. Consequently, parity bits on parity bits are obtained and the encoders are concatenated serially. The output is still a systematic vector, according to $\mathbf{y} = [\mathbf{x}_0, \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3]$. If C_1 , C_2 and C_3 are SPC codes, a single parity bit is generated by each encoder for each input block of size k_{C_l} . If we want to use the same code for each of the component codes, the length k of the input frame \mathbf{x} must be a multiple of k_{C_l} . This is to ensure that subsequent codes have sufficient input bits to form full SPC codewords. A concatenated code based on these three components with an input frame of size $k = (k_{C_l})^3$, results in a code rate of $(k_{C_l} / (k_{C_l} + 1))^3$.

Alternatively, the same serially concatenated code could be constructed as in Figure 2.7. Here, C_1 , C_2 and C_3 are all systematic encoders with outputs $\bar{\mathbf{y}}_l \in \{+1, -1\}^{n_{C_l}}$. Consequently, all C_l , $l = 1, 2, 3$ are now rate- $k_{C_l} / (k_{C_l} + 1)$ codes. This yields the systematic output packet $\mathbf{y} = \mathbf{y}_3$, and the code rate is still $(k_{C_l} / (k_{C_l} + 1))^3$. If not explicitly stated, the encoder from Figure 2.6 will be used henceforth.

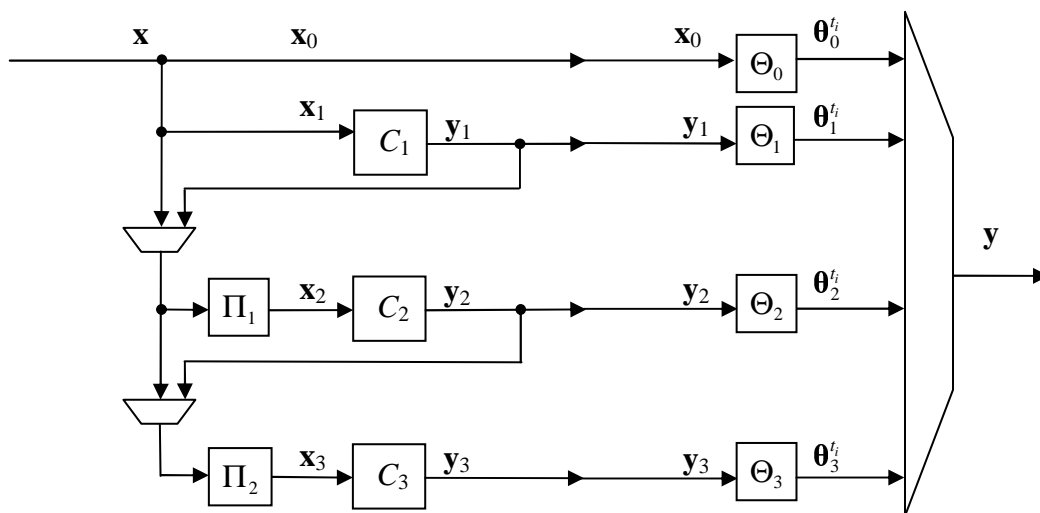


Figure 2.6. Schematically concatenated encoder with three component codes.

From Figure 2.6 it can be seen that since the code is systematic, each component code output can still be sent through individual puncturers before modulation. This results in different IR packets for transmission just as in the parallel case.

The *depuncturers* in the serially concatenated decoder still have the same function as in the parallel decoder. A depuncturer will position the received bits, $C(\theta_i^t)$ at the correct places in the component code vectors and thus the number of non-zero elements in the vectors $C(\mathbf{x}_0)$, $C(\mathbf{y}_1)$, $C(\mathbf{y}_2)$ and $C(\mathbf{y}_3)$ will increase for each additional transmission. The difference lies in the inputs to each component decoder. In Figure 2.8 the components of a serially concatenated decoder with three component codes are depicted. However, since the number of connections between the different components is large, they are omitted in the figure for clarity. In the serial case the outputs $E(\mathbf{y}_i)$ are also exchanged between the component decoders and this increases the number of connections. Recall that the output of the encoder is a systematic vector, according to $\mathbf{y} = [\mathbf{x}_0, \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3]$. Also note that $\mathbf{x}_3 = \Pi_2([\mathbf{x}_0, \mathbf{y}_1, \mathbf{y}_2])$, $\mathbf{x}_2 = \Pi_1([\mathbf{x}_0, \mathbf{y}_1])$ and $\mathbf{x}_1 = \mathbf{x}_0$. Consequently, $E(\mathbf{x}_3)$ contains information relevant for input to $A(\mathbf{x}_1)$, but only the first part of the vector is needed. Let $E(\mathbf{x}_3) = [E(\mathbf{x}_3^{\mathbf{x}_0}), E(\mathbf{x}_3^{\mathbf{y}_1}), E(\mathbf{x}_3^{\mathbf{y}_2})]$ denote the different parts of the vector of extrinsic

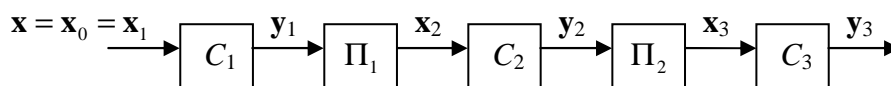


Figure 2.7 Schematically concatenated encoder with three component codes – equivalent to the version in Figure 2.6.

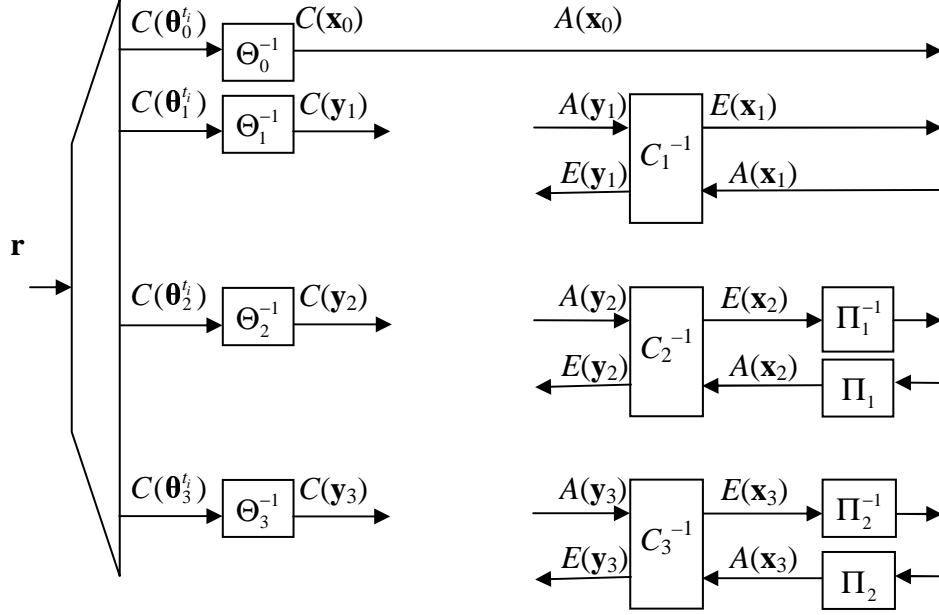


Figure 2.8. Decoder for a serially concatenated code with three components. In the serial case the outputs $E(\mathbf{y}_i)$ are also exchanged between the component decoders thereby increasing the number of connections. Hence, all connections are omitted for clarity and instead given only as equations in the text.

information obtained from C_3^{-1} . Using the appropriately interleaved version of each vector, this yields the following equations or connections:

$$A(\mathbf{x}_1) = C(\mathbf{x}_0) + \Pi_1^{-1}(E(\mathbf{x}_2^{\mathbf{x}_0})) + \Pi_2^{-1}(E(\mathbf{x}_3^{\mathbf{x}_0})), \quad (2.18)$$

$$A(\mathbf{y}_1) = C(\mathbf{y}_1) + \Pi_1^{-1}(E(\mathbf{x}_2^{\mathbf{y}_1})) + \Pi_2^{-1}(E(\mathbf{x}_3^{\mathbf{y}_1})), \quad (2.19)$$

$$A(\mathbf{x}_2) = \Pi_1 \left(\left[\left(C(\mathbf{x}_0) + E(\mathbf{x}_1) + \Pi_2^{-1}(E(\mathbf{x}_3^{\mathbf{x}_0})) \right), \left(C(\mathbf{y}_1) + E(\mathbf{y}_1) + \Pi_2^{-1}(E(\mathbf{x}_3^{\mathbf{y}_1})) \right) \right] \right), \quad (2.20)$$

$$A(\mathbf{y}_2) = C(\mathbf{y}_2) + \Pi_2^{-1}(E(\mathbf{x}_3^{\mathbf{y}_2})), \quad (2.21)$$

$$A(\mathbf{x}_3) =$$

$$\Pi_2 \left(\left[\left(C(\mathbf{x}_0) + E(\mathbf{x}_1) + \Pi_1^{-1}(E(\mathbf{x}_2^{\mathbf{x}_0})) \right), \left(C(\mathbf{y}_1) + E(\mathbf{y}_1) + \Pi_1^{-1}(E(\mathbf{x}_2^{\mathbf{y}_1})) \right), \left(C(\mathbf{y}_2) + E(\mathbf{y}_2) \right) \right] \right), \quad (2.22)$$

$$A(\mathbf{y}_3) = C(\mathbf{y}_3). \quad (2.23)$$

Finally, the decision statistics on \mathbf{x} is given by

$$D(\mathbf{x}) = C(\mathbf{x}_0) + E(\mathbf{x}_1) + \Pi_1^{-1}(E(\mathbf{x}_2^{\mathbf{x}_0})) + \Pi_2^{-1}(E(\mathbf{x}_3^{\mathbf{x}_0})). \quad (2.24)$$

Note that these decoder equations are matched to the encoder used in Figure 2.6.

Chapter 3

Incremental Redundancy Hybrid Automatic Repeat Request

The basic concepts of retransmission protocols are described in this chapter together with a detailed literature survey. The objectives behind the simple ARQ scheme and the more advanced HARQ scheme are discussed and contrasted. The fundamental mechanisms for initiating a retransmission are presented and classified as being either a one-code or a two-code approach depending on whether a separate error detection code is applied or not. The performance of a retransmission scheme can be improved by combining the received packets pertaining to the same information frame. One such packet combining method, namely code combining using rate compatible codes, is discussed in some detail. Rate compatible punctured concatenated codes with iterative decoding are presented and considered for use in HARQ schemes. The concepts of throughput, average code rate and FER in a retransmission scheme are defined and discussed. Finally, we show how to divide a rate compatible code into incremental redundancy packets of optimal lengths. The partitioning of the codeword into packets is optimal in the sense that it maximizes the average code rate given a specific puncturing pattern.

3.1 Pure ARQ Schemes

Whenever a feedback channel is available, an ARQ scheme [12] can be used. This implies that when a packet arrives, the receiver may choose *not* to accept it, but instead request a retransmission through the feedback channel. This request can be done either by sending a negative acknowledgement or simply by not sending any acknowledgement. To determine whether or not a retransmission request should be generated, the receiver checks the quality or the reliability of the received packet. Usually this is done by means of an error detecting code, e.g., a CRC code [11]. This two-way communication goes on until the receiver

obtains a packet that is considered sufficiently reliable. When this occurs the packet is accepted and an acknowledgement message is sent.

There are three basic ARQ strategies or protocols, Stop-and-Wait (SW), Go-Back- N (GBN) and Selective Repeat (SR). In an SW protocol the transmitter sends one frame at a time and waits for an acknowledgement before sending the next frame. This means that the transmitter is idle during the round-trip delay, i.e., the time between sending a frame and receiving an acknowledgement for that same frame. GBN and SR both imply that the transmitter sends frames continuously and expect to get acknowledgement messages continuously, after the round trip delay. The difference between GBN and SR lies in what happens if a frame needs to be retransmitted. GBN retransmits all frames starting with the erroneous frame that caused the retransmission, whereas SR only selects the erroneous frame for retransmission. SR is the most time-efficient protocol, but it requires buffers in both the transmitter and the receiver. GBN only requires a buffer in the transmitter, but may instead retransmit some unnecessary packets. SW is the least time-efficient protocol since it is idle during the entire round-trip delay, but does on the other hand not require any buffers and is more useful when traffic is sparse. These three protocols are extensively discussed in the literature, see e.g. [11, 32]. In this work SW is used, partly because we may expect traffic to be sparse and partly because it is simple and we are interested in the coding aspects of the ARQ scheme rather than the network protocol aspects.

3.2 Hybrid ARQ Schemes

An HARQ scheme [13] uses an error control code in conjunction with the retransmission scheme. Consequently, the received packet is first decoded and a retransmission is requested only if the quality of the decoding decision is not acceptable, i.e. if the decoded sequence is below a certain *reliability threshold*. There are different methods of determining whether a decoding decision is sufficiently reliable and hence different criteria for requesting a retransmission. The choice of method significantly affects the character of the retransmission scheme. When an error detection code, e.g. a CRC code as mentioned above, is used for requesting a retransmission, the resulting scheme is denoted a two-code approach [35], since two different codes are concatenated [23] in serial for HARQ purposes. Using a CRC code implies that the information frame is first encoded for error detection using the CRC code. Thereafter it is sent to the encoder of the error control code that will interpret the information sequence together with the redundancy added for the CRC code as one big input frame. Consequently, in the receiver the CRC detector will use the output from the error control decoder to determine whether this is a valid CRC codeword. If the decoder output is not a valid CRC codeword, the transmitter will receive a retransmission request through the feedback channel and a new packet will be transmitted. What the new packet will contain and how much redundancy it will include is determined by the type of HARQ scheme used. The retransmissions will continue until a reliable frame is obtained, which in this case means until the sequence delivered from the error control

decoder is a valid CRC codeword. Error detection using a CRC code will fail whenever the output from the error control decoder is a valid CRC codeword different to the one that was sent.

The second error detection method, termed one-code approach [35] since only the error control code is present, is to identify some sort of reliability measure within the decoding process that can be used to determine whether a retransmission is needed or not. In [36, 37] the one-code approach is used in conjunction with the Viterbi algorithm [38]. Instead of saving only the maximum likelihood sequence as is normally done in the Viterbi algorithm, the next best sequence in terms of maximum likelihood probability is also saved and compared with the best sequence. If the difference between these two sequences is below a certain threshold, a retransmission is requested.

The one-code approach using a traditional bounded distance decoder for Reed-Solomon codes is considered in [39] and [40]. A traditional bounded distance decoder assumes that a hard decision, i.e., a two-level quantization, has been made on the received sequence, resulting in what is called a received word. The bounded distance decoder will select the codeword closest in Hamming distance³ to the received word if and only if that distance is less than a certain bounded Hamming distance, [32]. If there is no codeword within the bounded Hamming distance a *decoder failure* is declared and consequently a retransmission will be requested. A bounded distance decoder will make an error whenever the received word is within the bounded Hamming distance of a codeword different to the one that was sent. A comparison between a two-code approach, using a trellis code together with a CRC code, and a one-code approach, using bounded distance decoding of Reed-Solomon codes, is made in [41]. It is concluded that the one-code approach provides better performance. In [42] a soft bounded distance decoder is evaluated. The DDC protocol was analyzed by the author in [43], using both soft and traditional bounded distance decoding of Reed-Solomon codes.

When concatenated codes are used as the error control code in an HARQ scheme, there are other possible retransmission criteria available based on the one-code approach. Typically, the iterative decoder needs a stopping criterion indicating when iterations should cease, which implicitly can be used also as retransmission criterion. This is discussed further in Section 3.3.

3.2.1 Packet Combining

There are different ways of handling packets responsible for causing a retransmission. These “erroneous” packets can either be dropped or they can be combined with one or more retransmitted packets. This procedure of using the information in previously received packets is termed packet combining and was first suggested in [14]. There are two different ways of using previously received packets in order to improve performance, i.e. two

³ The Hamming distance between two vectors is defined as the number of positions in which the two vectors differ [32].

different packet combining techniques, diversity combining, [14, 37, 44] and code combining, [15, 45, 46].

Diversity combining is typically implemented before the decoder on a symbol-by-symbol (or bit-by-bit) basis. Individual symbols from multiple identical copies of a packet are combined to create a single packet of the same length but with more reliable constituent symbols. In [44] each received symbol has a reliability value associated with it, which is updated every time a retransmission of the codeword is obtained. In [37] average diversity combining is considered, i.e., the received copies are combined into one packet by taking the symbol-by-symbol average of the packet symbols. This fairly simple strategy yields promising results. The performance of average diversity combining is further considered for trellis coded HARQ systems in [35]. Different diversity combining schemes for HARQ protocols using concatenated codes are suggested and analyzed in [17, 47].

A code combining scheme concatenates several packets pertaining to the same frame on a packet-by-packet basis to form a codeword of lower rate. In [45] only the information frame together with some bits for error detection is sent in the first transmission. If a retransmission is needed only parity bits are sent, that are then combined with the first packet to form a lower rate error control codeword. If additional retransmissions are needed, the information packet and the parity block are sent interchangeably, resulting in a truncated code combining system. In [15] each copy pertaining to the same frame is weighted by a reliability value, i.e. the known channel state information. Consequently, this method requires knowledge of the current channel conditions but results in maximum likelihood (ML) decoding [48], for an arbitrary number of packets. Punctured Reed-Solomon codes used in code combining HARQ schemes are investigated in [46].

There are different types of HARQ schemes which basically are defined by their respective packet combining method. HARQ type-I, [13] uses no packet combining and hence erroneous packets are simply dropped. HARQ type-II, [15] uses code combining so that increasingly longer codewords with lower and lower code rate are formed for each retransmission. Finally, HARQ type-III, [49] uses diversity combining implying that identical packets are retransmitted and combined to form a new more reliable packet with maintained code rate. HARQ type-III is essentially packet repetition and consequently just a variant of an HARQ type-II scheme using a repetition code.

In this work an HARQ scheme of type-II is used, i.e., a code combining scheme using IR. IR implies that the HARQ scheme responds to a retransmission request by transmitting increasingly more redundancy. This redundancy is then appended to the previously received packet, thus lowering the code rate. This is usually accomplished by so-called rate compatible punctured codes. IR-HARQ is tractable since no bits are transmitted more than once and hence repetition of previously transmitted bits is avoided. IR was first suggested in [18]. In [50] the concept of punctured codes was extended to the generation of a family of rate compatible punctured convolutional (RCPC) codes. A family of rate compatible codes implies that all coded bits that are part of a particular code in the family are also included in every other code with a lower rate in the family. The transmitter and receiver only share a puncturing table to determine which code symbols are to be transmitted next,

and the receiver simply inserts erasures for all code symbols that have not yet been received. That way, starting with a high rate code the transmitter only needs to transmit complementary code symbols to get to the next lowest rate code, and hence incremental redundancy can be accommodated. In [51] an IR-HARQ scheme using variable packet lengths is considered for WCDMA systems. An approach to reduce the signaling overhead in IR-HARQ schemes by semi-implicit retransmissions are described in [52].

3.3 Hybrid ARQ Schemes with Concatenated Codes

Concatenated codes were introduced in [23] as a way of providing long codes with manageable decoding complexity. The recently introduced turbo codes [22] are based on two constituent systematic convolutional codes joined together in parallel through an interleaver. The interleaver is used as an integrated part of the code to create longer, more powerful codes. In addition, the turbo decoder uses a modular decoding strategy and information is exchanged between the two decoders in an iterative fashion. The resulting decoder is in general sub-optimal with respect to the ML criterion, but in most cases its performance in terms of BER will converge to levels close to the ML performance for high SNR after some iterations.

The first time a turbo code was used in an HARQ scheme was in [54]. The turbo code is applied in a two-code approach, where an outer error detection CRC code is used both to implement the stopping criterion of the iterations and to generate retransmission requests. Several papers with similar ideas using turbo codes in HARQ schemes followed, e.g. [55-57]. In [58] the idea of categorizing the received packets into three classes as presented in [59] is extended to an HARQ scheme. A retransmission is requested for the packets categorized in a class where no convergence is observed. This paper was the first to use turbo codes in a one-code HARQ system. In [60] and [61] a neural network is suggested for predicting the cross entropy in the iterative decoder to be used as a retransmission criterion in HARQ schemes. In [62] the number of non-matching bits between the inner and the outer decoder, i.e. the bits the two decoders disagree on, is used as a retransmission criterion. In [24] an HARQ scheme is suggested as an application to the results given on convergence behavior and thresholding of the soft reliability information used in the iterative decoder. Thresholding the mean values of the soft output information used in the iterative decoder is suggested as a retransmission criterion in [63], similar to the approach in [24]. Finally, [64, 65] suggest using thresholding of the soft reliability information from the iterative decoder to determine the sizes of subsequent retransmissions. The resulting scheme will provide instantaneous adaptation the current channel conditions.

Rate compatible turbo codes, was first considered in [66] for unequal error protection. Rate compatible *punctured* turbo (RCPT) codes similar to the RCPC codes in [50], was considered independently by [67, 68], [69] and [19, 25] in an IR-HARQ scheme. These codes provide very good throughput and reliability. Many papers on RCPT codes used in IR-HARQ schemes followed. In [70] a serial concatenation of an inner RCPT code and an

outer Reed-Solomon code was considered. The outer Reed-Solomon code does, however, use hard decision algebraic decoding, [11]. An RCPT code is applied in an HARQ scheme in [71], with smaller packet sizes, accomplished by a so-called parity spreading interleaver. The first time a concatenated block code with iterative detection is considered in an HARQ scheme is in [72], where multiple product codes based on Hamming codes are used. In [73] the observation that a systematic parallel concatenated code can be interpreted as an equivalent punctured serial concatenated code is used in a code combining type-II HARQ scheme, with promising results. In [74] rate compatible punctured serially concatenated convolutional codes are used over AWGN and fading channels. Finally turbo trellis coded modulation is considered in [75, 76] obtaining noteworthy performance both in power and bandwidth efficiency.

Different puncturing patterns for the RCPT codes are considered in [77]. In [78] puncturing patterns are chosen to maximize the code weights of input words of weight two and three, which are known to dominate performance for turbo codes. The patterns are found through simulations. An RCPT code in an HARQ scheme using new puncturing patterns, e.g., the systematic part may also be punctured, is considered in [19]. The considerations presented in [79], when finding good component encoders for turbo codes, is extended in [80] to find good rate compatible puncturing patterns. In [69] the role of random puncturing in HARQ schemes is considered. Union bounds for random puncturing in each component code are derived in [81, 82] which is comparable to finding puncturing ratios using EXIT charts as described in Chapter 6.

3.4 Throughput, BER and Average Code Rate

The performance of ARQ schemes is often reported in terms of throughput and reliability e.g., BER. There are, however, many different definitions of throughput. There are also different methods of presenting the BER. Typically, the BER in a non-ARQ system is reported as a function of the bit-energy-to-noise ratio, E_b/N_0 . For ARQ systems the BER or the FER is usually reported as a function of the symbol-energy-to-noise ratio, E_s/N_0 or the average E_s/N_0 . This section serves the purpose of defining the methods used to present the performance of the truncated IR-HARQ schemes used in this work.

In a non-ARQ error control scheme, when decoding a packet we either succeed and correctly decode the packet with a certain probability, P_c , or the received packet is interpreted as another valid codeword, resulting in a certain probability of frame error, P_e . These probabilities are related as $P_c + P_e = 1$. In an ARQ scheme, we choose not to accept the packet if the reliability of the decoding decision is below a certain threshold, but instead request a retransmission. This results in a certain *probability of retransmission*, here denoted P_{ARQ} . The relationship between these probabilities in an ARQ scheme is depicted in Figure 3.1. If P_{ARQ} is increased consequently P_e and P_c is reduced, since $P_e^i + P_{ARQ}^i + P_c^i = 1$, where i represent the i :th transmission. Even though we want the probability of error P_e to be minimized, a large P_{ARQ} will result in numerous

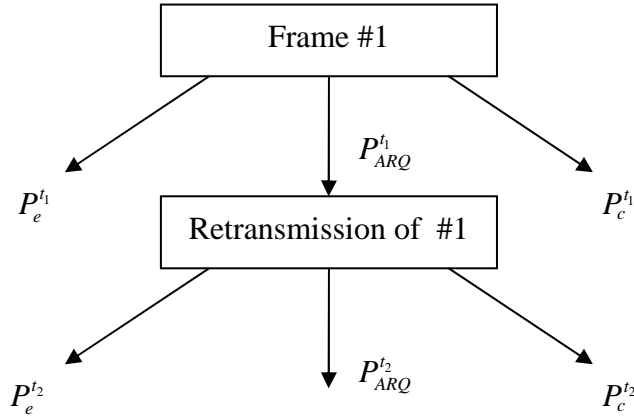


Figure 3.1. The different probabilities involved in a retransmission scheme.

retransmissions, yielding a very low throughput. Consequently, there are two different performance measures used in the evaluation of ARQ protocols, namely throughput and reliability.

Recall that a frame is a set of k information bits. The throughput, η , in a channel coding system is defined as the average number of accepted frames in the time it takes to transmit one frame. This implies that the maximal throughput that can be achieved is $\eta = 1$, which is the case for non-ARQ systems. However, in some cases the throughput is defined on a bit-level. Hence, the bit-level throughput, η_b , is defined as the average number of accepted information bits in the time it takes to transmit one frame, [32]. This implies that the maximal throughput, which is achieved for non-ARQ systems, is $\eta_b = k/n = r_c$. As can be seen the bit-level throughput is tightly connected to the code rate.

Sometimes, the throughput is instead defined as the average number of accepted *error-free* frames in the time it takes to transmit one frame, implying that we now use one performance measure within the other. Similarly, on a bit-level the throughput is then defined as the average number of accepted *error-free* information bits in the time it takes to transmit one frame. In this thesis η_b is the chosen definition of throughput and we will instead look at the accepted FER as a separate performance measure.

For an SW-HARQ scheme of type-I or type-III, let T_r be the average number of times a frame has to be retransmitted. The throughput of the system is then

$$\eta_b = \frac{k}{T_r(n + \Gamma)}, \quad (3.1)$$

where Γ is the round-trip delay expressed in terms of the number of bits that could have been transmitted during this idle time [32]. For an SW-HARQ scheme of type-II the length of the retransmitted packet is not necessarily n bits long.

A truncated HARQ scheme is considered in this thesis, implying a limited number of retransmissions, which is controlled by the QoS parameters set by the real-time constraints.

Consequently, we assume that T_{DL} together with knowledge of the bit transmission rate of the system will determine the maximum number of retransmissions allowed in order to get the required redundancy across in time. This means that while P_d determines the necessary amount of redundancy, the bit transmission rate and T_{DL} determines how many blocks the redundancy can be partitioned into, and hence also the maximum number of round-trip delays that can be afforded. *Therefore, the code rate of the system is more interesting in this context than the throughput.*

The code rate of a non-ARQ system is $r_c = k/n$, but for an ARQ-system we can only get an average code rate since the retransmissions yields a variable code rate based on the current channel conditions. Assume that we have an IR-HARQ system that is limited to two packets, $M = 2$, i.e., a packet of $n_1 = k + c_1$ is transmitted first and thereafter a block of c_2 parity bits is transmitted only when necessary, resulting in a packet of length $n_2 = k + c_1 + c_2$. Consequently, when using this scheme, some of the frames have been transmitted using a code rate of only $k/(k + c_1)$, whereas others needed a code rate of $k/(k + c_1 + c_2)$. The average code rate of the IR-HARQ system is the mean of these two code rates. Since a code rate in some sense is a velocity, the harmonic mean must be used according to

$$r_c^{M=2} = \left(a \frac{k + c_1}{k} + b \frac{k + c_1 + c_2}{k} \right)^{-1}, \quad (3.2)$$

where a and b are the percentage of frames of the respective rate. In a Monte-Carlo simulation, the average code rate used can be determined by counting the number of retransmissions made during the simulation. We can also express the average code rate as a function of the probability of a retransmission,

$$r_c^{M=2} = \frac{k}{k + c_1 + c_2 P_{ARQ}(n_1)}. \quad (3.3)$$

Note that the probability of a retransmission depends on the packet length n_1 in an IR-HARQ system. Generally, the more redundancy that is included in n_1 the smaller $P_{ARQ}(n_1)$ will be. On the other hand including more redundancy in the first transmission implies that the maximal average code rate is reduced. The expression in (3.3) can be generalize to M number of transmissions as

$$r_c^M = \frac{k}{k + c_1 + \sum_{i=2}^M c_i P_{ARQ}(n_{i-1})}. \quad (3.4)$$

If the retransmission criterion determines that the reliability of the decoding decision is not acceptable a retransmission will occur. Hence, errors will only be found among the accepted frames. The FER in an ARQ scheme, P_e is defined as the percentage of frames accepted by the receiver that contain one or more bit errors [32]. In a non-ARQ system the

probability of a frame error is $P_e = 1 - P_c$. In an ARQ systems frames are accepted after each transmission making the total FER dependent on P_e^i , for $i = 1, 2, \dots, M$. For a pure ARQ system P_e^i and P_{ARQ}^i are the same for all transmissions, since the same frame is always retransmitted and no packet combining is used. Hence, the FER for a truncated ARQ system with at most M transmissions is given by

$$\begin{aligned} P_e &= P_e^i + P_e^i P_{ARQ}^i + P_e^i (P_{ARQ}^i)^2 + \dots + P_e^i (P_{ARQ}^i)^{M-1} + P_{trunc}^i (P_{ARQ}^i)^{M-1} \\ &= P_e^i \sum_{l=0}^{M-1} (P_{ARQ}^i)^l + P_{trunc}^i (P_{ARQ}^i)^{M-1}. \end{aligned} \quad (3.5)$$

Note that following transmission M all remaining frames are accepted unconditionally and hence P_{trunc}^i denotes the fraction of frames in error in the portion that would have been retransmitted if the ARQ system was not truncated.

In an IR-HARQ scheme the blocks transmitted over the channel do not have the same contents and do not even need to be of the same length. Consequently, $P_e^i \neq P_e^{i+1}$ and depends on the length of the received packet. We therefore drop the i and instead denote the probability of frame error in accepted frames after decoding a packet of length n_i as $P_e(n_i)$. The FER for a truncated IR-HARQ system with at most M transmissions is then given by

$$P_e = P_e(n_1) + P_e(n_2) + \dots + P_e(n_M) + P_{trunc}(n_M). \quad (3.6)$$

Analogously, the BER for a truncated IR-HARQ system is

$$P_b = P_b(n_1) + P_b(n_2) + \dots + P_b(n_M) + P_{trunc_b}(n_M). \quad (3.7)$$

For simplicity, a noise-free feedback channel is assumed throughout this work, meaning that we assume that no acknowledgement messages are ever lost or corrupted. The BER is generally independent of the quality of the feedback channel, unless unnecessary retransmissions, caused by lost acknowledgement messages, are used by the decoder, in which case the BER may possibly be reduced. However, the *throughput* and the *code rate* are affected [32]. Assuming a noise-free feedback channel results in a higher throughput and a higher code rate than any physical system would have.

3.4.1 Maximizing Average Code Rate

From (3.3) it is clear that the more redundancy that is included in the first transmission, the smaller the probability of retransmission. On the other hand, a long initial packet may result in a reduced average code rate. *Given that the QoS parameters T_{DL} and P_d dictate the maximum number of allowed transmissions and the maximum amount of redundancy needed, respectively – we would like to maximize the average code rate.*

We assume that the criterion for declaring a retransmission is such that a frame that is

accepted based on a packet of length n_i would also be accepted based on a packet of length n_{i+1} and thus $P_{ARQ}(n_{i+1}) \leq P_{ARQ}(n_i)$ when $n_i < n_{i+1}$. Further, we assume that we have a given puncturing pattern, i.e., a given order in which the $n-k$ parity bits are to be transmitted. This implies that $P_{ARQ}(n_i)$ only depends on the length of the received packet n_i . The block lengths c_i , for $i=1,2,\dots,M$ can now be optimized so that the average code rate is maximized. For M transmissions this is an M dimensional discrete maximization problem according to [83]

$$[c_1^*, c_2^*, \dots, c_M^*] = \arg \max_{c_1, c_2, \dots, c_M} r_C^M, \quad (3.8)$$

where r_C^M is given by (3.4). However, since P_d has been used to determine the amount of redundancy needed, it is necessary to transmit all available parity bits in order to ensure the required QoS level. Consequently, the last transmission should always include the remaining parity bits that have not yet been transmitted so that

$$c_M = (n-k) - \sum_{i=1}^{M-1} c_i, \quad \text{where } c_i \in \{0, 1, \dots, n-k\}. \quad (3.9)$$

The discrete maximization problem then becomes $M-1$ dimensional according to

$$[c_1^*, c_2^*, \dots, c_{M-1}^*] = \arg \max_{c_1, c_2, \dots, c_{M-1}} \frac{k}{k + c_1 + \sum_{i=2}^{M-1} c_i P_{ARQ}(n_{i-1}) + c_M P_{ARQ}(n_{M-1})}. \quad (3.10)$$

Recall that P_d denotes the probability of correct delivery prior to reaching the deadline and that correct delivery implies that a certain target error rate, P_t , is met. Although the required amount of redundancy is determined by P_d , this is usually done for a particular low SNR. If the current channel conditions are improving, all the specified redundant bits may no longer be necessary. Hence, we may instead let the final transmission include only the amount of redundancy necessary to accommodate for the required P_t . Thus,

$$c_M = (n_{P_t} - k) - \sum_{i=1}^{M-1} c_i, \quad \text{where } c_i \in \{0, 1, \dots, n-k\}, \quad (3.11)$$

where n_{P_t} denote the packet length necessary to accommodate for the required level of P_t . This implies that we will maintain the specific target error rate over a range of SNRs and the amount of parity bits included in the last transmission is made SNR dependent [84].

Having obtained the optimal number of parity bits to be included in each of the M IR transmissions in order to maximize the average code rate for a given M , we can obtain the maximum average code rate as a function of the number of transmissions, i.e., for $M = 1, 2, \dots, n-k+1$. Hence, we can determine the optimal value of M , i.e., the value of M that maximizes the average code rate.

The average code rate in an IR-HARQ scheme depends on the probability of a

retransmission occurring. Ideally, we would like to have perfect error detection so that only packets containing bit errors would cause retransmissions and as soon as error free information is achieved transmissions would stop. If we assume a genie-aided retransmission criterion this results in PED. Any retransmission criterion other than PED always implies a risk of undetected errors as well as unnecessary retransmissions regardless of whether a one-code or a two-code approach is used. Consequently, using PED yields a lower bound on P_e of what is possible to achieve with a good retransmission criterion. This lower bound is fairly tight in most cases, since the probability of undetected errors is much smaller than the probability of detected errors and retransmission [32]. Assuming PED, the average code rate instead depends on the FER as

$$r_C^M = \frac{k}{k + c_1 + \sum_{i=2}^M c_i P_e(n_{i-1})}. \quad (3.12)$$

The FER defined as the percentage of *accepted* frames containing one or more bit errors for a truncated IR-HARQ system with at most M transmissions is now given by

$$P_e = P_{trunc}(n_M), \quad (3.13)$$

since all frames accepted prior to the M -th transmission are error-free due to the genie-assisted PED, $P_e(n_i) = 0 \quad \forall i \leq M$.

In order to get numerical results for the maximization of the average code rate in an IR-HARQ scheme by choosing optimal block lengths for each transmission, we need to select a code. Once this code is chosen, a particular puncturing pattern should be determined, i.e., an order in which to transmit the available code bits. Finally, using PED we need to know the FER as a function of the codeword length, for the chosen puncturing pattern. Hence the following design procedure is adopted:

1. Select a mother code to provide the required target FER.
2. Select a rate compatible coding family with corresponding puncturing patterns.
3. Select a maximum number of transmissions to provide the required target deadline.
4. Obtain the FER as a function of the codeword lengths.
5. Find optimal packet lengths that maximize the average code rate.

The remaining chapters will deal with each of the steps in this design procedure.

Chapter 4

Multiple Concatenated Single Parity Check Codes

This chapter discusses the component codes of the concatenated codes used in this thesis, namely SPC codes. These codes are relatively simple to decode and, based on low complexity, therefore ideal for use as component codes in a concatenated code. This way the concatenated code itself can instead be made more complex in the sense that we can concatenate more than two component codes. This is termed multiple concatenated codes. The performance of these codes is remarkably good given the low decoding complexity. Iterative decoding algorithms for SPC codes concatenated in parallel and serial are reviewed. Monte-Carlo simulation results of the performance of these codes are presented and discussed. The influence of the size and structure of the interleaver on the performance is exemplified. Finally, the different aspects of puncturing are discussed and exemplified.

The invention of turbo codes in 1993, [22] has shown that performance close to the channel capacity is possible, as predicted by Shannon, at reasonably complexity costs. However, the APP decoding algorithm used for each of the component codes in the turbo code is still relatively complex. Low-density parity check (LDPC) codes have also demonstrated performance that approaches the channel capacity [85], but also here decoding and encoding complexity can be high. In this thesis, the use of multidimensional concatenated SPC codes as a low complexity alternative for powerful IR-CHARQ schemes is suggested. SPC codes have a very simple structure and are thus ideal to use in a low-complexity concatenated structure.

In [26, 86] multidimensional SPC product codes are suggested. The product code structure with identical component codes implies that the information frame size grows exponentially for each dimension and thus the block length is relatively inflexible. Random interleaving of the multidimensional SPC product codes is further suggested. Based on this interpretation, the randomly interleaved product codes can alternatively be seen as multiple serially concatenated codes, since checks on checks are included. The concepts of product codes with random interleaving will be explained in more detail below. In [87] parallel

concatenated SPC codes are considered, providing more flexible block lengths. In [88-90] multiple serial and parallel SPC codes are considered where the code rates of the component codes are altered while the overall code rate is maintained. This allows for more flexibility, yielding performance versus complexity tradeoffs. The schemes are also combined with bit-interleaved coded modulation. In [91] an SPC product code is concatenated in serial with a turbo code and in [92], the results are extended to multidimensional SPC product codes.

The advantage of low-complexity decoding provided by concatenated SPC codes when compared to turbo codes is shown in [86, 93, 94]. In [95] a theorem proving the existence and uniqueness of a solution to iterative decoding of parallel concatenated SPC codes, using a product code interleaver is presented.

In this thesis, we consider multidimensional SPC product codes [26] with iterative decoding. Random interleaving as suggested in [86] for the SPC product code structure, leads to equivalent systematic concatenated SPC codes. These can be concatenated in serial or parallel, depending on whether checks on checks are included or not. This interpretation allows for extensions to the product code approach developed in [26, 86].

4.1 Encoding

The SPC codes employed in this work are constructed as follows. The information frame is arranged in a hypercube with U dimensions, where U is the number of component codes in the concatenated code. The set of component frame lengths from all dimension is then defined as

$$\{k_{C_1}, k_{C_2}, \dots, k_{C_U}\}, \quad \text{where } k = \prod_{l=1}^U k_{C_l}. \quad (4.1)$$

Each dimension l is then encoded with a systematic SPC (n_{C_l}, k_{C_l}) code, where $n_{C_l} = k_{C_l} + 1$. In the parallel case encoding is done according to Figure 4.1, i.e., either using row-column interleavers, here denoted product code interleavers, as depicted to the left or if random interleaving is employed, encoding may alternatively be depicted as in the rightmost figure. The term product code interleaver reflects that the hypercube arrangement and the row-column interleaver originate from product codes, introduced in [96]. However, when using random interleaves, the resulting codes have a stronger resemblance to concatenated block codes than to product codes, and hence the notation. Regardless of interleaver used, the code rate of the resulting parallel concatenated SPC code is

$$r_{C^{\parallel}} = \frac{1}{1 + \sum_{l=1}^U \frac{1}{k_{C_l}}}. \quad (4.2)$$

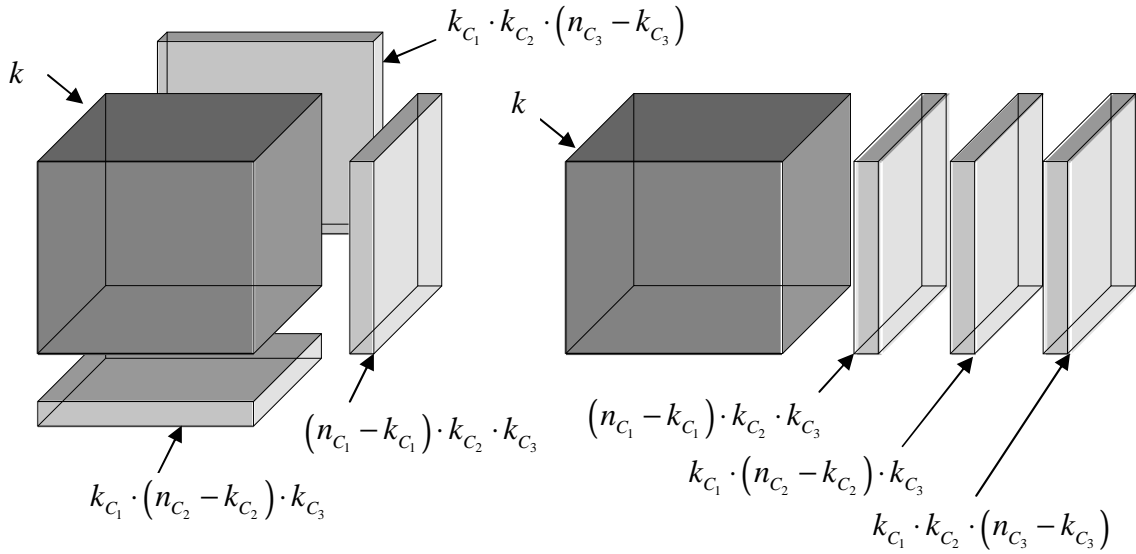


Figure 4.1. Parallel concatenated SPC code with three dimensions.

If the same component code is used in each dimension the code rate can be simplified to

$$r_{C^{\parallel}} = \frac{k'}{k'+U}, \quad \text{where } k_{C_l} = k', \forall l. \quad (4.3)$$

The minimum distance does, however, depend on the particular interleaver used. For a product code interleaver the minimum distance is

$$d_{\min}^{\parallel} = 1 + U, \quad (4.4)$$

whereas if random interleavers are used, the minimum distance may be as small as 2 and independent of U for some unfortunate combinations of interleaver selections. Performance bounds based on an average over all possible interleavers are investigated in Chapter 5.

In the serial case encoding is done according to Figure 4.2, where again product code interleavers are used to the left and random interleavers to the right. The code rate of the resulting serial concatenated SPC code is

$$r_{C^{\perp}} = \prod_{l=1}^U \left(\frac{k_{C_l}}{k_{C_l} + 1} \right) = \left(\frac{k'}{k'+1} \right)^U, \quad \text{where } k_{C_l} = k', \forall l. \quad (4.5)$$

The minimum distance for a serial concatenated SPC code also depends on the interleaver used. For a product code interleaver the minimum distance is

$$d_{\min}^{\perp} = 2^U, \quad (4.6)$$

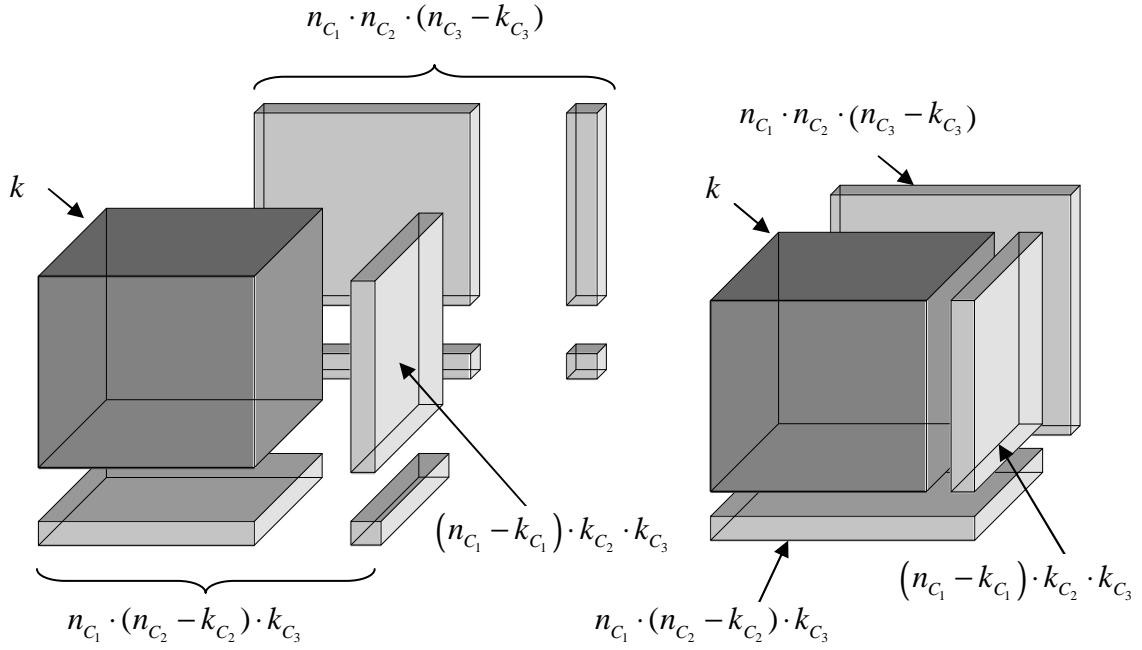


Figure 4.2. Specially concatenated SPC code with three dimensions.

whereas if random interleavers are used the minimum distance can be as small as 2 and independent of U .

When identical codes are used for each component code in a parallel concatenated SPC code, it is sufficient that the information frame size k is an even multiple of k' . However, when using identical codes in the serial case it is required that even multiples, a and b , are used so that $an_{C_i} = bk_{C_{i+1}}$, where $a, b \in \mathbb{N}$. The hypercube arrangement takes care of this.

The SPC(8,7) code is used as the component code throughout this work, both in parallel and serial SPC concatenations. Henceforth a parallel concatenated SPC (PCSPC) code with $U=2$, will be referred to as a two-dimensional (2D) PCSPC, whereas a serially concatenated SPC (SCSPC) code with $U=2$ will be termed a 2D SCSPC code.

4.2 Decoding

When performing optimal APP decoding the bit error probability is minimized by maximizing the APP of each individually bit in the sequence:

$$P(\mathbf{x}(q) | \mathbf{r}), \quad q = 0, 1, \dots, k-1. \quad (4.7)$$

Recall from Chapter 2 that $r = s + w = \mu \bar{y} + w$ and decoding is done using LLRs. The LLR of y conditioned on the matched filter output r is [34]

$$L(y|r) \triangleq \ln \left(\frac{P(\bar{y}=+1|r)}{P(\bar{y}=-1|r)} \right) = \log \left(\frac{p_r(\boldsymbol{\beta}|\bar{y}=+1)}{p_r(\boldsymbol{\beta}|\bar{y}=-1)} \right) + \log \left(\frac{P(\bar{y}=+1)}{P(\bar{y}=-1)} \right) = L_c r + L(y), \quad (4.8)$$

where $L_c = 2\mu/\sigma_w^2$. The codebook for the code C is then partitioned into two parts of equal size, one part having +1 in position q and one part having -1 in position q , according to [34]

$$\begin{aligned} L(\mathbf{x}(q)|\mathbf{r}) &= \ln \frac{\sum_{\mathbf{y} \in C, \bar{\mathbf{x}}(q)=+1} P(\mathbf{y}|\mathbf{r})}{\sum_{\mathbf{y} \in C, \bar{\mathbf{x}}(q)=-1} P(\mathbf{y}|\mathbf{r})} = \ln \frac{\sum_{\mathbf{y} \in C, \bar{\mathbf{x}}(q)=+1} \prod_{i=0}^{n-1} p_{\mathbf{r}(i)}(\boldsymbol{\beta}_i|\mathbf{y}(i)) \prod_{m=0}^{k-1} P(\mathbf{x}(m))}{\sum_{\mathbf{y} \in C, \bar{\mathbf{x}}(q)=-1} \prod_{i=0}^{n-1} p_{\mathbf{r}(i)}(\boldsymbol{\beta}_i|\mathbf{y}(i)) \prod_{m=0}^{k-1} P(\mathbf{x}(m))} \\ &= L_c \mathbf{r}(q) + L(\mathbf{x}(q)) + \frac{\sum_{\mathbf{y} \in C, \bar{\mathbf{x}}(q)=+1} \prod_{i=0, i \neq q}^{n-1} p_{\mathbf{r}(i)}(\boldsymbol{\beta}_i|\mathbf{y}(i)) \prod_{m=0, m \neq q}^{k-1} P(\mathbf{x}(m))}{\sum_{\mathbf{y} \in C, \bar{\mathbf{x}}(q)=-1} \prod_{i=0, i \neq q}^{n-1} p_{\mathbf{r}(i)}(\boldsymbol{\beta}_i|\mathbf{y}(i)) \prod_{m=0, m \neq q}^{k-1} P(\mathbf{x}(m))}, \end{aligned} \quad (4.9)$$

where the three last terms are referred to as the intrinsic information, the *a priori* information and the extrinsic information respectively. This partition is made for each $q = 0, 1, \dots, k-1$.

However, using iterative decoding we do not perform optimal APP decoding of the entire (n, k) code, but rather in each individual component SPC code separately. Thereafter, extrinsic information is exchanged between the constituent decoders in an iterative fashion. Consequently, each component decoder outputs extrinsic information and not APP. Hence we instead make the partition over the codewords in C_l according to

$$\begin{aligned} L(\mathbf{x}(q)|\mathbf{r}) &= L_c \mathbf{r}(q) + L(\mathbf{x}(q)) + \frac{\sum_{\mathbf{y} \in C_l, \bar{\mathbf{x}}(q)=+1} \prod_{i=0, i \neq q}^{n_{C_l}-1} p_{\mathbf{r}(i)}(\boldsymbol{\beta}_i|\mathbf{y}(i)) \prod_{m=0, m \neq q}^{k_{C_l}-1} P(\mathbf{x}(m))}{\sum_{\mathbf{y} \in C_l, \bar{\mathbf{x}}(q)=-1} \prod_{i=0, i \neq q}^{n_{C_l}-1} p_{\mathbf{r}(i)}(\boldsymbol{\beta}_i|\mathbf{y}(i)) \prod_{m=0, m \neq q}^{k_{C_l}-1} P(\mathbf{x}(m))} \end{aligned} \quad (4.10)$$

for each $q = 0, 1, \dots, k_{C_l}-1$, where $l = 1, 2, \dots, U$. For SPC component codes, decoding should be done on the dual code [11]. This will result in a reduced complexity since the dual code of an SPC code only has two code words in the codebook. This implies that the sums in (4.10) only have one element each and can therefore be simplified to [34]

$$L(\mathbf{x}(q)|\mathbf{r}) = L_c \mathbf{r}(q) + L(\mathbf{x}(q)) + 2 \operatorname{arctanh} \left(\prod_{\substack{i=0 \\ i \neq q}}^{n_{C_l}-1} \tanh \left(\frac{L(\mathbf{y}(i); \mathbf{r}(i))}{2} \right) \right), \quad (4.11)$$

where [34]

$$L(\mathbf{y}(i); \mathbf{r}(i)) \triangleq \begin{cases} L_c \mathbf{r}(i) + L(\mathbf{x}(i)), & 0 \leq i \leq k_{C_i} - 1 \\ L_c \mathbf{r}(i), & i = k_{C_i} = n_{C_i} - 1. \end{cases} \quad (4.12)$$

When SCSPC codes are considered, we use the extension of [34] provided in [26] so that extrinsic information can be obtained on all bits in the component codeword for $q = 0, 1, \dots, k_{C_i} - 1$

$$L(\mathbf{x}(q) | \mathbf{r}) = L_c \mathbf{r}(q) + L(\mathbf{x}(q)) + 2 \operatorname{arctanh} \left(\prod_{\substack{i=0 \\ i \neq q}}^{n_{C_i}-1} \tanh \left(\frac{L_c \mathbf{r}(i) + L(\mathbf{x}(i))}{2} \right) \right), \quad (4.13)$$

and $q = k_{C_i} = n_{C_i} - 1$

$$L(\mathbf{y}_l(q) | \mathbf{r}) = L_c \mathbf{r}(q) + L(\mathbf{y}_l(q)) + 2 \operatorname{arctanh} \left(\prod_{\substack{i=0 \\ i \neq q}}^{n_{C_i}-1} \tanh \left(\frac{L_c \mathbf{r}(i) + L(\mathbf{y}_l(i))}{2} \right) \right). \quad (4.14)$$

Note that this extension allows not only information bits, but parity bits as well to have both extrinsic and prior information. Hence extrinsic information can be calculated and exchanged also on the parity bits that are used as input bits by subsequent decoders, as is the case for a serial concatenation.

4.3 Interleaver Design

From Section 4.1, it is apparent that random interleaving may reduce the minimum distance. However, it may also reduce the multiplicity of codewords located at the minimum distance and other adjacent distances. It is mainly this feature that results in the good performance for turbo codes at low SNRs. For high SNRs, the reduced minimum distance is the cause of an elevated error floor [27]. The reason for this is that at low SNRs where the noise is high, the number of codewords close to the minimum distance is more important than their actual distances. However, when the noise is reduced for high SNRs, the actual distance is more important.

In [26] SCSPC codes using random and product code interleavers are compared. The SPC(8,7) component code is used and codes from 2D to 5D are considered. It is concluded that for a 2D SCSPC code, performance using a random interleaver is worse than when using a product code interleaver. For a 3D code, however, performance using a random interleaver is slightly better, whereas a 4D code yields performance that is considerably better when a random interleaver is used as opposed to a product code interleaver. Finally, for a 5D SCSPC code, a random interleaver provides slightly better performance than using a product code interleaver. However, in [26] the considered codes all have interleaver sizes that are related to the number of dimensions. Consequently, the 2D code has one interleaver

of size $7 \cdot 8 = 56$, the 3D code has one interleaver of size $7^2 \cdot 8 = 392$ and one of $7 \cdot 8^2 = 448$, the 4D has three interleavers of sizes 2744, 3136 and 3584, and finally a 5D has four interleaver sizes: 19208, 21952, 25088 and 28672. It is reasonable to assume that the size and not just the structure of the interleaver may influence the performance.

If instead the frame length is defined as

$$k = k_{\Pi} \prod_{i=1}^U k_{C_i}, \quad (4.15)$$

where k_{Π} denotes the interleaver size factor, i.e., the number of times the minimal block is repeated, a larger range of interleaver sizes can be catered for. Consequently, data is arranged in a $(U+1)$ dimensional hypercube even if only encoded in U dimensions. Each of the l dimensions is still encoded with a systematic SPC(8,7) and consequently the code rate remains unchanged. Since the interleaver sizes are no longer constrained by the product code dimensions, much larger interleavers are possible. In turn, these larger interleaver sizes lead to better performance at the same code rate as compared to the SPC codes with random interleavers used in [26]. In Figure 4.3, 2D, 3D and 4D SCSPC codes are compared for different interleaver structures and sizes, using $k_{\Pi} = 7$. The size of the smallest interleaver used in the system is presented in the legend together with the interleaver type used, i.e., a random or a product code interleaver. It can be concluded that for small interleavers the structure is of considerable importance. For a 2D SCSPC code using a small random interleaver, the performance is worse than using a product code interleaver, as was also concluded in [26]. However, if the interleaver size is increased slightly, here from size 56 to 392, a random interleaver is better, although an error floor can be seen. If the size of a product code interleaver is increased by a factor seven, no improvement occurs. This is due to the fact that the product code interleaver in some sense has a memory restricted to the size of the product code and hence it will simply form independent blocks that are transmitted in serial over the channel. For a small sized 2D code the product code interleaver is good since no bits joined together in a code block by one component code are joined together by the next component code. However, when the size of the interleaver is increased, better options become available.

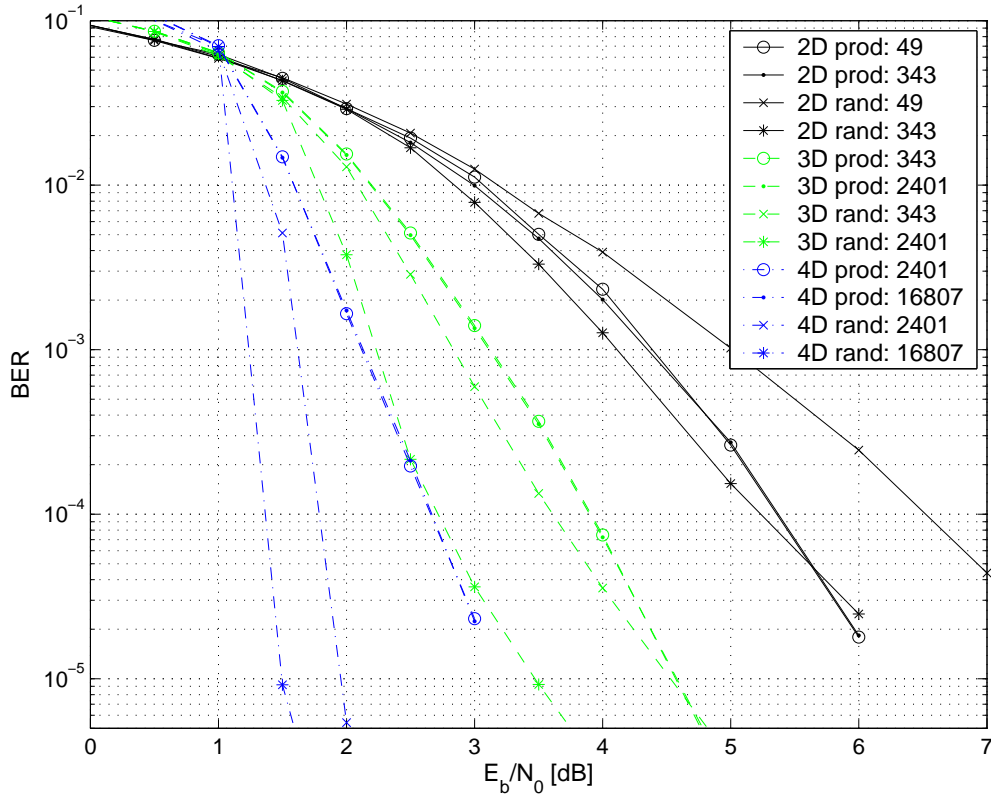


Figure 4.3. Comparison of different interleaver structures and sizes for SCSPC codes.

It can be concluded from the simulations that increasing the size of a product code interleaver implies no interleaver gain, but only generate independent sub-blocks to be transmitted in serial over the channel. It therefore seems to be the case that when designing interleavers for block codes it is important to strive for dependence between component code blocks. This implies that bits that have pertained to the same code block in one component code should not be joined together again. For a 2D PCSPC code with $k_{\Pi} = 1$ the product code interleaver is best, as long as the interleaver size is not increased beyond the size of the product code by setting $k_{\Pi} = 7$. For $k_{\Pi} = 1$, no bits joined together in the first component code are joined together in the next. However, for $k_{\Pi} = 7$ seven independent sub-blocks are formed. For a 3D PCSPC code with $k_{\Pi} = 1$, independent sub-blocks are created by the first interleaver and they are not made dependent until after the second interleaver. It is reasonable to assume that the goal should be to increase the dependencies between all bits inside the entire code block so that no small independent sub-blocks are formed. This is quite different to a good interleaver for convolutional codes, which strives to move previously adjacent bits further apart [97]. We will use random interleavers for all example codes used in this thesis.

In order to compare the performance of a serial system to a parallel, it is interesting to look at systems with fairly large interleavers to see the asymptotic behavior for long codes. In Figure 4.4 the performance of PCSPC codes using large interleavers, size $7^5 = 16807$, is

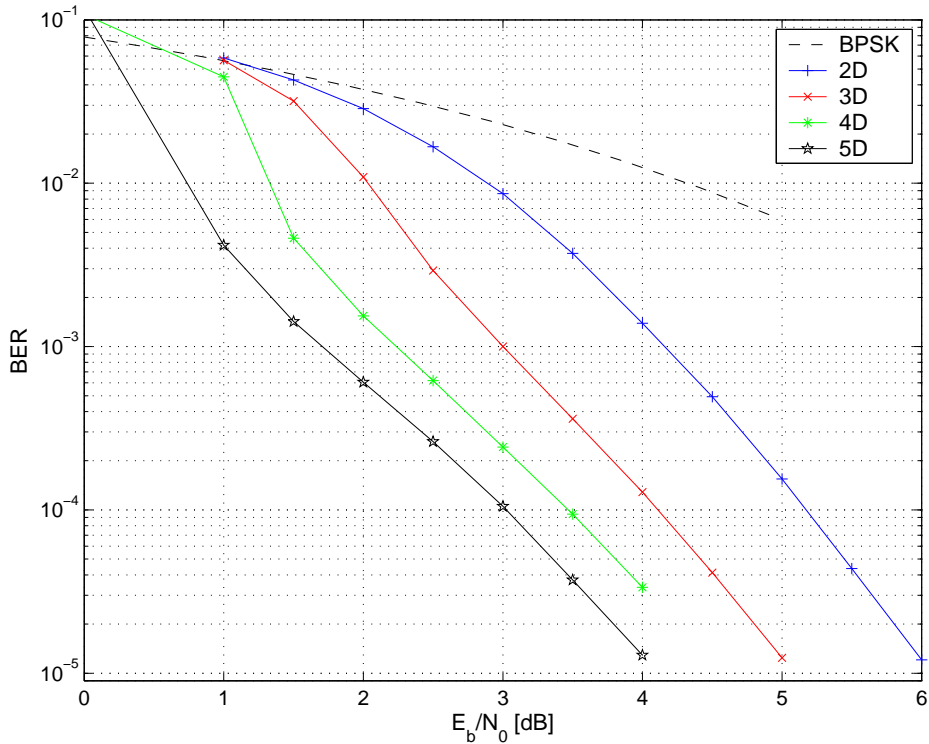


Figure 4.4. PCSPC(8,7) codes with 2-5D using large interleavers.

shown for systems with 2D-5D. In Figure 4.5 the corresponding SCSPC codes are shown. However, since a serial system implies that parity bits are applied to the parity bits generated by previous component codes, the code rate of a SCSPC code is lower than its PCSPC counterpart. Also the size of the interleavers increases with each component code and hence the outermost, smallest interleaver has size $7^4 \cdot 8 = 19208$. We can see that the parallel codes do not have any real waterfall region and the performance improves as the number of dimensions increases. The serial codes experience waterfall regions for codes with more than two components. Also, it seems as if adding more dimensions yields diminishing returns as soon as 4D has been reached. This is different from the result in [26], but the codes in Figure 4.5 are compared when all codes use same interleaver sizes, whereas in [26] the product code structure is allowed to limit the interleaver sizes so that the 4D and the 5D codes are compared using different interleaver sizes.

In the remainder of this work we will mainly focus on 3D codes, since they provide good and fairly simple examples codes. With their three component codes, they enable sufficient flexibility to make interesting examples, while retaining relatively low complexity for medium sized interleavers.

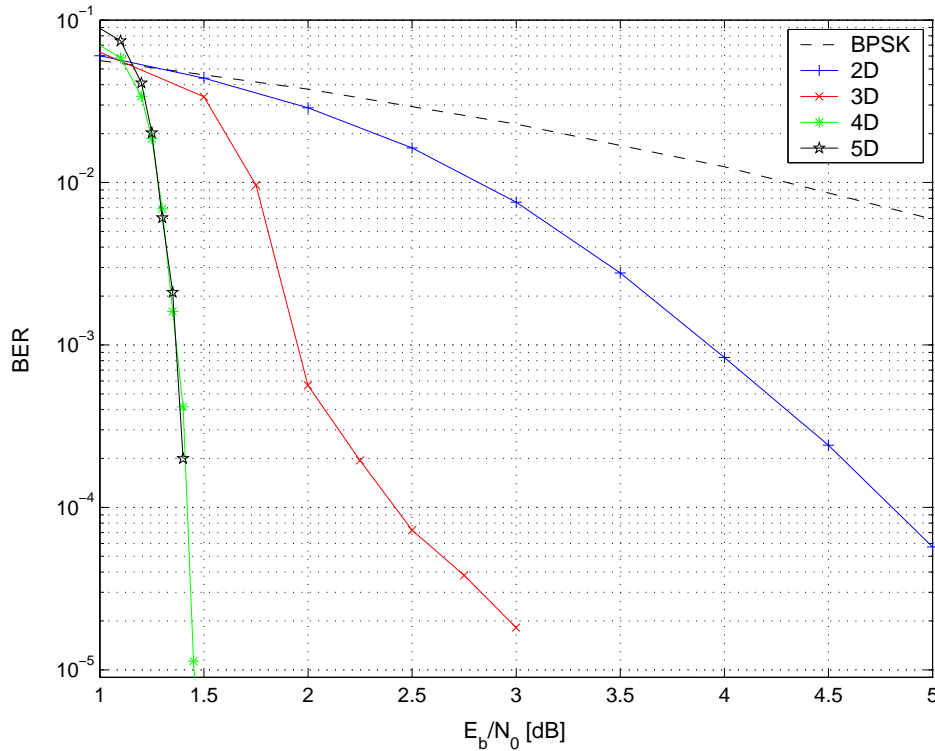


Figure 4.5. SCSPC(8,7) codes with 2-5D using large interleavers.

4.4 Rate Compatible Puncturing

Multidimensional concatenated codes are especially advantageous in conjunction with retransmission schemes, creating highly flexible and adaptive CHARQ systems. The concatenated structure with U component encoders lends itself nicely to IR-HARQ schemes, [19]. It is possible to adaptively obtain increasingly stronger codes, with increasingly lower code rate through rate compatible puncturing. Rate compatible codes are not only relevant for IR-HARQ. They are also applicable to systems using unequal error protection or adaptive systems offering a series of code rates, while using identical decoders in all receivers. With rate compatible codes, the same decoder can be used to decode all code rates. For applications requiring low decoding complexity, SPC codes are well-suited as component codes.

Consider a 3D PCSPC(8,7) code, resulting in an information frame of size $k = 7^3 = 343$. The two interleavers are then also of size 343 and each encoder adds 7^2 parity bits summing up to a total codeword of length $n = 7^3 + 3 \cdot 7^2 = 490$. Suppose we only transmit the information frame in the first transmission and thereafter half of the parity bits pertaining to C_1 in a second transmission, followed by the remaining half of the parity bits from C_1 in a third transmission, and so on. Hence, the puncturing pattern is such that parity

bits are chosen consecutively from each dimension, i.e., first parity check bits are selected consecutively among the 1D parity bits, thereafter follows consecutive selection among the 2D parity bits, and the 3D parity bits, respectively. We term this *dimension-wise puncturing*. The performance in terms of BER for the 3D PCSPC(8,7) code using this puncturing pattern is shown in Figure 4.6. BPSK shows the uncoded performance, which consequently coincides with the first transmission, when only the information frame is transmitted and hence no parity bits are included. In the second transmission, half of the parity bits in the first dimension are sent, i.e., 24 bits in this case. As can be seen from Figure 4.6 this does not offer any significant improvement, since it basically implies that half of the information bits are sent uncoded whereas the other half have been coded with one parity bits per sub-block using SPC(8,7). After transmission three, marked 49 in Figure 4.6 corresponding to the number of parity bits used in the decoding process, the minimum distance is increased from one to two, since all information bits now have a parity bit and thus the code rate is now 7/8. A full dimension is indicated by a solid curve in Figure 4.6, whereas if only part of the parity bits pertaining to a particular dimension is yet included, this is indicated by a dashed curve. It can be observed in Figure 4.6 that although the dashed curves contain half of the parity bits of the following dimension, they are not located half way between their respective solid curves. Whenever a full dimension is obtained this implies that every information bit is now protected by one additional parity bit. If we puncture one parity bit from a “new” dimension, we have consequently created a sub-codeword with a lower number of redundant bits. However, if we keep on puncturing in the same dimension, we are only increasing the multiplicity of such lower protected sub-codewords. For low SNR it may be advantageous with lower multiplicities whereas for high SNR the minimum number of parity bits per information bit is more important.

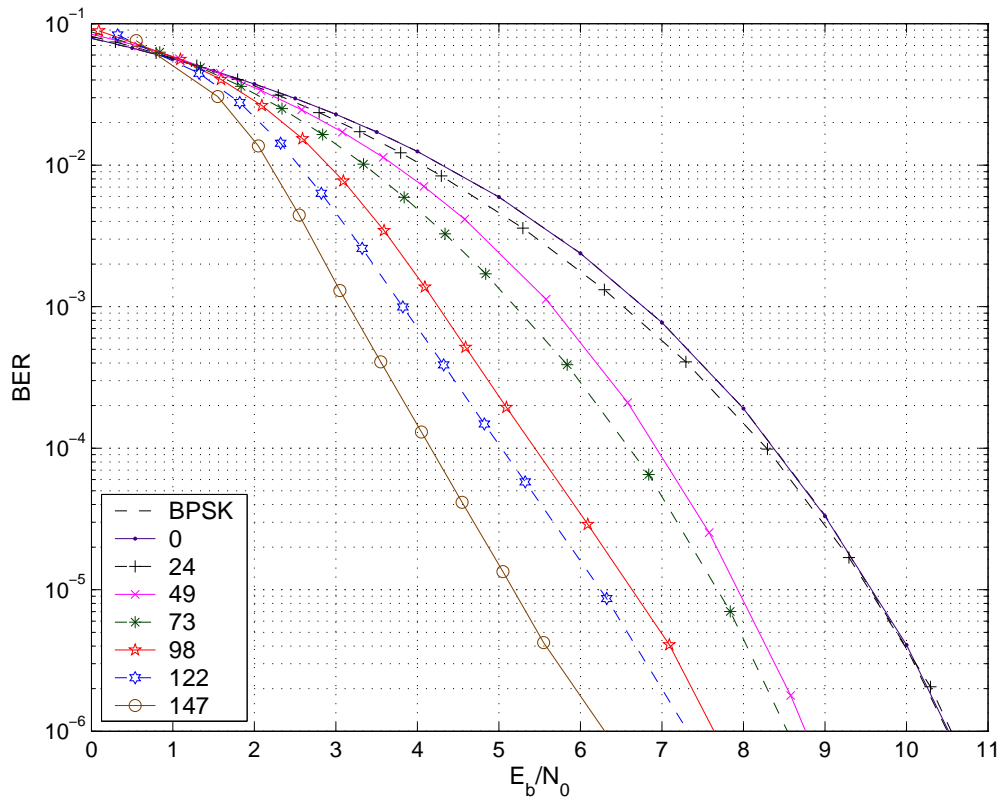


Figure 4.6. Performance of the 3D PCSPC(8,7) code for a different number of parity bits transmitted.

The performance of a comparable 3D SCSPC(8,7) code with $k = 7^3 = 343$ and interleaver sizes $7^2 \cdot 8 = 392$ and $7 \cdot 8^2 = 448$ is shown in Figure 4.7. Here the first encoder adds 7^2 parity bits, the second $7 \cdot 8$ and the third 8^2 summing up to a total codeword of length $n = 7^3 + 49 + 56 + 64 = 512$. Dimension-wise puncturing is applied, but here the maximum number of parity bits is 169 instead of the PCSPC code with its 147 parity bits, resulting in a lower code rate. For the first dimension the two systems are identical. The difference lies when the second and the third decoders are involved. We can see that better performance can be achieved when a serial code is used even for a fairly few additional parity bits.

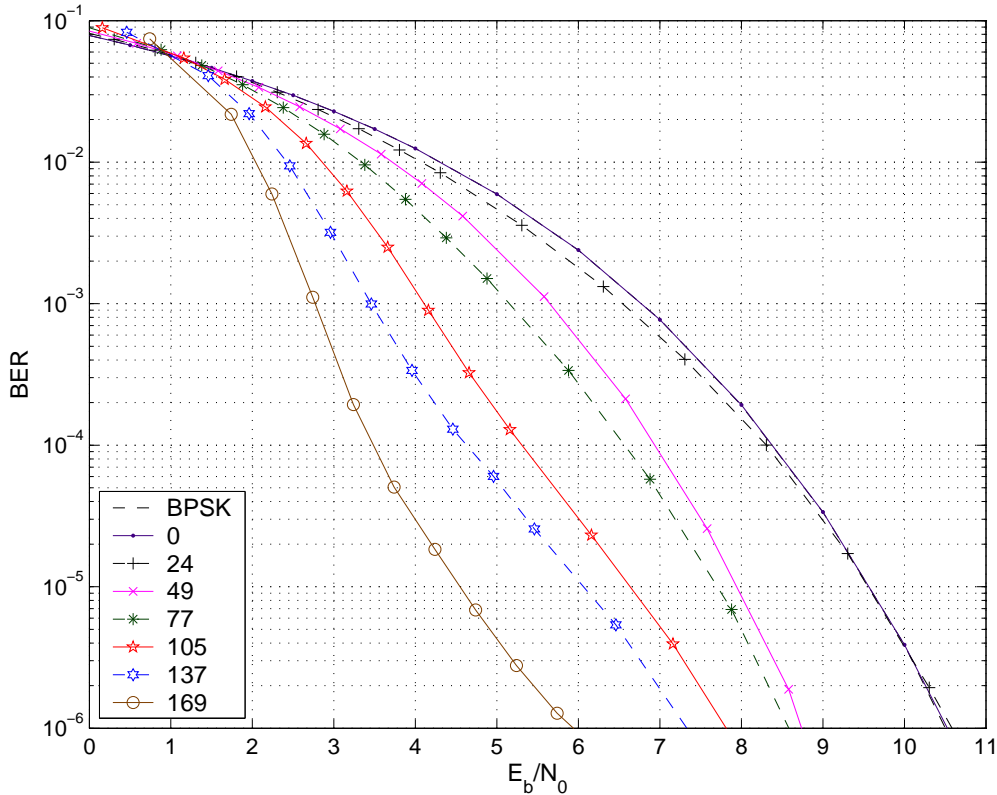


Figure 4.7. Performance of the 3D SCSPC(8,7) for a different number of parity bits transmitted.

The decoding complexity of applying SPC codes in an IR-HARQ scheme can be reduced by a conveniently chosen puncturing pattern such as dimension-wise puncturing. By systematically selecting parity bits for inclusion in IR packets, exhausting all parity bits from each SPC encoder successively, we only need to activate the constituent decoders for which the parity bits are associated with, since the number of necessary constituent decoders is determined by the number of encoders contributing parity bits to the accumulated received packets. The decoding complexity thus increases with the strength of the code, as parity bits belonging to additional dimensions arrive. In addition, the use of iterative decoding allows for “running-start” decoding, following the reception of additional IR packets. As shown in [99], given a sufficient number of activations, the activation order for multiple concatenated decoders is irrelevant for reaching the convergence fix point, as long as each constituent decoder works on all information relevant for its particular code constraints. Thus, when parity bits from a new encoder arrive we do not have to restart the decoding process, but may just include the newly arrived parity bits in the ongoing iterative decoding process.

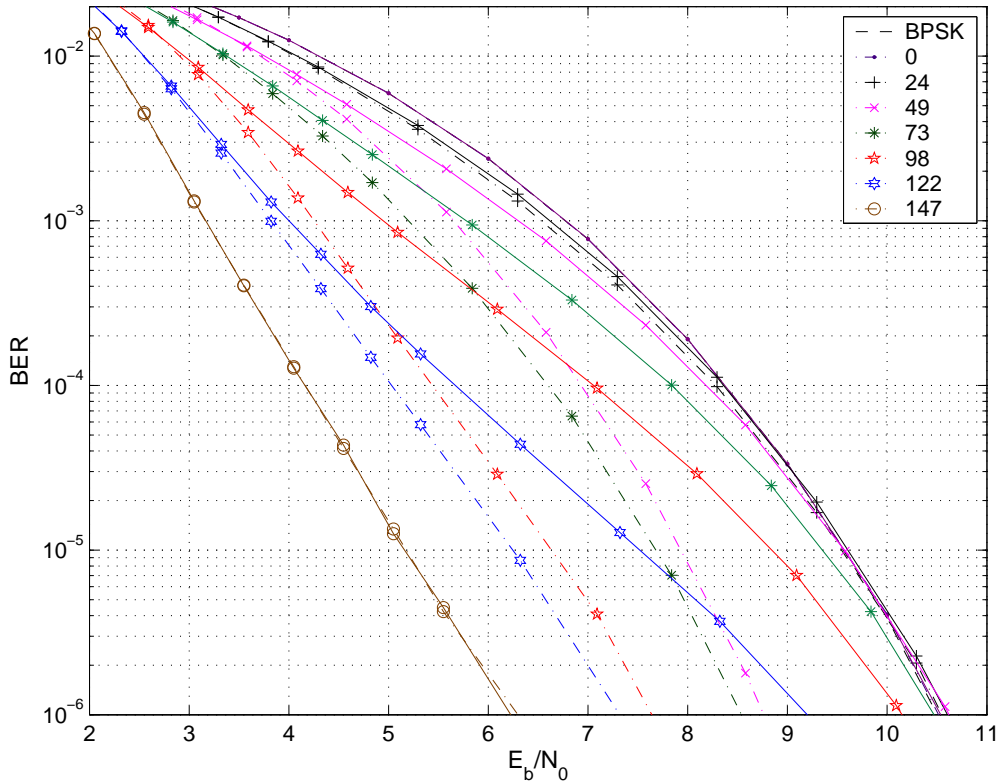


Figure 4.8. Comparison of two puncturing patterns; random (solid lines) and dimension-wise (dash-dotted lines) selection.

Even though dimension-wise puncturing is a low-complexity puncturing pattern, it does not necessarily provide the best performance. Dimension-wise puncturing is compared to random puncturing using a Monte-Carlo simulation of a 3D PCSPC code in Figure 4.8. Random puncturing is represented with solid lines and dimension-wise with dotted lines. Note that for zero and 147 parity bits the performance of the two puncturing patterns are the same since there is only one way of selecting no or all parity bits respectively. It can be seen that dimension-wise puncturing outperforms random puncturing. However, it should be noted that as soon as a Monte-Carlo simulation is used to determine the performance of a particular puncturing pattern, it may be the case that a particular interleaver favors a particular puncturing pattern. Even if the Monte-Carlo simulation is performed with a new random interleaver for every transmission, as in Figure 4.8, it is still not possible to determine the influence of the particular random interleaver selected on the performance of the particular puncturing pattern chosen. In some sense the random interleaver adds an extra dimension to the search for good puncturing patterns since one puncturing pattern may work excellent together with one particular interleaver but have very bad performance together with another. The following chapters will discuss aspects to consider when choosing a good puncturing pattern as well as how to select a pattern that is good for most interleavers. The 3D PCSPC(8,7) code in Figure 4.6 and the 3D SCSPC(8,7) code in Figure

4.7 will be used as example codes. Further, in Appendix A, a 3D PCZZ(8,7) code is analyzed and suggested as an alternative component code.

Chapter 5

Performance Analysis Using Union Bounds

Analytical expressions for upper bounds on error probabilities are presented in this chapter, based on the concept of a uniform interleaver. The error performance of the numerical examples in Chapter 4 is compared to the corresponding upper bounds derived here. The bounds are presented for both punctured and full-rate multiple parallel and serial SPC codes. Bounds for multiple parallel SPC codes with punctured information bits are also given. The bounds are found to be tight for a relatively low SNR for both BER and FER.

The FER for all possible rate-compatible codes using the chosen puncturing pattern can be obtained and used directly in the optimization procedure of Chapter 3 where the FER as a function of the block length is required. In addition, the union bounds can be used to select good puncturing patterns, which are chosen to give low FER for low SNRs.

5.1 Union Bounds on Concatenated Block Codes

The procedure for calculating the union bound on the bit and frame error probability for basic linear block codes is described below, under the assumption that ML detection of BPSK symbols over an AWGN channel is used, [33]. This procedure is then extended to provide an average upper bound for concatenated codes. The average is taken over all possible interleavers of a given length, as derived in [27] and [28] for parallel and serial concatenation, respectively. Finally, some modifications are made so that the bounds apply to multiple parallel and serially concatenated SPC codes as well as punctured versions of the same.

Assume that the binary linear block code C with parameters (n, k) is used to transmit information over an AWGN channel. Each codeword in C has k information bits and $n - k$

redundant bits, resulting in n code bits. The code bits in the codeword are transmitted over the channel using BPSK modulation. The BPSK modulation will map:

$$0 \rightarrow +\sqrt{E_s} \quad (5.1)$$

$$1 \rightarrow -\sqrt{E_s} \quad (5.2)$$

for transmission over the channel, where E_s is the symbol energy, i.e., the energy required to transmit one single code bit in the codeword. Let E_{tot} denote the total transmitted signal energy per codeword. Since there are n code bits in each codeword,

$$E_{tot} = nE_s \quad (5.3)$$

and since each codeword is based on k information bits,

$$E_b = \frac{E_{tot}}{k} = \frac{n}{k} E_s = \frac{E_s}{r_c}, \quad (5.4)$$

where $r_c = k/n$ is the code rate of code C and E_b is the total energy required to transmit a single information bit. Since the AWGN vector channel adds a random noise vector \mathbf{w} to every signal vector \mathbf{s} , each bit in a received frame is described as $\mathbf{r} = \mathbf{s} + \mathbf{w}$, where \mathbf{r} is a random variable with the following distribution:

$$\mathbf{r}(j) \in \mathcal{N}(\pm\mu, \sigma_w^2); \quad j = 0, 1, \dots, n-1 \quad (5.5)$$

or, with $\sigma_w^2 = N_0/2$, $\mu = \sqrt{E_s}$, and normalized with respect to the expectation value:

$$\bar{\mathbf{r}}(j) \in \mathcal{N}\left(\pm 1, \frac{N_0}{2E_s}\right); \quad j = 0, 1, \dots, n-1. \quad (5.6)$$

In order to provide an upper bound on the probability of error when ML decoding is performed, we start by defining the pair-wise codeword error probability [33], $P_2[\tilde{\mathbf{s}}_i, \tilde{\mathbf{s}}_l]$, as the probability of the event that the received vector \mathbf{r} is closer to $\tilde{\mathbf{s}}_l$ than to $\tilde{\mathbf{s}}_i$ when $\tilde{\mathbf{s}}_i$ is the signal vector transmitted and $i \neq l$. For any set of G equally likely signals $\{\tilde{\mathbf{s}}_i\}$, where $i = 0, 1, \dots, G-1$ and in the binary case $G = 2^k$, an upper bound on the probability of frame error, P_e can be obtained using the union bound as [33]

$$P_e(\mathcal{E} | \tilde{\mathbf{x}}_i) \leq \sum_{\substack{l=0 \\ (l \neq i)}}^{G-1} P_2[\tilde{\mathbf{s}}_i, \tilde{\mathbf{s}}_l]. \quad (5.7)$$

Here $P_e(\mathcal{E} | \tilde{\mathbf{x}}_i)$ is the probability of codeword error (frame error) conditioned on the fact that $\tilde{\mathbf{x}}_i$ was transmitted. This bound relies on the fact that the probability of a finite union

of events is bounded above by the sum of the probabilities of the constituent events, i.e. the union bound. The union bound is especially useful when the signal set $\{\tilde{\mathbf{s}}_i\}$ is completely symmetric since then the unconditional error probability equals the error probability conditioned on any particular signal in the set $\{\tilde{\mathbf{s}}_i\}$ and hence we may chose $\tilde{\mathbf{s}}_{i=0}$ as the reference signal. It follows that [33]

$$P_e \leq \sum_{l=1}^{G-1} P_2[\tilde{\mathbf{s}}_i, \tilde{\mathbf{s}}_l], \quad i = 0. \quad (5.8)$$

Any linear block code mapped onto BPSK has a completely symmetric signal set, i.e. the performance is independent of the particular codeword transmitted since the code structure is regular. For the AWGN channel $P_2[\tilde{\mathbf{s}}_i, \tilde{\mathbf{s}}_l]$ simplifies to

$$P_2[\tilde{\mathbf{s}}_i, \tilde{\mathbf{s}}_l] = Q\left(\frac{|\tilde{\mathbf{s}}_i - \tilde{\mathbf{s}}_l|}{\sqrt{2N_0}}\right), \quad (5.9)$$

where

$$Q(\beta) \triangleq \int_{\beta}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{\alpha^2}{2}} d\alpha. \quad (5.10)$$

Since we are using BPSK we have

$$|\tilde{\mathbf{s}}_i - \tilde{\mathbf{s}}_l|^2 = \sum_{j=0}^{n-1} (\tilde{\mathbf{s}}_i(j) - \tilde{\mathbf{s}}_l(j))^2 = h_{i,l} (2\sqrt{E_s})^2 = 4h_{i,l} r_C E_b, \quad (5.11)$$

where $h_{i,l}$ is the number of positions in which the two vectors $\tilde{\mathbf{s}}_i$ and $\tilde{\mathbf{s}}_l$ differs. If the all-zero codeword is used as a reference, h_i is just the Hamming weight⁴ [32] of the signal vector $\tilde{\mathbf{s}}_i$. The upper bound then follows from (5.8) and becomes

$$P_e \leq \sum_{l=1}^{G-1} Q\left(\sqrt{\frac{2h_l r_C E_b}{N_0}}\right). \quad (5.12)$$

Grouping together codeword vectors with the same Hamming weight d we can rewrite (5.12) as

$$P_e \leq \sum_{d=d_{\min}}^n A_d Q\left(\sqrt{\frac{2dr_C E_b}{N_0}}\right). \quad (5.13)$$

⁴ The Hamming weight of a codeword is defined as the number of non-zero positions in the codeword vector.

Note that we no longer sum over all G existing codewords in C , but rather over all the possible weights a codeword belonging to C can have. Recall that n is the number of bits in the codeword, and consequently also the maximal weight. A_d , called the weight distribution of the code, is the number of codewords of weight d in the code C . It follows that $A_i = 0$ when $i = 1, 2, \dots, d_{min} - 1$, where d_{min} is the minimum distance of the code C . The weight distribution is often written as a polynomial according to

$$A^C(D) = A_0 + A_1D + A_2D^2 + \dots + A_nD^n = \sum_{d=0}^n A_d D^d, \quad (5.14)$$

where D is a dummy variable. This representation of A_d is often called the weight enumerator [11] or the weight enumerating function (WEF) [27]. Since we are using A_d the expression in equation (5.13) gives us an upper bound on the probability of a frame error, P_e . If we want an upper bound on the probability of information bit error, P_b , we need to find the relation between the weight distribution of the codewords and the weight distribution of the information frames. The input-output weight distribution, $B_{w,d}$, denotes the number of codewords in C with Hamming weight d generated by information frames of Hamming weight w . The input-output weight enumerating function (IOWEF) [28], of a linear block code C is then given by

$$B^C(W, D) \triangleq \sum_{w=0}^k \sum_{d=0}^n B_{w,d} W^w D^d. \quad (5.15)$$

Note that

$$A_d \triangleq \sum_{w=0}^k B_{w,d}. \quad (5.16)$$

This yields

$$P_b \leq \sum_{d=1}^n \sum_{w=1}^k \frac{w}{k} B_{w,d} Q\left(\sqrt{\frac{2dr_c E_b}{N_0}}\right). \quad (5.17)$$

For a systematic code having $d = w + h$, where h is the weight of the parity bits, the input-redundancy weight distribution, $B_{w,h}$, may be defined as the number of codewords generated by information frames of Hamming weight w , whose parity bits have Hamming weight h , so that the overall Hamming weight is $d = w + h$. Thus, the input-redundancy weight enumeration function (IRWEF) of a systematic (n, k) block code C is defined as [27]

$$B^C(W, H) \triangleq \sum_{w=0}^k \sum_{h=0}^{n-k} B_{w,h} W^w H^h. \quad (5.18)$$

Note that

$$A_d \triangleq \sum_{w+h=d} B_{w,h}. \quad (5.19)$$

The IRWEF can be used to obtain an upper bound on the bit error probability for ML decoding of a systematic linear block code C transmitted using BPSK over an AWGN channel as

$$P_b \leq \sum_{w+h=d} \sum_w \frac{w}{k} B_{w,h} Q\left(\sqrt{\frac{2dr_c E_b}{N_0}}\right). \quad (5.20)$$

Note that the IOWEF exists for a linear block code regardless whether or not it is in systematic form, whereas the IRWEF only exists for a systematic code.

Consequently, calculating the union bounds on P_e and P_b require knowledge of the WEF and the IOWEF or the IRWEF respectively. For some codes there are closed form expressions available, but since concatenated codes have weight spectra that are dependent of the interleaver, no closed form expression is available. Unfortunately, they also have large codebooks, making the task of generating all possible codewords and counting their weights tedious and potentially infeasible. In order to overcome this difficulty, an abstract interleaver called a uniform interleaver was introduced in [27]:

Definition: A *uniform interleaver* of length k is a probabilistic device, mapping a given input word of weight w into all distinct $\binom{k}{w}$ permutations of it with equal probability $\binom{k}{w}^{-1}$.

Using the uniform interleaver, the IRWEF or the IOWEF for a concatenated code can be obtained based on knowledge of the IRWEF or the IOWEF for its constituent codes. This is tractable since the constituent codes have weight spectra that are independent of the interleaver and are based on significantly smaller codebooks.

5.1.1 Multiple Parallel Concatenated Block Codes

A systematic 2D PCSPC, based on two constituent systematic SPC codes, C_1 with parameters (n_{C_1}, k_{C_1}) , and C_2 with parameters (n_{C_2}, k_{C_2}) , linked together in parallel through an interleaver, Π_1 , will be denoted C_{2D}^{\parallel} and have parameters (n, k) . Note that when identical SPC codes are used $n_{C_1} = n_{C_2} = k_{C_1} + 1 = k_{C_2} + 1$. Since the component codes are linear, the resultant PCSPC code will also be linear since the interleaver performs a linear operation on the input bits. Let the Hamming weight of the information frame be w , and h_1 and h_2 the weights of the parity bits added by the two constituent codes respectively, then the weight of the resultant C_{2D}^{\parallel} codeword will be $w + h_1 + h_2$.

Consider now the conditional IRWEF, $B^C(w, H)$ conditioned on w . This function

enumerates the parity bits generated by the code C corresponding to the respective input frames of weight w . It is related to the IRWEF as [27]

$$B^C(W, H) \triangleq \sum_w W^w B^C(w, H). \quad (5.21)$$

It is apparent from the above definition, that the conditional IRWEF of the second code, $B^{C_2}(w, H)$, becomes independent from that of the first code, $B^{C_1}(w, H)$ due to the uniform randomization produced by the interleaver. Hence [27]

$$B^{C_{2D}^{\parallel}}(w, H) = \frac{B^{C_1}(w, H) \cdot B^{C_2}(w, H)}{\binom{k}{w}}. \quad (5.22)$$

The conditional IRWEF in (5.22) can then be used to obtain the IRWEF for C_{2D}^{\parallel} from (5.21) and finally (5.20) can be used to attain the union bound.

The upper bound obtained using the IRWEF for the C_{2D}^{\parallel} , $B^{C_{2D}^{\parallel}}(W, H)$, based on the uniform interleaver, gives the average of all the upper bounds obtained with all possible random interleavers of a given length. Consequently, for each value of E_b/N_0 , the performance obtained with the uniform interleaver can be achieved by at least one random interleaver. In (5.22) we have assumed that the interleaver size is equal to $k = k_{C_1} = k_{C_2}$. The result can easily be extended to a more general case where the interleaver is lk_{C_i} , with $i = 1, 2$ and $l \in \mathbb{N}$. This corresponds to an information frame of size $k = lk_{C_1} = lk_{C_2}$ where each component encoder is used l times for every frame. The IRWEF for this longer component code, $C_{i,l}$, with (ln_{C_i}, lk_{C_i}) can be obtained by convolving the original IRWEF l times, [27]

$$B^{C_{i,l}}(W, H) = \left[B^{C_i}(W, H) \right]^l. \quad (5.23)$$

The conditional IRWEF of the resulting code is then [27]

$$B_l^{C_{2D}^{\parallel}}(w, H) = \frac{\left[B^{C_1}(w, H) \right]^l \cdot \left[B^{C_2}(w, H) \right]^l}{\binom{k}{w}}, \quad \text{where } k = lk_{C_1} = lk_{C_2}. \quad (5.24)$$

The result in (5.24) can also be extended to the case when $k_{C_1} \neq k_{C_2}$ by choosing a and b so that $k = alk_{C_1} = blk_{C_2}$, where $a, b \in \mathbb{N}$.

When multiple component codes are concatenated in parallel the extension of (5.24) is straightforward. For example, the component codes C_1 , C_2 and C_3 concatenated in parallel and separated by two interleavers yield

$$B_i^{C_{\text{3D}}} (w, H) = \frac{[B^{C_1}(w, H)]^l \cdot [B^{C_2}(w, H)]^l \cdot [B^{C_3}(w, H)]^l}{\binom{k}{w} \cdot \binom{k}{w}}, \quad \text{where } k = lk_{C_i}, \quad (5.25)$$

for $i = 1, 2, 3$. This can be generalized to U component codes concatenated in parallel and separated by $U - 1$ interleavers as

$$B_i^{C_{\text{UD}}} (w, H) = \frac{\prod_{j=1}^U [B^{C_j}(w, H)]^l}{\binom{k}{w}^{U-1}}, \quad \text{where } k = lk_{C_j}. \quad (5.26)$$

This has also been concluded independently in [90, 100].

5.1.2 Multiple Serially Concatenated Block Codes

A SCSPC code based on two constituent codes, C_1 with parameters (n_{C_1}, k_{C_1}) and C_2 with parameters (n_{C_2}, k_{C_2}) , linked together in serial through an interleaver, will be denoted C_{2D}^\perp and have parameters (n, k) where $k = k_{C_1}$ and $n = n_{C_2}$ if we assume that $n_{C_1} = k_{C_2}$ and the system model in Figure 2.7 is used. If $w = w_1$ is the Hamming weight of the actual information frame, i.e. the information that is fed into the outer code C_1 , then d_1 will be the weight of the codeword generated by C_1 . Since the codes are serially concatenated with $n_{C_1} = k_{C_2}$, it follows that $d_1 = w_2$ and finally $d = d_2$ is the weight of the actual codeword, i.e. the codeword delivered at the output of C_2 and also at the output of C_{2D}^\perp .

Even though the SCSPC codes considered in this thesis are systematic, the bounds are still computed using the IOWEF, rather than the IRWEF, since the actual information frame is no longer the common denominator between the component codes, but rather the output of one component code is the input to the next. However, in order to obtain the IOWEF for the SCBC, denoted $B^{C_{2D}^\perp}(W, D)$, it is no longer possible to condition on w for each component code, since that is not the common denominator. If we assume that $n_{C_1} = k_{C_2} = v$ so that $d_1 = w_2 = m$ and then condition on m , we have for the outer code [28]

$$B^{C_1}(W, m) \triangleq \sum_{w_1} B_{w_1, m} W^{w_1} \quad (5.27)$$

and consequently $B^{C_1}(W, m)$ enumerates the weight distribution of the information frame that generates the respective codewords of a given weight m . For the inner code we have [28]

$$B^{C_2}(m, D) \triangleq \sum_{d_2} B_{m, d_2} D^{d_2} \quad (5.28)$$

where $B^{C_2}(m, D)$ enumerates the weights of the codewords generated by the code C_2

corresponding to the respective input words of weight m . The IOWEF of C_{2D}^\perp is then given by [28]

$$B^{C_{2D}^\perp}(W, D) = \sum_{m=1}^v \left(\frac{B^{C_1}(W, m) \cdot B^{C_2}(m, D)}{\binom{v}{m}} \right) \quad (5.29)$$

where $n_{C_1} = k_{C_2} = v$ is the interleaver size. Note that $B^{C_{2D}^\perp}(W, D)$ in (5.29) is the unconditional IOWEF and it can thus be used directly in (5.17) to obtain the union bound. If, however, $n_{C_1} \neq k_{C_2}$, d_1 will differ from w_2 and the interleaver size now has to be a multiple of codewords according to, e.g., $v = n_{C_1} \cdot k_{C_2}$. The IOWEF of the resulting code is given by [28]

$$B^{C_{2D}^\perp}(W, D) = \sum_{m=1}^v \left(\frac{[B^{C_1}(W, m)]^{k_{C_2}} \cdot [B^{C_2}(m, D)]^{n_{C_1}}}{\binom{v}{m}} \right), \quad (5.30)$$

where v is still the size of the interleaver, but now $v = n_{C_1} \cdot k_{C_2}$, since $n_{C_1} \neq k_{C_2}$.

The result can be extended to the more general case where the interleaver size is lv , where $l \in \mathbb{N}$ and the IOWEF for the resulting code is given by [28]

$$B_l^{C_{2D}^\perp}(W, D) = \sum_{m=1}^{lv} \left(\frac{[B^{C_1}(W, m)]^{lk_{C_2}} \cdot [B^{C_2}(m, D)]^{ln_{C_1}}}{\binom{lv}{m}} \right). \quad (5.31)$$

When multiple component codes are concatenated in serial an extension to (5.31) is required. For example, the C_{3D}^\perp code with the component codes C_1 , C_2 and C_3 concatenated in serial and separated by two interleavers yield an IOWEF that can be calculated in two steps according to

$$B_l^{C_{2D}^\perp}(W, D) = \sum_{m=1}^{lv_1} \left(\frac{[B^{C_1}(W, m)]^{lk_{C_2}k_{C_3}} \cdot [B^{C_2}(m, D)]^{ln_{C_1}k_{C_3}}}{\binom{lv_1}{m}} \right), \quad (5.32)$$

with $v_1 = n_{C_1} \cdot k_{C_2} \cdot k_{C_3}$ and

$$B_l^{C_{3D}^\perp}(W, D) = \sum_{m=1}^{lv_2} \left(\frac{B_l^{C_{2D}^\perp}(W, m) \cdot [B^{C_3}(m, D)]^{ln_{C_1}n_{C_2}}}{\binom{lv_2}{m}} \right), \quad (5.33)$$

with $v_2 = n_{C_1} \cdot n_{C_2} \cdot k_{C_3}$. Note that $lv_1 \neq lv_2$. A similar derivation of the IOWEF for a 3D SCSPC code has also been done independently in [90].

5.1.3 Punctured Multiple Concatenated SPC Codes

This section does not attach any importance to the performance of a particular puncturing pattern, but rather concentrates on how to generate the joint distance spectra required when calculating bounds for different puncturing strategies. The following section will focus on how to choose a good puncturing pattern based on the bounds obtained here.

Starting with parallel concatenated codes, the IRWEF for a SPC(8,7) code is

$$B^{\text{SPC}(8,7)}(W, H) = 1 + 7WH + 21W^2 + 35W^3H + 35W^4 + 21W^5H + 7W^6 + W^7H, \quad (5.34)$$

and can be obtained by simply generating all the $2^7 = 128$ codewords and counting their respective weights. If three of these SPC(8,7) codes are to be used in a 3D PCSPC code, C_{3D}^{\parallel} , with parameters (n, k) , for $k = 7^3 = 343$ that implies $k = lk_{\text{SPC}(8,7)}$, with $l = 49$. Hence, the IRWEF for these longer component codes, $C_{i,l}$, for $i = 1, 2, 3$ can be obtained by convolving the IRWEF in (5.34) $l = 49$ times,

$$B^{C_{i,l=49}}(W, H) = \left[B^{\text{SPC}(8,7)}(W, H) \right]^{l=49}. \quad (5.35)$$

Thereafter it can be used to obtain the joint distance spectrum as

$$B_{l=49}^{C_{3D}^{\parallel}}(w, H) = \frac{B^{C_{1,49}}(w, H) \cdot B^{C_{2,49}}(w, H) \cdot B^{C_{3,49}}(w, H)}{\binom{343}{w}}. \quad (5.36)$$

Assume now that we want the joint distance spectrum for the C_{3D}^{\parallel} with 25 parity bits punctured in the third dimension. The IRWEF for an uncoded SPC(8,7) is then required, or rather the weights of an information frame of size seven, since puncturing a single parity bit from an SPC code implies that the information frame is now uncoded. The IRWEF for an SPC(8,7) with the parity bit punctured is given by

$$B_{pp=1}^{\text{SPC}(8,7)}(W, H) = 1 + 7W + 21W^2 + 35W^3 + 35W^4 + 21W^5 + 7W^6 + W^7, \quad (5.37)$$

where $pp = 1$ denotes that one parity bit is punctured. Note that the information weights, W , do not change, since that would imply puncturing before encoding. The IRWEF for the component code, $C_{3,l}$, for $l = 49$ when 25 bits are punctured is given by

$$B_{pp=25}^{C_{3,l=49}}(W, H) = \left[B_{pp=1}^{\text{SPC}(8,7)}(W, H) \right]^{25} \cdot \left[B^{\text{SPC}(8,7)}(W, H) \right]^{24}. \quad (5.38)$$

Consequently, for each parity bit we wish to puncture we will use the $B_{pp=1}^{\text{SPC}(8,7)}(W, H)$ and for each parity bit we wish to keep the $B^{\text{SPC}(8,7)}(W, H)$ is used. This procedure can then be employed for each dimension we wish to puncture. Note that even though we may puncture several parity bits from one dimension, no more than one parity bit from each component codeword can be punctured, since we are dealing with SPC codes. If more advanced codes are used, each component codeword may contain more than one parity bit and that must be treated specifically since the different puncturing patterns may result in different weight spectra. For example, puncturing two parity bits pertaining to the same component codeword is likely to be different to puncturing two parity bits in the same dimension but from different component codewords. A similar scenario occurs when puncturing systematic bits in SPC codes, since each component codeword has more than one information bit. Hence, this is discussed further in the following subsection.

In Figure 5.1 the performance in terms of FER of the 3D PCSPC(8,7) example code from Chapter 4 is plotted, obtained using both the Monte-Carlo simulation and the average union bounds derived above. Recall that this example code uses dimension-wise puncturing and has an interleaver of size $k = 7^3 = 343$. The notation in the legend states the number of parity bits used in the decoding process and thus ‘24’ denotes that 24 parity bits are sent, or equivalently, that $147 - 24 = 123$ parity bits have been punctured. In Chapter 4 the performance was given in terms of BER, whereas here FER is illustrated since that is required in the optimization process of Chapter 3. Consequently, once the $B_{l=49}^{C_3^D}(w, H)$ has been obtained for different number of puncturing ratios, equation (5.19) and finally (5.13) are used to obtain the FER.

The bounds are fairly tight, although a bit pessimistic for low SNRs. Recall that the bounds assume optimal ML decoding, an infinite number of trials and yields an average based on the uniform interleaver. The simulations use sub-optimal iterative decoding, here with 60 component code activations, corresponding to at least 20 iterations, around 500000 trials and random interleavers, that may or may not be favorable compared to the average uniform interleaver. Further, the union bound relies on the fact that the probability of a union of events is bounded above by the sum of the probabilities of the constituent events. These constituent events are increasingly overlapping for lower SNR and hence the sum of these events may result in a probability that is larger than one. This, in turn results in a bound that is pessimistic for low SNRs. Since the FER can never be larger than one, the bound may simply be set to one for all values larger than one.

5.1.3.1 Puncturing of Information Bits

So far, when looking at bounds for parallel concatenated codes the IRWEF has been used since each component code simply adds redundancy to one common information frame. However, when information bits are to be punctured the IOWEF is needed. This is partly due to the fact that d can no longer be obtained from $w + h$ and partly since puncturing more information bits from a component codeword than the number of parity bits added, would lead to the weight of the redundancy, H , being lower than zero. Obviously, puncturing more bits than what has been added in terms of redundancy makes little sense in

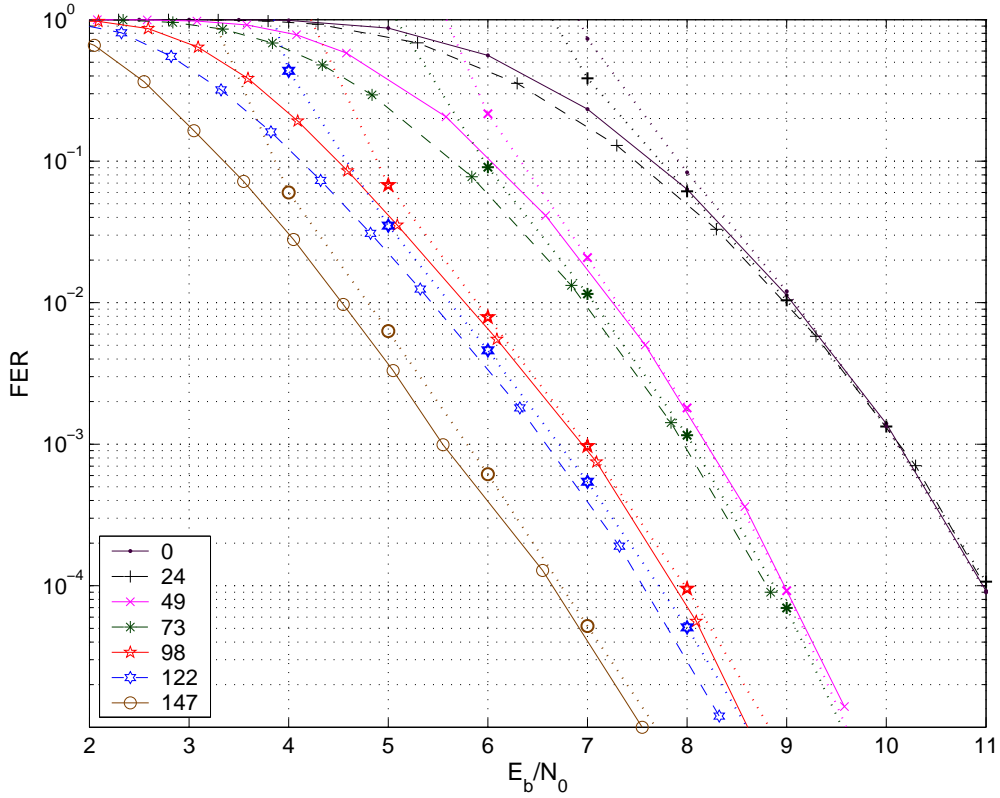


Figure 5.1. FER performance for a 3D PCSPC(8,7) code with a different number of parity bits punctured – obtained using simulations (solid and dashed) and bounds (dotted).

a stand alone component code since the invertibility of the code has to be guaranteed. When using the component code in a concatenated code this may still be relevant.

Recall that a PCSPC code can be constructed either using a separate systematic part as in Figure 2.3 or, alternatively, as a having the systematic part embedded in one of the component codes as in Figure 2.4. For the encoder in Figure 2.3, we would use three IRWEFs when obtaining the joint IRWEF of the 3D PCSPC code, whereas for the encoder in Figure 2.4, one IOWEF and two IRWEFs can be used to obtain the joint IOWEF of the equivalent 3D PCSPC code as

$$B_{l=49}^{C_{3D}^{\parallel}}(w, D) = \frac{B^{C_{1,49}}(w, D) B^{C_{2,49}}(w, H) B^{C_{3,49}}(w, H)}{\binom{343}{w}}. \quad (5.39)$$

Note that using the IOWEF of code C_1 results in the IOWEF, $B_{l=49}^{C_{3D}^{\parallel}}(w, D)$, rather than the IRWEF of C_{3D}^{\parallel} since the weight of the systematic part has already been added. All puncturing of information bits is now made in the component code contributing with its IOWEF, in this example C_1 , whereas the puncturing of parity bits is made in the IRWEF or IOWEF of the corresponding code. The information bits are common to all component

codes and this implies that if information bits are punctured it affects all component codes and hence it is only necessary to puncture in one of the weight spectra. Calculating the joint distance spectrum using the IOWEF of one of the component codes as in (5.39) is only recommended if puncturing of information bits are needed. If no systematic bits are to be punctured, it is recommended to use only IRWEFs since that makes the computation of the bound less complex.

The IOWEF of a SPC(8,7) is

$$B^{\text{SPC}(8,7)}(W, D) = 1 + 7WD^2 + 21W^2D^2 + 35W^3D^4 + 35W^4D^4 + 21W^5D^6 + 7W^6D^6 + W^7D^8. \quad (5.40)$$

Note that only even multiples of D exists since even parity is used in the SPC code. If one information bit, or rather one systematic code bit is punctured the resulting IOWEF becomes

$$B_{ip=1}^{\text{SPC}(8,7)}(W, D) = 1 + WD + 6WD^2 + 6W^2D + 15W^2D^2 + 15W^3D^3 + 20W^3D^4 + 20W^4D^3 + 15W^4D^4 + 15W^5D^5 + 6W^5D^6 + 6W^6D^5 + W^6D^6 + W^7D^7, \quad (5.41)$$

where $ip=1$ denotes that one information bit is punctured. Note that the information weight, W , remains unchanged, since we do not puncture the actual information frame, but rather a systematic code bit at the output of the encoder is punctured. Consequently, the puncturing will result in odd multiples of D .

In an SPC code two or more information bits belonging to the same component codeword can be punctured, but only one parity bit, due to the single parity structure. Whether or not two systematic bits are actually paired up in the same component codeword depends on the particular interleavers used. The uniform interleavers used when calculating the joint distance spectrum, take into account that the information bits may be paired up in the same component codeword once more in another dimension. Consequently, we only need to consider the possibility of it happening in the very first component code, before the first interleaver. Since an SPC code contains more than one information bit, the weight spectra of all the different puncturing scenarios that may occur are required. Hence component code distance spectra with one or more information bits punctured – both when the corresponding single parity bit is punctured and unpunctured are required.

If several information bits from different component codewords in the first dimension, are punctured, (5.41) is just convolved the corresponding number of times. For example, if we want to puncture 25 information bits from different component codewords the resulting IOWEF is

$$B_{ip=25}^{C_{1,l=49}}(W, D) = \left[B_{ip=1}^{C_{\text{SPC}(8,7)}}(W, D) \right]^{25} \cdot \left[B^{C_{\text{SPC}(8,7)}}(W, D) \right]^{24}. \quad (5.42)$$

Figure 5.2 depicts the performance in terms of FER for a 3D PCSPC(8,7) code with 25 punctured information bits. Note that this code has the same code rate and the same number

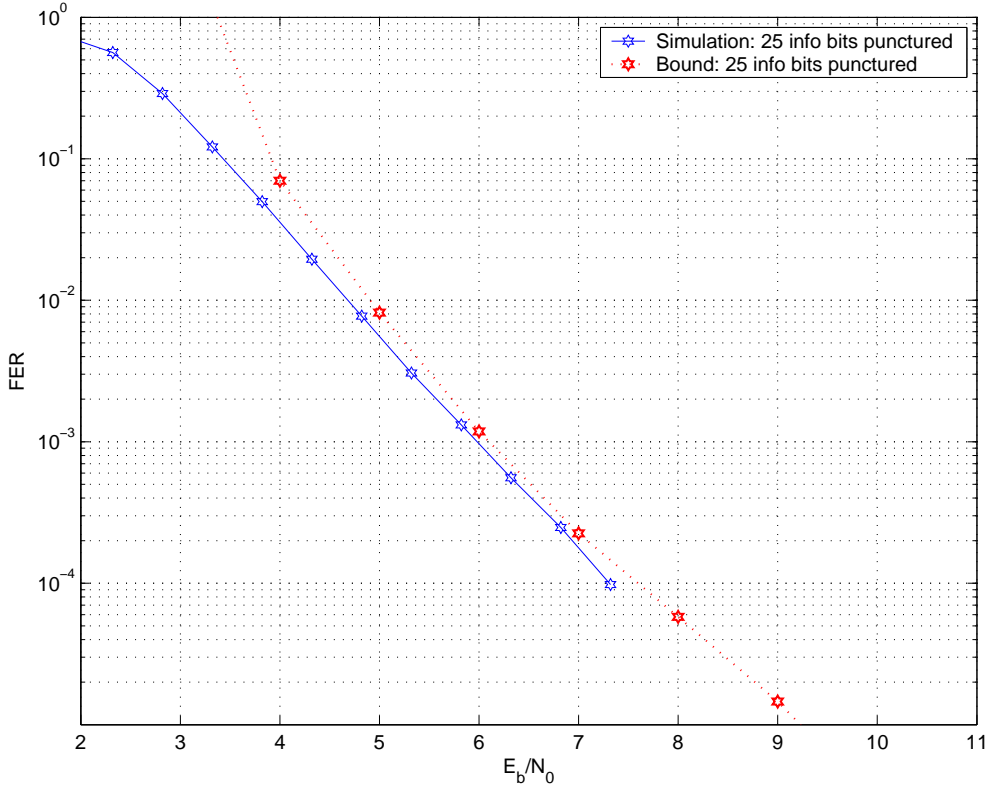


Figure 5.2. FER performance using simulation (solid line) and bound (dotted line) for a 3D PCSPC code with 25 systematic bits punctured.

of punctured bits as the code with the curve marked ‘122’ in Figure 5.1. However, the code in Figure 5.1 has 25 parity bits punctured whereas the code in Figure 5.2 has 25 information bits punctured. If we want to puncture 25 information bits and two parity bits from code C_1 the resulting IOWEF is

$$B_{ip=25,pp=2}^{C_{1,l=49}}(W, D) = \left[B_{ip=1}^{C_{\text{SPC}(8,7)}}(W, D) \right]^{25} \cdot \left[B_{pp=1}^{C_{\text{SPC}(8,7)}}(W, D) \right]^2 \cdot \left[B^{C_{\text{SPC}(8,7)}}(W, D) \right]^{22}, \quad (5.43)$$

if the punctured parity bits do *not* belong to the same component codewords as any of the punctured information bits. If, on the other hand, the parity bits are punctured from the component codewords that already have an information bit punctured the IOWEF becomes

$$B_{ip=25,pp=2}^{C_{1,l=49}}(W, D) = \left[B_{ip=1}^{C_{\text{SPC}(8,7)}}(W, D) \right]^{23} \cdot \left[B_{ip=1,pp=1}^{C_{\text{SPC}(8,7)}}(W, D) \right]^2 \cdot \left[B^{C_{\text{SPC}(8,7)}}(W, D) \right]^{24}. \quad (5.44)$$

If instead two information bits from the same component codeword are to be punctured, the IOWEF for an SPC(8,7) code with two punctured information bits is needed according to either $B_{ip=2}^{C_{\text{SPC}(8,7)}}(W, D)$ or $B_{ip=2,pp=1}^{C_{\text{SPC}(8,7)}}(W, D)$ depending on whether the corresponding parity bit is punctured or not. Note that when more than one bit is punctured from an SPC component codeword, the local invertibility for that component codeword is lost. However, overall

invertibility may still be obtained due to the addition of new parity bits in other dimensions.

5.1.3.2 Puncturing of Serially Concatenated Codes

For simplicity we will consider the 2D SCSPC code, C_{2D}^\perp , with parameters (n, k) , for $k = 7^3 = 343$, based on two SPC(7,8) codes, C_1 and C_2 . This implies that $k = lk_{C_1}k_{C_2}$, with $l = 7$ and $v = n_{C_1} \cdot k_{C_2} = 56$. C_2 being closest to the channel will be referred to as the inner code whereas C_1 is denoted the outer code. The IOWEF for these longer component codes, $C_{i,lv}$, for $i = 1, 2$ can be obtained by convolving the IOWEF in (5.40) according to

$$B^{C_{1,49}}(W, D) = \left[B^{C_{\text{SPC}(8,7)}}(W, D) \right]^{lk_{C_2}=49}, \quad (5.45)$$

and

$$B^{C_{2,56}}(W, D) = \left[B^{C_{\text{SPC}(8,7)}}(W, D) \right]^{ln_{C_1}=56}. \quad (5.46)$$

Thereafter they can be used to obtain the joint distance spectrum as

$$B_{lv=392}^{C_{2D}^\perp}(W, D) = \sum_{m=1}^{lv} \left(\frac{B^{C_{1,49}}(W, m) \cdot B^{C_{2,56}}(m, D)}{\binom{lv}{m}} \right). \quad (5.47)$$

If 25 parity bits from the inner code are punctured the resulting $B_{pp=25}^{C_{2,56}}(W, D)$ can be obtained by

$$B_{pp=25}^{C_{2,56}}(W, D) = \left[B_{pp=1}^{\text{SPC}(8,7)}(W, D) \right]^{25} \cdot \left[B^{\text{SPC}(8,7)}(W, D) \right]^{31}. \quad (5.48)$$

However, when information bits or parity bits from the outer code are to be punctured the scenario is quite different. If we try to puncture by altering the weight spectrum of the outer code, constructing $B_{pp=1}^{C_{1,49}}(W, D)$ or $B_{ip=1}^{C_{1,49}}(W, D)$ and using that in (5.47), the resulting joint distance spectrum is not correct. This is due to the fact that the output of C_1 is used as input to C_2 , and hence altering the weight spectrum of C_1 before it is used in (5.47) yields the same result as if the bits were punctured at the output of C_1 before they were encoded by C_2 . If instead the input bits of C_2 are punctured by constructing $B_{ip=1}^{C_{2,56}}(W, D)$, this results in an average of two possible weight spectra, namely that in which an information bit is punctured and that in which a parity bit from C_1 is punctured. This is due to the fact that C_2 is actually constructed assuming that all input weights are possible, whereas in effect this is not the case since it is connected to the output of another encoder that uses even parity. However, using (5.47) these ‘‘forbidden’’ input weights naturally disappears. Unfortunately, the result does not extend to the punctured case. This means that even though it is possible in practice to puncture a parity bit from C_1 given knowledge of the particular random interleaver used, it is not possible to calculate the exact bound for it using the above

equations, but merely an average – as if the puncturing was made without knowledge of the deinterleavers. For a multiple concatenated code, e.g., a 3D SCSPC, the average is over three scenarios, an information bit being punctured, a parity bit from the first dimension or a parity bit from the second dimension.

However, if we use dimension-wise puncturing, starting from the innermost code the exact bound is possible to calculate, using (5.48) and (5.47) since then we only puncture parity bits from the code that is currently the innermost one. Therefore, in Figure 5.3 the performance in terms of FER of the 3D SCSPC(8,7) example code from Chapter 4 is plotted, obtained using both the Monte-Carlo simulation and the average union bounds as derived above. Recall that this example code uses dimension-wise puncturing, starting from the innermost code and has interleavers of size $k = 7^2 \cdot 8 = 392$ and $k = 7 \cdot 8^2 = 448$ respectively. The notation in the legend states the number of parity bits used in the decoding process. Also in this case the bounds are found to be quite tight and hence useful in the optimization process in Chapter 3.

5.2 Selection of Good Puncturing Patterns

Given a family of rate compatible codes obtained from puncturing of a mother code, e.g., the 3D PCSPC(8,7) code, we would like to select a puncturing pattern that is good in the sense that it tries to maximize the throughput or the average code rate of the corresponding IR-HARQ system. By this is meant that we want to obtain as low FER as possible as fast as possible, i.e., a frame should be accepted after transmission of a rate compatible code of as high code rate as possible. Selecting a puncturing pattern by means of simulations is time consuming and far from flawless since the particular interleaver used may favor one pattern over another, whereas the roles may be reversed given another interleaver. Looking at the average union bound derived in the previous section, the distance spectrum averaged over all possible interleavers may be used to select a good puncturing pattern.

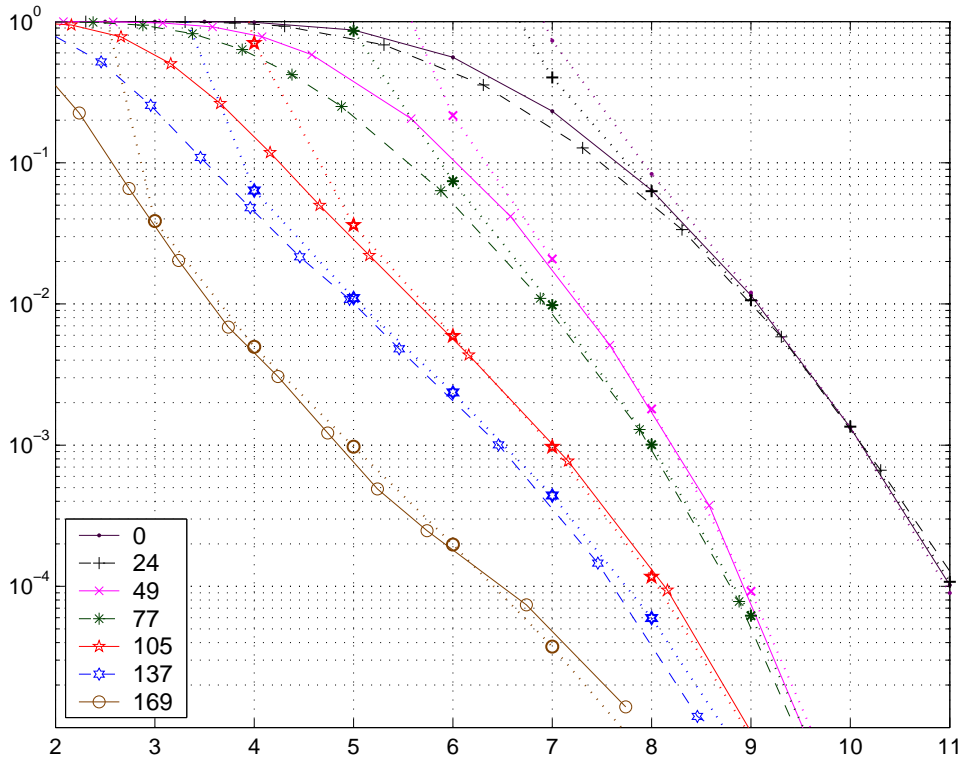


Figure 5.3. FER performance for a 3D SCSPC(8,7) code with a different number of parity bits punctured – obtained using simulations (solid and dashed) and bounds (dotted).

Several methods to select the better candidate given two alternative puncturing patterns and their respective distance spectrum have been suggested. Previous work has for example proposed to fit a regression line to the first 30 components of this joint weight spectrum and then selected the puncturing pattern which has a regression line with a minimum slope, [19]. Another suggestion is to pick the puncturing pattern that results in the maximal minimum distance, and if there is a tie, pick the one with the lowest multiplicity at this distance, [19, 80]. Since the latter suggestion first maximizes d_{min} and secondly minimizes the multiplicities, it is likely to choose a pattern that performs well at high SNRs where the minimum distance is the dominating factor. At low SNRs, higher weights than the minimum distance yield a non-negligible contribution and hence the multiplicities at the closest distances are more dominating in this area. The regression line suggestion provides a measure of the rate of growth of the weight multiplicities and it is thus likely to find patterns that perform better at low SNRs. Since an IR-HARQ system generates retransmissions when the SNR is low, we are interested in patterns that perform well in this area. Once the joint distance spectrum has been calculated, obtaining the bound is fast and simple. Further, the bound on FER for multiple concatenated SPC codes is found to be fairly tight in the relevant SNR region. Consequently, in this work the candidate puncturing pattern that results in the lowest bound on the FER in the SNR region of interest for the

chosen IR-HARQ scheme is preferred.

In order to obtain different candidate puncturing patterns to compare, we use the following procedure, which is similar to the procedure used in [19]:

1. Define an all ones vector, \mathbf{p} , of the same length as the number of bits that is possible to puncture. The vector \mathbf{p} is typically of length $n - k$. Set the code rate to the lowest allowed value $k/n_{i=M}$, where $n_{i=M}$ is the length of the mother code and the resulting block length after all of the M allowed transmissions have been made.
2. Select the next highest rate in the rate compatible code family by setting $i = i - 1$ so that transmission, t_i , is considered.
3. Form a set of all possible candidate puncturing patterns of this rate. This is done by selecting the appropriate number of bits among the elements in \mathbf{p} that are set to one, and temporarily resetting them, which symbolizes that the corresponding bits are punctured.
4. Calculate the union bound on the FER for each of the puncturing patterns obtained in Step 3.
5. Select the candidate pattern with a bound that yields the lowest FER in the relevant SNR region. If there is a tie, increase the SNR region.
6. Assign the winning puncturing pattern to the corresponding rate, $\mathbf{p}(t_i)$ and permanently reset the corresponding positions in \mathbf{p} .
7. If $i > 1$, continue to Step 2, otherwise the search is finished.

Assume that we want to maximize the number of members in the rate compatible code family, by allowing bit-by-bit puncturing. This results in a rate compatible code family with all possible code rates available, from the rate of the mother code up to rate one. These rates imply that only one position at a time is reset in Step 3.

Consider a 3D PCSPC code with all possible code rates available, when only parity bits are allowed to be punctured. The above search procedure can then conclude that *the best puncturing pattern is dimension-wise puncturing*, i.e., if we have started puncturing bits pertaining to one encoder, we should continue doing so until all these bits are punctured. Hence parity bits from a “new” encoder should only be punctured once all the bits from the “old” encoder have been removed. Therefore, the 3D PCSPC example code in Chapter 4 does in fact already use the best possible puncturing pattern. In some sense this result is intuitive, since each time we start puncturing a new dimension, the maximum number of parity bits protecting an information bit is reduced. However, if we continue to puncture in the same dimension, we no longer affect this maximum number, but are only increasing the multiplicities of these lower protected information bits – something that has less influence at low SNRs.

Due to the very simple structure of SPC codes it is irrelevant which parity bit from a particular dimension that is punctured. Further, for PCSPC codes it is irrelevant which dimension is chosen first when, e.g., a dimension-wise puncturing pattern is to be used. This drastically reduces the number of possible candidate puncturing patterns in Step 3. In fact, the only two candidates that needed to be compared are “continue to puncture in the

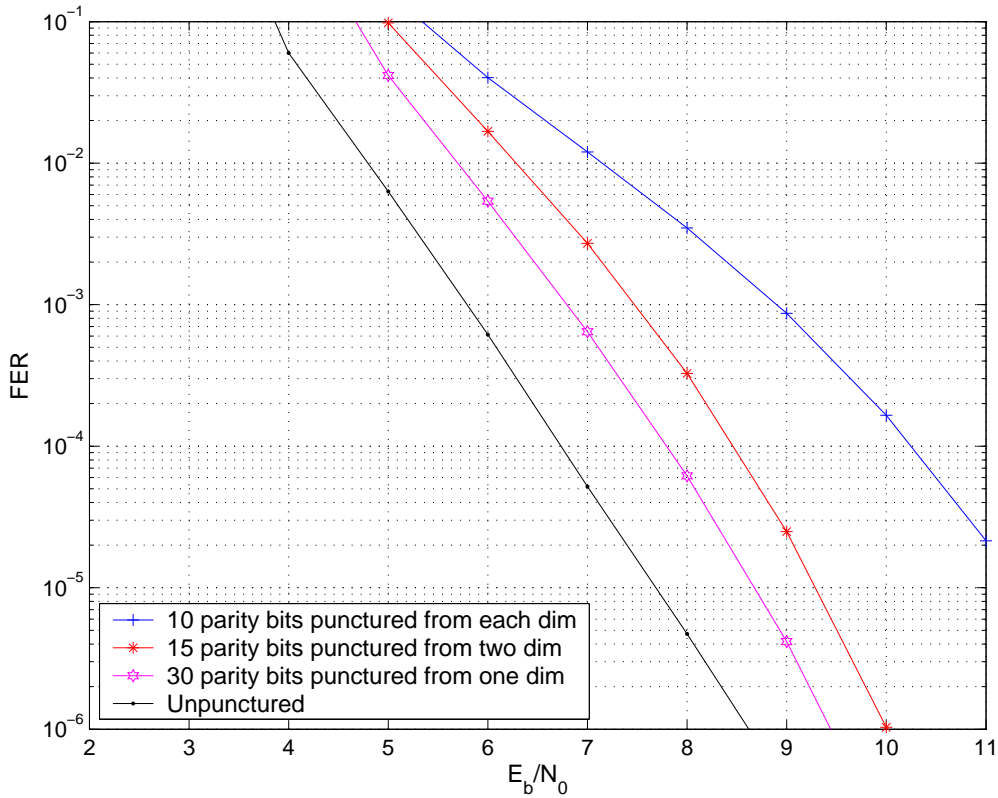


Figure 5.4. Comparison of three different puncturing patterns for a 3D PCSPC(8,7) code.

same dimension” versus “start puncturing a new dimension”. If we instead want a different set of rate compatible codes, more than one bit may be punctured concurrently in Step 3. A search for the best puncturing pattern when say 30 parity bits are punctured simultaneously may not result in the same pattern since this implies a different rate compatible code family. The search also inherently leads to more options and hence more time consuming searches if all code families should be considered. Therefore the resulting dimension-wise puncturing pattern is only guaranteed to be the best for the rate compatible code family including all possible rates. However, due to the regular structure of the PCSPC codes, it is likely that the result will be the same regardless of code families. As an example, compare three different candidate patterns that all puncture 30 parity bits. One punctures 10 parity bits from each of the three dimensions, the second 15 parity bits from two dimensions, whereas the last punctures all 30 parity bits in the same dimension. The resulting bounds on the FER are shown in Figure 5.4. As can be expected dimension-wise puncturing is the best candidate.

Puncturing information bits yields more candidate puncturing patterns in Step 3, since each component codeword has more than one information bit. For an unpunctured component codeword it does not matter which information bit is chosen, only how many bits in total, information or parity, that are punctured. Figure 5.5 shows the performance when two code bits are punctured, for all different puncturing patterns that result in a

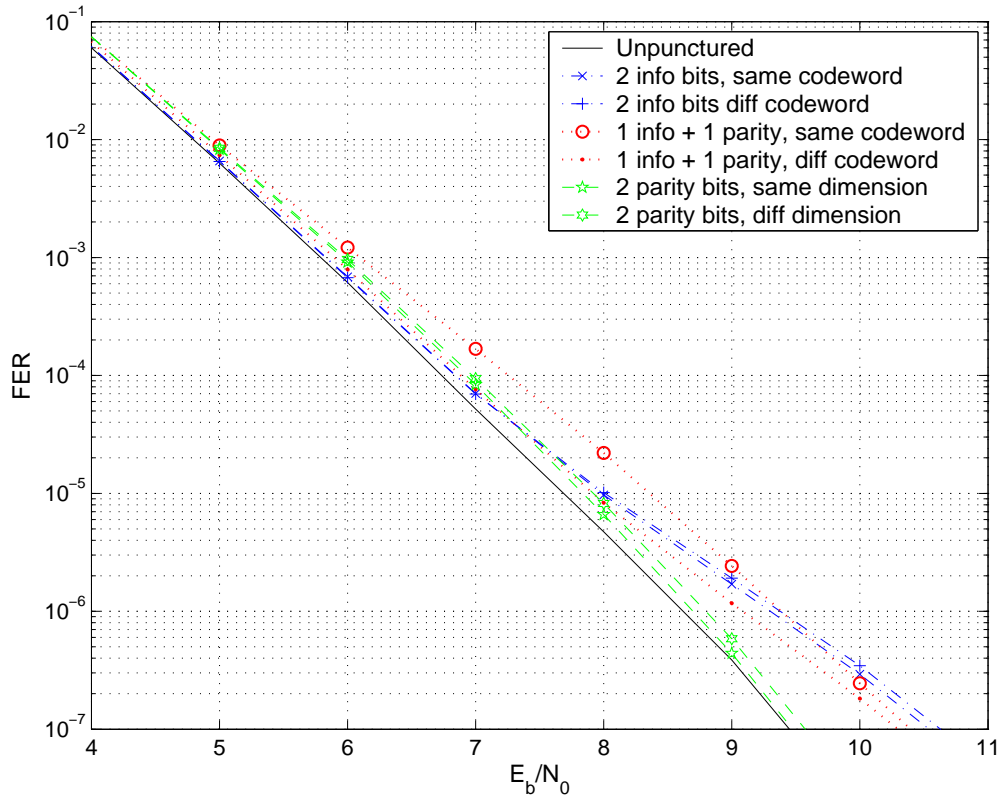


Figure 5.5. Comparison of different puncturing patterns for a 3D PCSPC(8,7) code.

unique joint distance spectrum. The first thing that can be noticed is that the performance is different for different SNR regions. In the search including only parity bits, the winning candidate always performed better for all SNRs. Here, it seems as it is best to puncture information bits for low SNRs and parity bits for high SNRs. Also, we can conclude that it is always better to choose the two bits from different component codewords. When puncturing a parity bit in a particular dimension, the minimum distance for the corresponding component code is reduced. When puncturing an information bit, the minimum distance of all component codes is reduced, since the systematic information bit is part of all component codes. Consequently, at high SNR, where the minimum distance is the dominating factor it is best to puncture two parity bits in the same dimension since that only reduces the minimum distance in one dimension. Thereafter it is best to puncture two parity bits from different dimensions, thus reducing the minimum distance in two dimensions. The third best option is one information bit and one parity bit from different codewords since that reduces the minimum distance in all three dimensions. At lower SNRs the multiplicities are more important and hence it seems preferable to puncture information bits.

It is interesting to notice that puncturing one information bit and one parity bit from the same component codeword yields worst performance over almost the entire range of SNRs. This is due to the fact that the local invertibility of that particular component codeword is

lost since we have punctured two code bits from a codeword that only has had one parity bit added. Depending on the particular interleavers chosen, the remaining dimensions may restore the invertibility if the corresponding code bits are not punctured. This means, however, that we cannot be certain that the overall invertibility is guaranteed, due to the use of uniform interleavers. Obviously, the same scenario may occur when two information bits from the same component codeword are punctured – or from different component codewords for that matter. In fact, the only time we can ensure that the invertibility is guaranteed is when only parity bits are punctured or in the special case when only one information bit together with parity bits from only one dimension and from different component codewords are punctured. In all other cases there exist interleavers that may violate the invertibility. Since the bound and the average weight spectrum considers all possible interleavers it also considers these ones. Whenever the invertibility is lost it is implied that a particular information bit or bits can never be recreated even if the channel is noiseless. Hence, it is not possible to tell whether the BER is good due to good performance of the puncturing pattern or due to such an unequal error protection that one or a few information bits have a probability of error equal to 0.5. Consequently, the joint distance spectrum based on uniform interleavers is not convenient for selecting a good puncturing pattern when puncturing of information bits is allowed.

In [19], where the joint distance spectrum is used to determine good puncturing patterns, puncturing of information bits is considered. However, the puncturing of information bits is not included in the subsequent search for puncturing patterns. Instead it is determined by means of simulations that it is good to puncture information bits for low SNRs. Thereafter, a few information bits are always punctured in the first transmission in order to give room for more parity bits from at least two dimensions so that iterative decoding can take place as early as possible. The second transmission always includes the remaining information bits that were punctured in the first transmission. Finally, the search for good puncturing patterns takes place after these initial transmissions.

In [101] the performance of partially systematic turbo codes is investigated. Partially systematic implies that information bits are punctured. Turbo codes are, however, based on recursive convolutional codes, which are more complex than SPC codes. This in turn makes them more robust against puncturing, the recursive structure allows more flexible puncturing and the component codewords are of the same size as the interleaver. Hence, it may not be possible to make any conclusions about punctured concatenated SPC codes with component codewords of size seven based on the analysis of punctured turbo codes. Some of the result may, however, still be relevant. The performance evaluation of the selected puncturing patterns in [101] is made mainly using simulations. Analytical results based on the joint distance spectrum using uniform interleavers are obtained and compared to the simulations and as in Figure 5.2 the analytical result matches the simulation. It is concluded that puncturing a few systematic bits leads to improved waterfall regions, whereas extensive puncturing deteriorates the waterfall region with respect to the unpunctured case. It may be the case that the probability of losing the invertibility is fairly small when puncturing only a few information bits for interleavers of sufficient size,

whereas eventually loosing the invertibility will become inevitable, when the puncturing ratio is increased.

Note that with a simulation, the interleaver, although random, is known and hence the invertibility can be guaranteed by appropriate *ad hoc* puncturing of information bits. The simulation in Figure 5.2 is however, not optimized this way. The information bits are chosen from different component codewords in the first dimension, but thereafter randomly selected interleavers are used. This was made in order to mimic the assumptions for the bound calculations.

In [102] different aspects of punctured serially concatenated turbo codes are considered. In particular, interleavers for punctured codes are discussed. It is concluded that the unequal error protection introduced by puncturing may be reduced to more uniform error protection by a properly designed interleaver.

In [103] good puncturing patterns for serially concatenated convolutional codes are obtained using simulations. In order to guarantee the local invertibility, only parity bits from the inner or the outer code are punctured. However, when using simulations to obtain good puncturing patterns it is difficult to determine the interleaver influence on the particular puncturing pattern chosen.

If we attempt to use bounds to search for good puncturing pattern for SCSPC codes, the same problem occurs. Recall that it is only possible to calculate the average bound when information bits *or* parity bits from outer component codes are punctured. Hence, the local invertibility can not be guaranteed by only puncturing parity bits or by *ad hoc* puncturing of information bits and thereby the joint distance spectrum is not appropriate to use for this case. Instead, EXIT charts can be used for performance prediction of punctured SCSPC codes, following the approach detailed in Chapter 6.

Chapter 6

Performance Analysis Using EXIT Charts

An EXIT chart is a powerful semi-analytical tool used to monitor and analyze the convergence behavior of an iterative decoder as a function of the number of iterations, [29]. EXIT charts can also be used to obtain an estimate of the BER performance and convergence thresholds for punctured multiple concatenated codes, [29, 30]. The procedure is described in this chapter together with the necessary tools required to construct EXIT charts. Hence, a description of entropy and mutual information is first given. This is then followed by a description of Shannon's channel capacity and the implications that follows from this. Thereafter, EXIT charts are constructed for both multiple parallel and serially concatenated SPC codes. The EXIT chart is two-dimensional when two component codes are concatenated, but multidimensional when more than two codes are used. Hence, projections onto two dimensions are necessary in these cases, [30].

Although EXIT charts assume infinite interleavers and model the inputs and outputs to the respective decoders as Gaussian random variables, they still manage to estimate the BER quite accurately in the majority of cases considered. Unfortunately, there is a numerical problem for very low values of the BER, but in most cases the BER estimation is quite good for low SNR values. Assuming that the bit errors are randomly distributed in all frames, a scenario that can be obtained in a wireless channel using a channel interleaver in addition to the interleavers used within the concatenated code, we can get an approximate value of the FER to be used in the optimization process of Chapter 3. Thus, in cases where it is not possible to obtain a bound, this approach provides a valid alternative.

EXIT charts can also be made for punctured codes. However, in this case we do not know exactly which bits are punctured, since EXIT charts assume infinitely interleavers and random puncturing. Hence, we only get a puncturing ratio for each encoder that tells us what percentage of bits to puncture. Luckily, for SPC codes it does not matter which bit from a particular encoder is punctured as long as random interleavers are used, and hence for SPC codes the puncturing ratios also gives the puncturing pattern. Consequently, we

can search for good puncturing ratios for specific code rates that yield good performance in terms of convergence at a low SNR. We can also search for good rate compatible codes using EXIT charts in a similar way.

6.1 Entropy and Mutual Information

In the following two sections, describing the general concepts of entropy, mutual information and channel capacity, x and y are redefined to be random variables with no connection to the system model. Thereafter, the systems used in this work are analyzed using EXIT charts and hence x and y are once more defined according to the system model.

The entropy for a discrete set of probabilities $P(x_1), \dots, P(x_l)$ has been defined as [10]:

$$H(X) \triangleq -\sum_{i=1}^l P(x_i) \log P(x_i) = -E[\log P(X)], \quad (6.1)$$

where $E[\cdot]$ denotes the expected value of the random variable. The entropy $H(X)$ denotes the uncertainty of the discrete random variable X . For a given l , $H(X)$ is maximum and equal to $\log l$ when all the $P(x_i)$ are equal. $H(X) = 0$ if and only if all but one of the $P(x_i)$ are zero while the remaining probability has unit value. Hence, only if we know the outcome $H(X) = 0$, otherwise the entropy of a discrete random variable is always positive.

The conditional entropy is defined as [10]:

$$H(X|Y) \triangleq -\sum_{i=1}^l \sum_{j=1}^m P(x_i, y_j) \log P(x_i | y_j) = -E[\log P(X|Y)]. \quad (6.2)$$

$H(X|Y)$ denotes the entropy of the discrete random variable X after the discrete random variable Y is known. Hence, if $H(X)$ is the entropy of X , then $H(X) - H(X|Y)$ denotes the amount of uncertainty that has been removed when Y is revealed. Thus, $I(X;Y) \triangleq H(X) - H(X|Y)$ is the amount of information provided by Y about X . $I(X;Y)$ is called the average mutual information (MI) between the two discrete random variables X and Y and is given by [48]:

$$I(X;Y) \triangleq \sum_{i=1}^l \sum_{j=1}^m P(x_i, y_j) \log \frac{P(x_i, y_j)}{P(x_i)P(y_j)}. \quad (6.3)$$

Analogously, the average mutual information between two continuous random variables X and Y with a joint PDF $p_{x,y}(\alpha, \beta)$ and marginal PDFs $p_x(\alpha)$ and $p_y(\beta)$ respectively, has been defined as:

$$I(X;Y) \triangleq \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} p_{x,y}(\alpha, \beta) \log \frac{p_{x,y}(\alpha, \beta)}{p_x(\alpha)p_y(\beta)} d\alpha d\beta. \quad (6.4)$$

The entropy of a continuous random variable is infinite since it requires an infinite number of binary digits to represent it exactly. Instead, a quantity known as the differential entropy of the continuous random variable X has been defined as [48]:

$$H(X) \triangleq -\int_{-\infty}^{+\infty} p_x(\alpha) \log p_x(\alpha) d\alpha. \quad (6.5)$$

The conditional entropy of continuous random variable X given continuous random variable Y is then defined as:

$$H(X|Y) \triangleq \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} p_{x,y}(\alpha, \beta) \log p_{x|y}(\alpha|\beta) d\alpha d\beta. \quad (6.6)$$

Consequently, the mutual information between two continuous random variables may be expressed as $I(X;Y) \triangleq H(X) - H(X|Y)$.

If discrete information symbols are transmitted over a noisy channel we are dealing with one continuous random variable, the channel, and one discrete, the source. Denote the entropy of the output of the channel, i.e. the received signal, $H(Y)$ and the entropy of the input to the channel, the source, $H(X)$. If the channel had been noiseless $H(X) = H(Y)$. The mutual information provided about X upon reception of Y is:

$$I(X;Y) = \sum_{i=1}^n \int_{-\infty}^{+\infty} p_y(x_i, \beta) \log \frac{p_y(x_i, \beta)}{P(x_i) p_y(\beta)} d\beta. \quad (6.7)$$

Since $p_y(x_i, \beta) = p_y(\beta|x_i)P(x_i)$ equation (6.7) can also be written

$$I(X;Y) = \sum_{i=1}^n \int_{-\infty}^{+\infty} p_y(\beta|x_i)P(x_i) \log \frac{p_y(\beta|x_i)}{p_y(\beta)} d\beta. \quad (6.8)$$

Note that when X and Y are statistically independent, $p_y(\beta|x_i) = p_y(\beta)$ and hence $I(X;Y) = 0$.

6.2 Channel Capacity

Consider a memoryless AWGN channel, W , with random input X so that

$$Y = X + W. \quad (6.9)$$

W is a zero-mean Gaussian random variable with variance σ_w^2 . The channel capacity is the maximum average mutual information (MI) between the received signal, $Y = X + W$, and the transmitted symbol, X , [10]:

$$C \triangleq \max_{p_x(\alpha)} \{I(X;Y)\}. \quad (6.10)$$

The capacity is given in bits per channel use when the logarithmic base is two, so that the unit of the entropy is bits. Note that regardless of encoding or decoding method used, the channel capacity can never be exceeded, i.e., communication with an arbitrary low BER can never be obtained using a transmission rate higher than the capacity.

For the AWGN channel the transmitted signal and the noise are independent of each other and the received signal is the sum of these independent random variables. Then [10]

$$C \triangleq \max_{p_x(\alpha)} \{H(Y)\} - H(W) \quad (6.11)$$

since $H(X) - H(X|Y) = H(Y) - H(Y|X)$. As $H(W)$ is independent of $p_x(\alpha)$, the rate is maximized by maximizing $H(Y)$. When the transmitted signals are limited to a certain average power, the maximum of $H(Y)$ over the input $p_x(\alpha)$ is obtained when X is a zero-mean Gaussian random variable [10], so that

$$p_y(\beta) = \frac{1}{\sqrt{2\pi\sigma_y^2}} e^{-\frac{\beta^2}{2\sigma_y^2}}; \quad \text{where } \sigma_y^2 = (\sigma_x^2 + \sigma_w^2). \quad (6.12)$$

Since

$$-\log_2 p_y(\beta) = \log_2 \sqrt{2\pi\sigma_y^2} + \frac{\beta^2}{2\sigma_y^2} \frac{1}{\ln 2}, \quad (6.13)$$

we have

$$H(Y) = \int_{-\infty}^{+\infty} p_y(\beta) \log_2 \sqrt{2\pi\sigma_y^2} d\beta + \int_{-\infty}^{+\infty} p_y(\beta) \frac{\beta^2}{2\sigma_y^2} \frac{1}{\ln 2} d\beta = \log_2 \sqrt{2\pi\sigma_y^2} + \frac{\sigma_y^2}{2\sigma_y^2} \frac{1}{\ln 2}. \quad (6.14)$$

Finally

$$H(Y) = \log_2 \sqrt{2\pi\sigma_y^2} + \frac{\log_2 e}{2 \log_2 2} = \log_2 \sqrt{2\pi e \sigma_y^2} = \frac{1}{2} \log_2 (2\pi e (\sigma_x^2 + \sigma_w^2)). \quad (6.15)$$

Analogously

$$H(W) = \frac{1}{2} \log_2 (2\pi\sigma_w^2). \quad (6.16)$$

The channel capacity is then [10]

$$C = \frac{1}{2} \log_2(2\pi e(\sigma_x^2 + \sigma_w^2)) - \frac{1}{2} \log_2(2\pi e\sigma_w^2) = \frac{1}{2} \log_2\left(1 + \frac{\sigma_x^2}{\sigma_w^2}\right), \quad (6.17)$$

with $\sigma_w^2 = N_0/2$ and $\sigma_x^2 = E_s$.

6.2.1 Capacity for the Binary Input AWGN Channel

Consider an AWGN channel, W , with binary inputs $X \in \{+\mu, -\mu\}$. The average MI between the received signal $Y = X + W$, and the transmitted signal X is maximized when the input probabilities are equally likely, i.e., $P(X = +\mu) = P(X = -\mu) = \frac{1}{2}$, [48], yielding

$$C^* = \frac{1}{2} \sum_{x_i = \pm\mu} \int_{-\infty}^{+\infty} p_y(\beta | x_i) \log_2 \frac{p_y(\beta | x_i)}{p_y(\beta)} d\beta, \quad (6.18)$$

where

$$p_y(\beta) = \sum_{x_k = \pm\mu} p_y(x_k, \beta) = \sum_{x_k = \pm\mu} p_y(\beta | x_k) P(x_k). \quad (6.19)$$

and

$$p_y(\beta | x_i) = p_w(\beta - x_i) = \frac{1}{\sqrt{2\pi\sigma_w^2}} e^{-\frac{(\beta - x_i)^2}{2\sigma_w^2}}. \quad (6.20)$$

Using (6.19) and (6.20) in (6.18) yields

$$C^* = \frac{1}{2} \sum_{x_i = \pm\mu} \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi\sigma_w^2}} e^{-\frac{(\beta - x_i)^2}{2\sigma_w^2}} \log_2 \frac{1}{\frac{1}{\sqrt{2\pi\sigma_w^2}} e^{-\frac{(\beta - x_i)^2}{2\sigma_w^2}} \sum_{x_k = \pm\mu} \frac{1}{\sqrt{2\pi\sigma_w^2}} e^{-\frac{(\beta - x_k)^2}{2\sigma_w^2}}} d\beta, \quad (6.21)$$

and further

$$\begin{aligned} C^* &= \frac{1}{2} \sum_{x_i = \pm\mu} \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi\sigma_w^2}} e^{-\frac{(\beta - x_i)^2}{2\sigma_w^2}} \left(1 - \log_2 \left(1 + e^{-\frac{(\beta + x_i)^2}{2\sigma_w^2} - \frac{(\beta - x_i)^2}{2\sigma_w^2}} \right) \right) d\beta \\ &= 1 - \frac{1}{2} \sum_{x_i = \pm\mu} \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi\sigma_w^2}} e^{-\frac{(\beta - x_i)^2}{2\sigma_w^2}} \log_2 \left(1 + e^{-\frac{2\beta x_i}{\sigma_w^2}} \right) d\beta. \end{aligned} \quad (6.22)$$

By using symmetry

$$C^* = 1 - \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi\sigma_w^2}} e^{-\frac{(\beta-\mu)^2}{2\sigma_w^2}} \log_2 \left(1 + e^{-\frac{2\beta\mu}{\sigma_w^2}} \right) d\beta. \quad (6.23)$$

Substituting $\alpha = 2\beta\mu / \sigma_w^2$ we get

$$C^* = 1 - \int_{-\infty}^{+\infty} \frac{\sigma_w}{\sqrt{8\pi\mu^2}} e^{-\frac{\left(\frac{\alpha^2\sigma_w^4 + \mu^2 - \alpha\sigma_w^2}{4\mu^2}\right)}{2\sigma_w^2}} \log_2(1 + e^{-\alpha}) d\alpha. \quad (6.24)$$

and finally by defining $\sigma = 2\mu / \sigma_w$, we can express C^* as:

$$C^*(\sigma) = 1 - \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\left(\frac{\alpha - \sigma^2}{2}\right)^2}{2\sigma^2}} \log_2(1 + e^{-\alpha}) d\alpha. \quad (6.25)$$

In a BPSK system $\mu = \sqrt{E_s}$ and $\sigma_w^2 = N_0/2$, which yields $\sigma = \sqrt{8E_s/N_0}$. There is no closed form solution for C^* and hence it has to be evaluated numerically.

C and C^* are plotted in Figure 6.1 as a function of E_s/N_0 . Note that according to [10] reliable communication can be achieved with arbitrary low error rate as long as the code rate is kept below C . We can also see that constraining the channel input to discrete binary values yields a C^* that has a ceiling at one bit per symbol. Hence, if we want to transmit k information bits using BPSK, we need to use a codeword of at least k bits so that the code rate does not exceed one.

6.3 Extrinsic Information Transfer Characteristics

When making an EXIT chart analysis, we want to predict the behavior of the iterative decoder by looking solely at the input/output relations of its component decoders. Hence, we want to isolate each component decoder and look at its input and output relations separately, as illustrated in Figure 6.2. Since the component decoders are no longer connected to the channel we instead need to model the channel output. Observations in [29] suggest that the *a priori* inputs to a component decoder, $A(\mathbf{x})$ and $A(\mathbf{y}_l)$, can be modeled by adding an independent Gaussian random variable to the known transmitted sequences, \mathbf{x} and \mathbf{y}_l . Having done this, the MI between the known transmitted sequence, \mathbf{x} , and the *a priori* input, $A(\mathbf{x})$, can be evaluated for each component decoder. Further, the MI between the known transmitted information and the extrinsic outputs may be obtained by activating the component decoder with the modeled inputs on $A(\mathbf{x})$ and $A(\mathbf{y}_l)$. This procedure is explained in detail in this section.

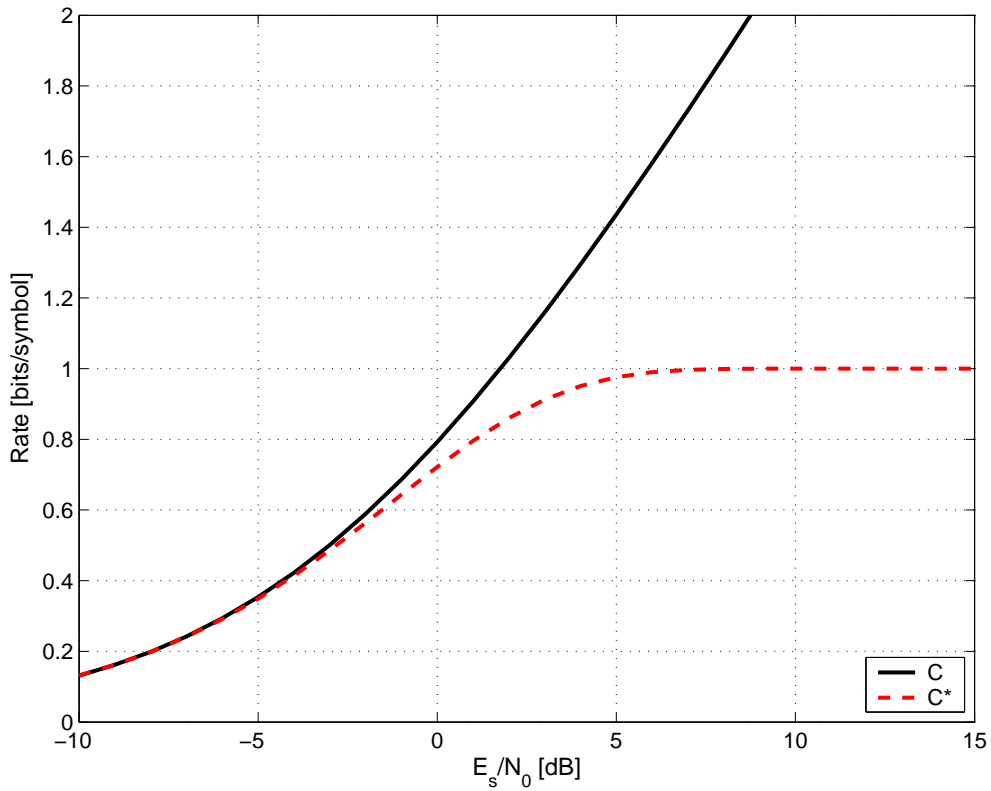


Figure 6.1. Capacity for the AWGN channel, C , together with the maximal rate for BPSK signaling on the AWGN channel, C^* .

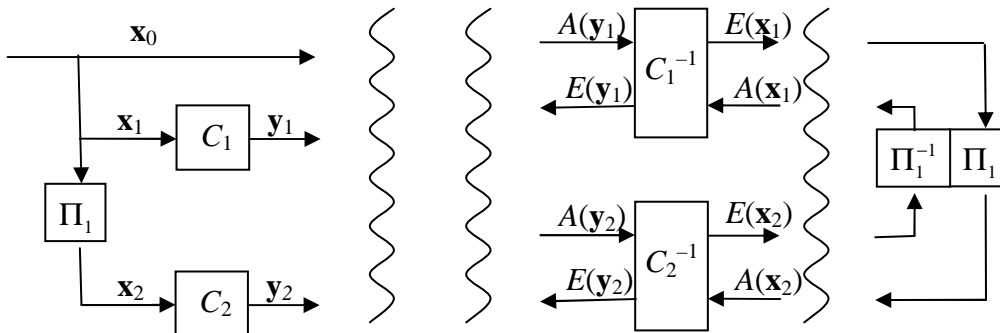


Figure 6.2. Each component decoder is extracted from the iterative decoder. Instead $A(\cdot)$ are modeled by adding an independent Gaussian random variable to the known transmitted information.

Let y again be defined as a uniformly distributed binary random variable that is to be transmitted over the AWGN channel according to the system model, so that $r = s + w = \mu\bar{y} + w$. Recall that the LLR of y conditioned on the matched filter output r is

$$L(y|r) = \ln\left(\frac{e^{-(r-\mu)^2/2\sigma_w^2}}{e^{-(r+\mu)^2/2\sigma_w^2}}\right) + \ln\left(\frac{P(\bar{y}=+1)}{P(\bar{y}=-1)}\right) = L_c r + L(y), \quad (6.26)$$

where $L_c = 2\mu/\sigma_w^2$. Since $L(y) = 0$ before decoding starts, $L_c r$ is what is fed into a decoder working in the LLR domain. $L_c r$ can also be formulated as

$$L_c r = \frac{2\mu}{\sigma_w^2} r = \frac{2\mu}{\sigma_w^2} (\mu\bar{y} + w) = \frac{2\mu^2}{\sigma_w^2} \bar{y} + n \quad (6.27)$$

where $n = 2\mu w/\sigma_w^2$. Note that

$$\sigma_n^2 = \frac{4\mu^2}{\sigma_w^4} \sigma_w^2 = \frac{4E_s}{\sigma_w^2} = \frac{8E_s}{N_0} \triangleq \sigma^2 \quad (6.28)$$

so that

$$L_c r = \frac{2\mu^2}{\sigma_w^2} \bar{y} + n = \frac{\sigma^2}{2} \bar{y} + n, \quad (6.29)$$

where $n \in \mathcal{N}(0, \sigma^2)$. Also note that the σ defined here is the same as was used for $C^*(\sigma)$ in equation (6.25).

We now modeled the prior information on x , $A(x)$, by applying $n_A \in \mathcal{N}(0, \sigma_A^2)$ similar to (6.29) so that

$$A(x) = \frac{\sigma_A^2}{2} \bar{x} + n_A, \quad \sigma_A^2 = \frac{8E_s}{N_0}. \quad (6.30)$$

Then

$$p_A(\alpha|\bar{x}) = p_{n_A}(\alpha - \bar{x}) = \frac{1}{\sqrt{2\pi\sigma_A^2}} e^{-\left(\alpha - \frac{\sigma_A^2}{2}\bar{x}\right)^2/2\sigma_A^2}. \quad (6.31)$$

The mutual information between the transmitted symbols x and the prior information of x at the receiver, $A(x)$, can then be expressed as

$$I_A \triangleq I(x; A(x)) = \frac{1}{2} \sum_{x=\pm 1} \int_{-\infty}^{+\infty} p_A(\alpha|x) \log_2 \left(\frac{2p_A(\alpha|x)}{p_A(\alpha|x) + p_A(\alpha|-x)} \right) d\alpha. \quad (6.32)$$

Using (6.31) in (6.32) we have

$$I_A(\sigma_A) = -\frac{1}{2} \int_{-\infty}^{+\infty} \frac{e^{-\left(\alpha - \frac{\sigma_A^2}{2}\right)^2 / 2\sigma_A^2}}{\sqrt{2\pi\sigma_A^2}} \log_2(2 + 2e^{-\alpha}) d\alpha - \frac{1}{2} \int_{-\infty}^{+\infty} \frac{e^{-\left(\alpha + \frac{\sigma_A^2}{2}\right)^2 / 2\sigma_A^2}}{\sqrt{2\pi\sigma_A^2}} \log_2(2e^\alpha + 2) d\alpha, \quad (6.33)$$

which can be further simplified to

$$I_A(\sigma_A) = \int_{-\infty}^{+\infty} \frac{e^{-\left(\alpha - \frac{\sigma_A^2}{2}\right)^2 / 2\sigma_A^2}}{\sqrt{2\pi\sigma_A^2}} (1 - \log_2(1 + e^{-\alpha})) d\alpha. \quad (6.34)$$

To maintain a notation similar to [29] we define

$$J(\sigma) \triangleq I_A(\sigma_A = \sigma), \quad (6.35)$$

with

$$\lim_{\sigma \rightarrow 0} J(\sigma) = 0, \quad \lim_{\sigma \rightarrow \infty} J(\sigma) = 1 \quad \text{and} \quad \sigma > 0. \quad (6.36)$$

Note that the function $J(\sigma)$ is identical to (6.25). $J(\sigma)$ cannot be expressed in closed form, but from numerical integration it is found that $J(\sigma)$ is monotonically increasing and hence reversible [29]

$$\sigma = J^{-1}(I_A). \quad (6.37)$$

$J(\sigma)$ and its inverse can be closely approximated by [30]:

$$J(\sigma) \approx \left(1 - 2^{-H_1 \sigma^{2H_2}}\right)^{H_3}, \quad (6.38)$$

$$J^{-1}(I) \approx \left(-\frac{1}{H_1} \log_2 \left(1 - I^{\frac{1}{H_3}}\right)\right)^{\frac{1}{2H_2}}, \quad (6.39)$$

where numerical optimization, minimizing the total squared difference between (6.34) and (6.38), yields the parameter values $H_1=0.3073$, $H_2=0.8935$ and $H_3=1.1064$ [30]. In Figure 6.3 $J(\sigma)$ and its approximation are plotted.

6.3.1 Extrinsic Information Transfer Functions

Recall that we isolate each component decoder and look at its input and output relations

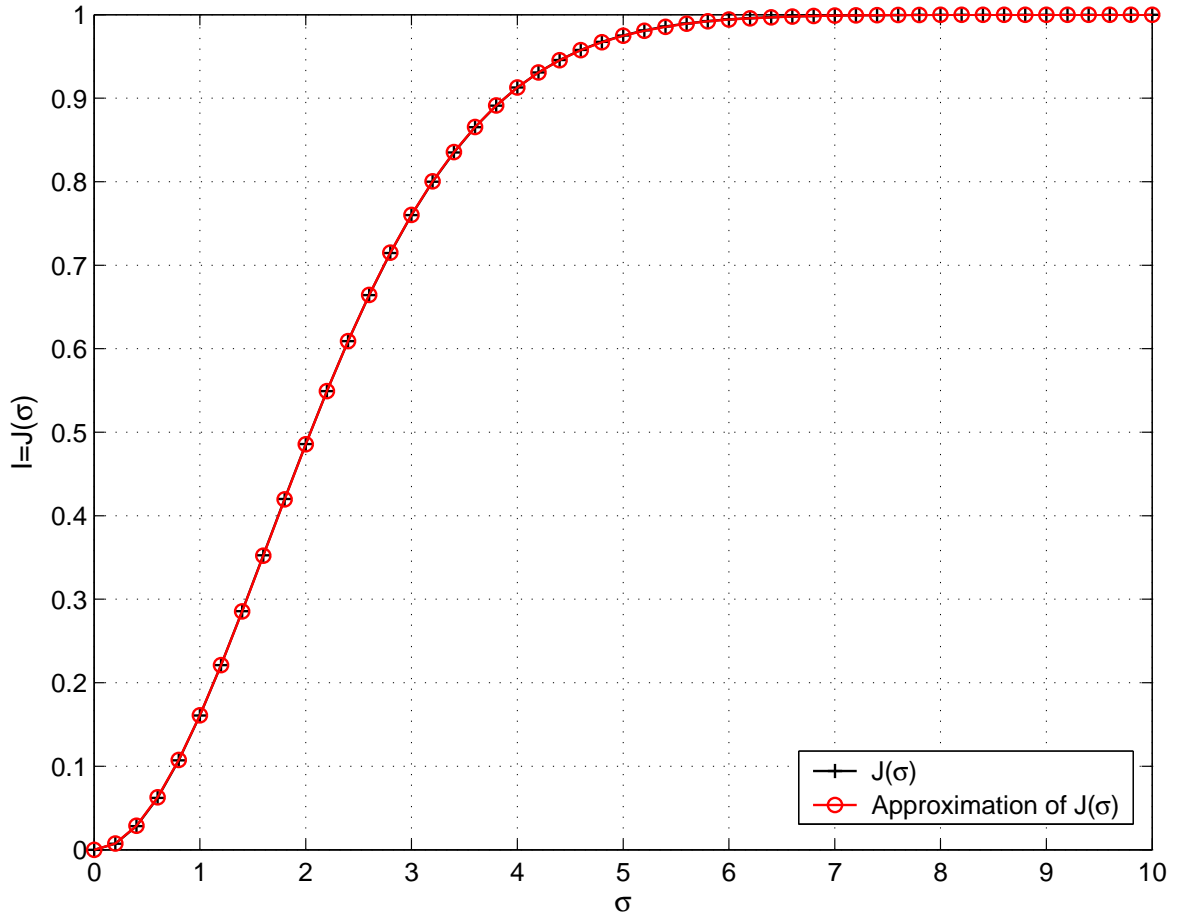


Figure 6.3. $I = J(\sigma)$ and its approximation.

separately. Since the component decoders are no longer connected to the channel we instead model the channel output using a Gaussian random variable that is added to the known transmitted information. Since $\bar{\mathbf{x}} \in \{+1, -1\}^k$ and $\bar{\mathbf{y}}_l \in \{+1, -1\}^{n_{C_l} - k_{C_l}}$, where C_l denotes the specific component code, then

$$A(\mathbf{x}) = \frac{4E_s}{N_0} \bar{\mathbf{x}} + \mathbf{n}_x = \frac{4r_C E_b}{N_0} \bar{\mathbf{x}} + \mathbf{n}_x = \frac{\sigma_x^2}{2} \bar{\mathbf{x}} + \mathbf{n}_x, \quad \mathbf{n}_x(i) \in \mathcal{N}(0, \sigma_x^2) \quad \forall i, \quad (6.40)$$

and

$$A(\mathbf{y}_l) = \frac{4E_s}{N_0} \bar{\mathbf{y}}_l + \mathbf{n}_{y_l} = \frac{4r_C E_b}{N_0} \bar{\mathbf{y}}_l + \mathbf{n}_{y_l} = \frac{\sigma_{y_l}^2}{2} \bar{\mathbf{y}}_l + \mathbf{n}_{y_l}, \quad \mathbf{n}_{y_l}(i) \in \mathcal{N}(0, \sigma_{y_l}^2) \quad \forall i. \quad (6.41)$$

Having done this, the MI between the transmitted information, $\mathbf{x}(i)$, and the *a priori* input, $A(\mathbf{x}(i))$, denoted $I(\mathbf{x}(i); A(\mathbf{x}(i)))$, can be evaluated by means of the $J(\sigma)$ function since

$A(\mathbf{x}(i))$ is Gaussian. The average MI is defined as

$$I(x; A(x)) \triangleq \frac{1}{k} \sum_{i=1}^k I(\mathbf{x}(i); A(\mathbf{x}(i))). \quad (6.42)$$

Then the average MI is given as [29]:

$$I_{A(x)} \triangleq I(x; A(x)) = J(\sigma_x) = J\left(\sqrt{\frac{8r_c E_b}{N_0}}\right), \quad (6.43)$$

and similar for \mathbf{y}_l

$$I_{A(y_l)} \triangleq I(y_l; A(y_l)) = J(\sigma_{y_l}) = J\left(\sqrt{\frac{8r_c E_b}{N_0}}\right). \quad (6.44)$$

The extrinsic outputs on \mathbf{x} and \mathbf{y}_l , denoted $E(\mathbf{x})$ and $E(\mathbf{y}_l)$ can also be quantified using average MI [29]

$$I_{E(x)} \triangleq I(x; E(x)) = \frac{1}{2} \sum_{x=\pm 1} \int_{-\infty}^{+\infty} p_E(\xi | x) \log_2 \left(\frac{2p_E(\xi | x)}{p_E(\xi | x) + p_E(\xi | -x)} \right) d\xi. \quad (6.45)$$

However, this assumes that $p_E(\xi | x)$ and $p_E(\xi | y_l)$ are known. These conditional PDFs depend on the code and can usually not be found analytically. Monte Carlo simulations are therefore applied to determine these probabilities [29]. To do this we first recognize that $I_{E(x)}$ and $I_{E(y_l)}$ are functions of the inputs $I_{A(x)}$ and $I_{A(y_l)}$ according to

$$I_{E(x)} = f_x(I_{A(x)}, I_{A(y_l)}), \quad (6.46)$$

$$I_{E(y_l)} = f_{y_l}(I_{A(x)}, I_{A(y_l)}). \quad (6.47)$$

Then, the independent Gaussian random variables are applied to the inputs to generate all values of $I_{A(x)}$ and $I_{A(y_l)}$. Next, the decoder is activated using the inputs $A(\mathbf{x})$ and $A(\mathbf{y}_l)$ so that $E(\mathbf{x})$ and $E(\mathbf{y}_l)$ are obtained. Finally, histogram approximations are made on $E(\mathbf{x})$ and $E(\mathbf{y}_l)$ so that $p_E(\xi | \pm x)$ and $p_E(\xi | \pm y_l)$ can be found [29]. In [30] a detailed algorithm describing how to generate $p_E(\xi | \pm x)$ and $p_E(\xi | \pm y_l)$ by Monte Carlo simulations can be found. These PDFs are then inserted in (6.45) to give the MI on the extrinsic outputs.

Having obtained the average MI on the extrinsic outputs, we can plot $I_{E(x)}$ and $I_{E(y_l)}$ as functions of $I_{A(x)}$ and $I_{A(y_l)}$ [30]. In Figure 6.4, these so-called EXIT functions are plotted for an SPC(8,7) code when seen as a rate-7 code, i.e., we input seven information bits and get one parity bit out, so that $\bar{\mathbf{y}}_l \in \{+1, -1\}^{n_{c_l} - k_{c_l}}$. The EXIT functions for the SPC(8,7) code when seen as a rate-7/8 code, i.e., seven information bits in give eight coded bits out, $\bar{\mathbf{y}}_l \in \{+1, -1\}^{n_{c_l}}$, are depicted in Figure 6.5. Recall that a 2D parallel concatenated code is

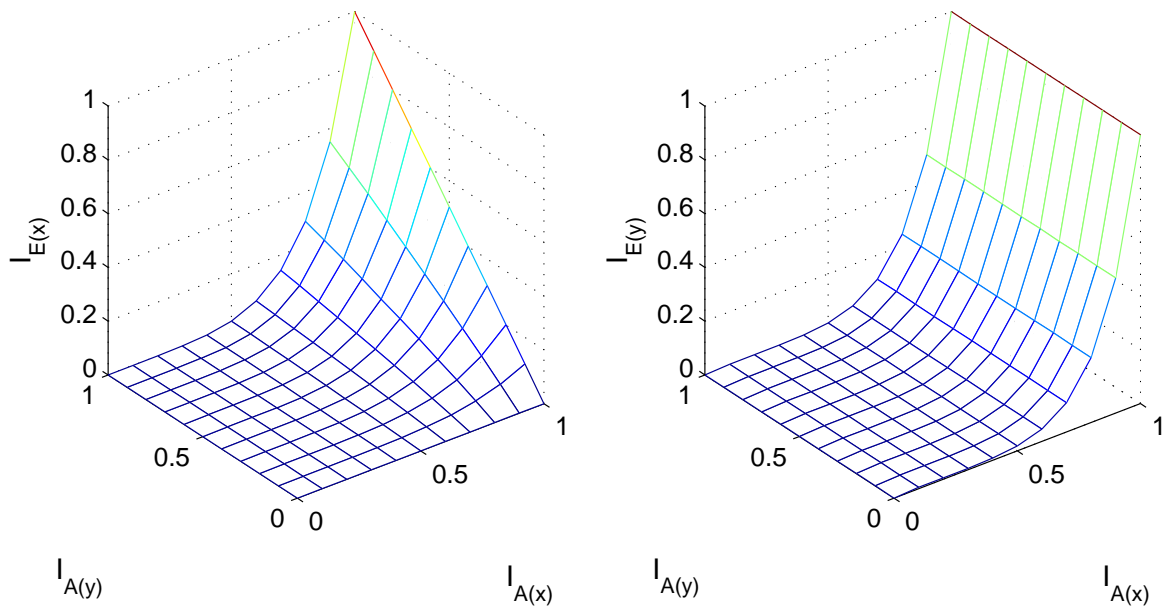


Figure 6.4. EXIT functions for a SPC(8,7) code viewed as a rate-7 code.

constructed either using one rate-1 systematic part and two rate-7 SPC(8,7) codes or using one rate-7 code and one rate-7/8 code. A 2D serial concatenated code can be constructed in the same two ways, but also using two rate-7/8 codes.

Note that the parallel concatenated code uses only $E(x)$ and never $E(y)$, since the input bits, x , are connecting the two component decoders. The serial concatenated code uses both

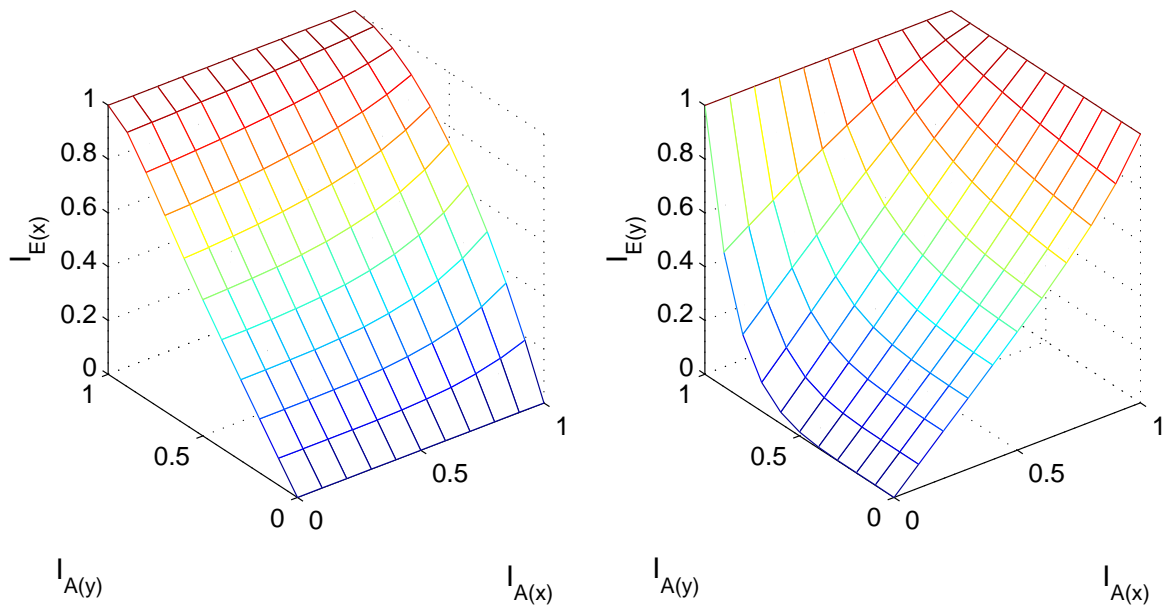


Figure 6.5. EXIT functions for a SPC(8,7) code viewed as a rate-7/8 code.

$E(x)$ and $E(y_l)$, since the output of one component code, y_l , is used as input to the next component code. In Figure 6.4 and Figure 6.5 it can be seen in that $f_x(0,0) = f_{y_l}(0,0) = 0$ and $f_x(1,1) = f_{y_l}(1,1) = 1$. This implies that with no prior information, the average MI on the extrinsic outputs is zero, whereas with full prior information, the average MI on the extrinsics is one. In [30] EXIT functions for feed-forward and feed-backward convolutional codes are plotted. From these plots, it can be seen that recursive codes have $I_{E(x)} = f_x(1, I_{A(y_l)}) = 1$. As can be seen in Figure 6.4 and Figure 6.5, this is not the case for the SPC(8,7) code, and hence it does not have a recursive structure.

In Appendix A, the EXIT functions for a ZZ code are given and it is found to have the recursive structure lacking from the SPC codes.

6.3.2 Extrinsic Information Transfer Charts

For a 2D parallel concatenated system, extrinsic information on the information bits, \mathbf{x} , are exchanged between the two component decoders. Consequently, we are interested in looking at how $I_{E(x_1)}$ and $I_{E(x_2)}$ evolve when the component decoders are activated. Recall from the system model of a parallel concatenated decoder in Figure 2.5 that $A(\mathbf{y}_l) = C(\mathbf{y}_l)$. Hence, (6.44) yields

$$I_{A(y_l)} = I_{C(y_l)} = J \left(\sqrt{\frac{8r_{C_{2D}^{\parallel}} E_b}{N_0}} \right), \quad (6.48)$$

as the average MI on the prior information on \mathbf{y}_l that is received from the channel for each decoder, $l = 1, 2$. As prior information on \mathbf{x} for C_1^{-1} we have both the information on \mathbf{x} that is received from the channel, $A(\mathbf{x}_0) = C(\mathbf{x}_0)$ and the extrinsic information on \mathbf{x} received from the other component decoder, $E(\mathbf{x}_2)$. Thus

$$I_{A(x_1)} = J \left(\sqrt{J^{-1} (I_{A(x_0)})^2 + J^{-1} (I_{E(x_2)})^2} \right), \quad (6.49)$$

since the addition is made in the variance domain [29]. Because $I_{A(x_0)} = I_{C(x_0)}$ comes directly from the channel it is equal to $J(\sqrt{8r_{C_{2D}^{\parallel}} E_b / N_0})$. Consequently,

$$I_{A(x_1)} = J \left(\sqrt{\frac{8r_{C_{2D}^{\parallel}} E_b}{N_0} + J^{-1} (I_{E(x_2)})^2} \right), \quad (6.50)$$

$$I_{A(x_2)} = J \left(\sqrt{\frac{8r_{C_{2D}^{\parallel}} E_b}{N_0} + J^{-1} (I_{E(x_1)})^2} \right). \quad (6.51)$$

Finally,

$$I_{E(x_1)} = f_{x_1}(I_{A(x_1)}, I_{A(y_1)}) = f_{x_1}\left(J\left(\sqrt{\frac{8r_{C_{2D}} E_b}{N_0} + J^{-1}(I_{E(x_2)})^2}\right), J\left(\sqrt{\frac{8r_{C_{2D}} E_b}{N_0}}\right)\right), \quad (6.52)$$

$$I_{E(x_2)} = f_{x_2}(I_{A(x_2)}, I_{A(y_2)}) = f_{x_2}\left(J\left(\sqrt{\frac{8r_{C_{2D}} E_b}{N_0} + J^{-1}(I_{E(x_1)})^2}\right), J\left(\sqrt{\frac{8r_{C_{2D}} E_b}{N_0}}\right)\right). \quad (6.53)$$

The above EXIT functions depends both on each other and on E_b/N_0 . However, for a fixed E_b/N_0 we can plot them as functions of each other [29]. In Figure 6.6 this is done for a two-dimensional parallel concatenated SPC(8,7) code at $E_b/N_0 = 3$ dB. Note that the dashed curve indicating $I_{E(x_1)}$ is plotted against the abscissa; whereas the solid curve indicating $I_{E(x_2)}$ has its axes swapped and is thus plotted against the ordinate. Also, note that the two curves in Figure 6.6 are symmetric. This is due to the fact that we have identical component codes and that the EXIT chart is using identical EXIT functions, i.e., two rate-7 SPC(8,7) codes.

Several insights can be gained from the EXIT chart. The exchange of extrinsic information between the component decoders can be visualized as a decoding trajectory in the EXIT chart. Decoding starts with no prior information at the origin in the EXIT chart. If decoder C_1^{-1} is activated we move vertically up to $I_{E(x_1)} = 0.36$. This extrinsic information is then used as *a priori* information to the second decoder. Hence, if C_2^{-1} is activated next, we move horizontally to $I_{E(x_2)} = 0.46$. When activating C_1^{-1} a second time we move vertically again and so on. If a staircase shaped trajectory bounded by the decoder transfer characteristic is drawn into the EXIT chart, the number of activations can be visualized [29].

The two curves in Figure 6.6 intersect at $I_{E(x_1)} = I_{E(x_2)} = 0.58$. Consequently, the iterative decoder gets stuck here and further activations do not result in further improvements. Ideally, we would like a tunnel between the two decoder transfer characteristics so that the decoding trajectory can “sneak through” all the way up to the upper right corner [29]. This will result in a low BER. In [29] three different scenarios are discussed. The first one, depicted in Figure 6.6, is referred to as the pinch-off region since there is no tunnel and the decoder transfer characteristics intersect at a low average MI resulting in a high BER. This corresponds to the region above the waterfall region in turbo decoding. The second scenario is when a small tunnel is just opening up so that convergence to a low BER is slow but possible. This corresponds to the waterfall region. Finally in the third case, the BER floor region corresponds to a big tunnel yielding fast convergence to a low BER. We will see examples of the two later scenarios below, but since PCSPC codes do not have any real waterfall as confirmed by the simulations, the two remaining scenarios cannot be exemplified with this system.

After a number of activations the iterative decoder converges with the extrinsic MI obtained where the two decoder transfer characteristics first intersect. Based on this, the EXIT chart can be used to obtain the approximate BER after the convergence point has been reached [29]. The BER can also be approximated after each iteration, but since we are

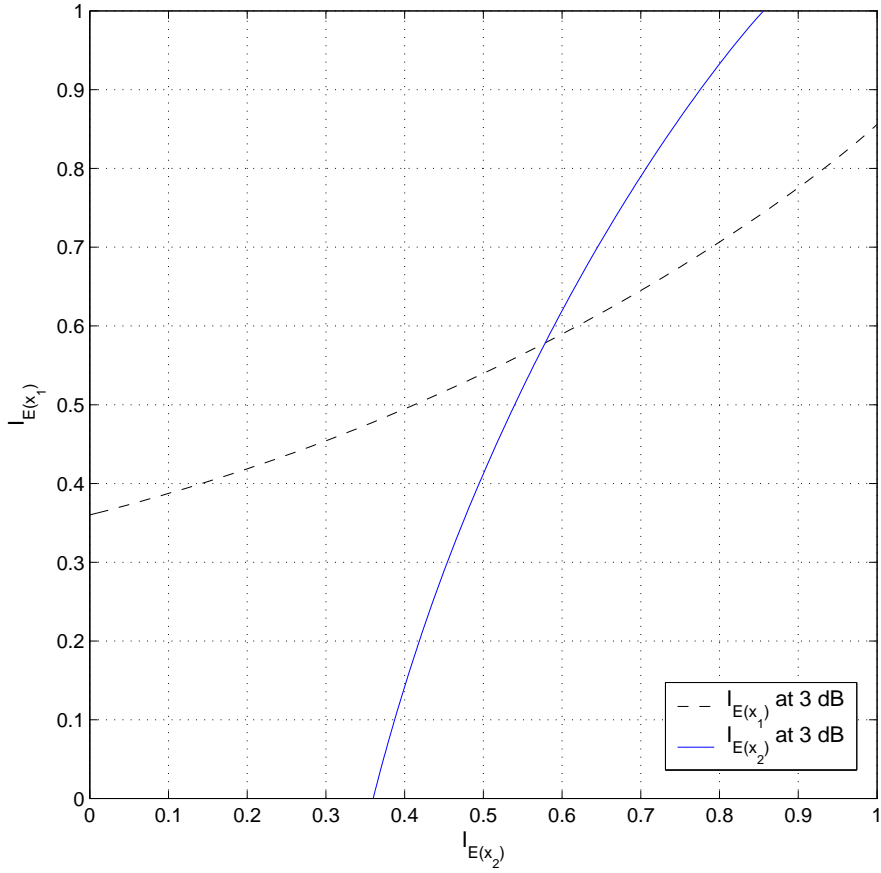


Figure 6.6. EXIT chart for a 2D parallel concatenated SPC(8,7) code at 3 dB.

interested in the convergence behavior, the approximation is made after convergence is reached. Recall that $D(\mathbf{x})$, the decision statistics for \mathbf{x} , for a 2D PCSPC code can be expressed as

$$D(\mathbf{x}) = C(\mathbf{x}_0) + E(\mathbf{x}_1) + \Pi_1^{-1}(E(\mathbf{x}_2)), \quad (6.54)$$

representing the intrinsic, the extrinsic and the *a priori* information on \mathbf{x} respectively. Note that \mathbf{x}_2 is simply an interleaved version of \mathbf{x}_0 and that $\mathbf{x}_0 = \mathbf{x}_1$. Denote the average MI on the decision statistics for \mathbf{x} as $I_{D(\mathbf{x})}$ so that

$$I_{D(\mathbf{x})} = J \left(\sqrt{J^{-1}(I_{A(\mathbf{x}_0)})^2 + J^{-1}(I_{E(\mathbf{x}_1)})^2 + J^{-1}(I_{E(\mathbf{x}_2)})^2} \right). \quad (6.55)$$

Hence, the values of $I_{E(\mathbf{x}_1)}$ and $I_{E(\mathbf{x}_2)}$ after the convergence point has been reached, i.e., where the two curves intersect, here $I_{E(\mathbf{x}_1)} = I_{E(\mathbf{x}_2)} = 0.58$, are used in (6.55) to obtain the average MI on $D(\mathbf{x})$. When using EXIT charts we assume that $A(\mathbf{x}_0)$, $E(\mathbf{x}_1)$ and $E(\mathbf{x}_2)$ are Gaussian distributed. Hence, $D(\mathbf{x})$ is also Gaussian distributed with variance $\sigma_{D(\mathbf{x})}^2$ and

mean $\mu_{D(x)} = \sigma_{D(x)}^2 / 2$. Further, we assume independence so that

$$\sigma_{D(x)}^2 = \sigma_{A(x_0)}^2 + \sigma_{E(x_1)}^2 + \sigma_{E(x_2)}^2. \quad (6.56)$$

For information transmitted over the AWGN channel using BPSK we have [48]

$$P_b = Q\left(\frac{\mu}{\sigma}\right). \quad (6.57)$$

Then [29]

$$\begin{aligned} P_b &\approx Q\left(\frac{\sigma_{D(x)}}{2}\right) = Q\left(\frac{J^{-1}(I_{D(x)})}{2}\right) = \\ &= Q\left(\frac{\sqrt{\sigma_{A(x_0)}^2 + \sigma_{E(x_1)}^2 + \sigma_{E(x_2)}^2}}{2}\right) = Q\left(\frac{\sqrt{\frac{8r_{\text{C2D}} E_b}{N_0} + J^{-1}(I_{E(x_1)})^2 + J^{-1}(I_{E(x_2)})^2}}{2}\right), \end{aligned} \quad (6.58)$$

since σ can be found through equation (6.37).

In Figure 6.7 we have plotted the EXIT chart for the same two-dimensional parallel concatenated SPC(8,7) code, but for a range of E_b/N_0 values. From these curves we can get $I_{E(x_1)}$ and $I_{E(x_2)}$ where the decoder transfer characteristics intersect for the first time and hence find the BER for different values of E_b/N_0 . The BER obtained this way is based on the assumptions made for EXIT charts, namely that the extrinsic information has a Gaussian distribution and that the interleavers used are infinitely large. Since there is no real tunnel opening up in Figure 6.7, this code does not have any real waterfall region, but instead a rather slow drop in BER as E_b/N_0 increases. This is confirmed both by simulations of the code and by the BER that can be obtained using the EXIT chart. In Figure 6.8 we have plotted the BER curve obtained from the EXIT chart of a 2D parallel concatenated SPC(8,7) code and compared it to simulations of the same code using two different sizes of the interleaver. As can be seen in Figure 6.8 the BER from EXIT charts are quite accurate in this example, especially when the interleaver is sufficiently large.

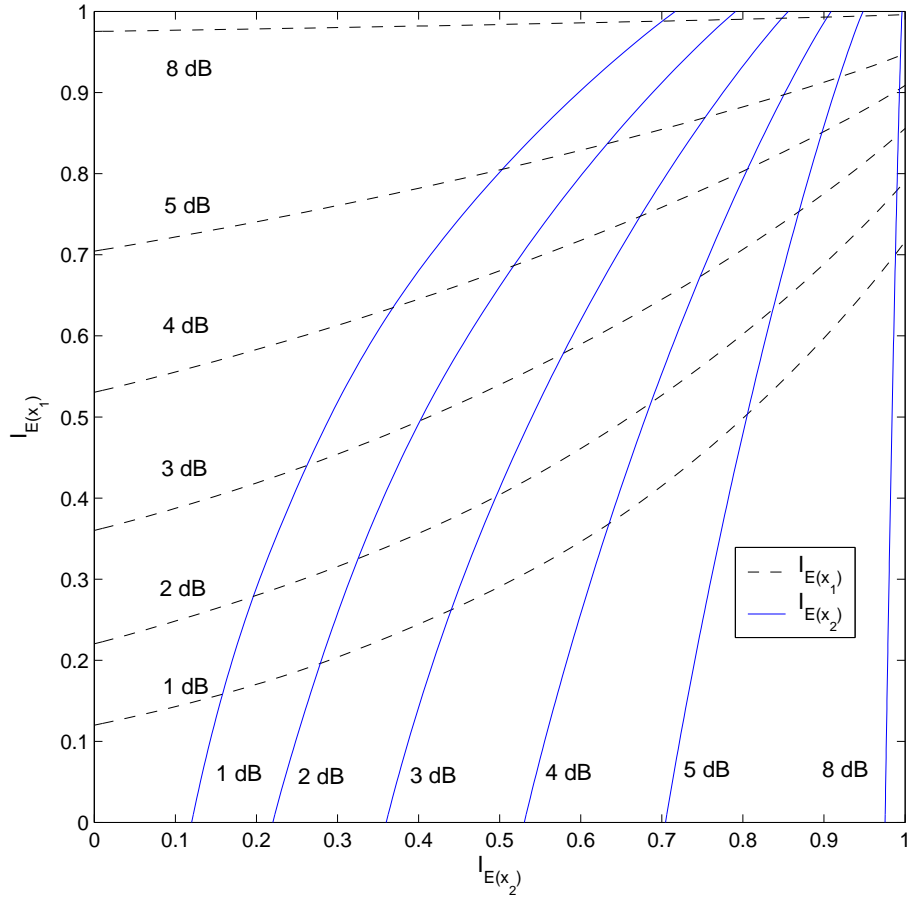


Figure 6.7. EXIT chart for a 2D parallel concatenated SPC(8,7) code.

For a 2D serially concatenated system, extrinsic information on both the information bits, \mathbf{x} , and the parity bits \mathbf{y}_1 are exchanged between the two component decoders. Recall that $\mathbf{y} = [\mathbf{x}_0, \mathbf{y}_1, \mathbf{y}_2]$, and we are interested in looking at $[\mathbf{x}_0, \mathbf{y}_1]$ from decoder C_1^{-1} and $\mathbf{x}_2 = [\mathbf{x}_0, \mathbf{y}_1]$ from decoder C_2^{-1} . For simplicity, we will assume that we have the encoder depicted in Figure 2.7 in the system model when deriving the expressions for an EXIT chart for a 2D SPC code. This implies that we instead have $\mathbf{y} = \mathbf{y}_2$ and we are interested in looking at \mathbf{y}_1 from decoder C_1^{-1} and \mathbf{x}_2 from decoder C_2^{-1} and how $I_{E(y_1)}$ and $I_{E(x_2)}$ evolve when the component decoders are activated. The notation in the following paragraph follows that of the system model in Figure 2.7. Since C_2^{-1} is connected to the channel, $I_{A(y_2)} = I_{C(y)}$ we have

$$I_{A(y_2)} = J \left(\sqrt{\frac{8r_{\perp} E_b}{C_{2D} N_0}} \right) \quad (6.59)$$

as the average MI on the prior information on \mathbf{y}_2 that is received from the channel.

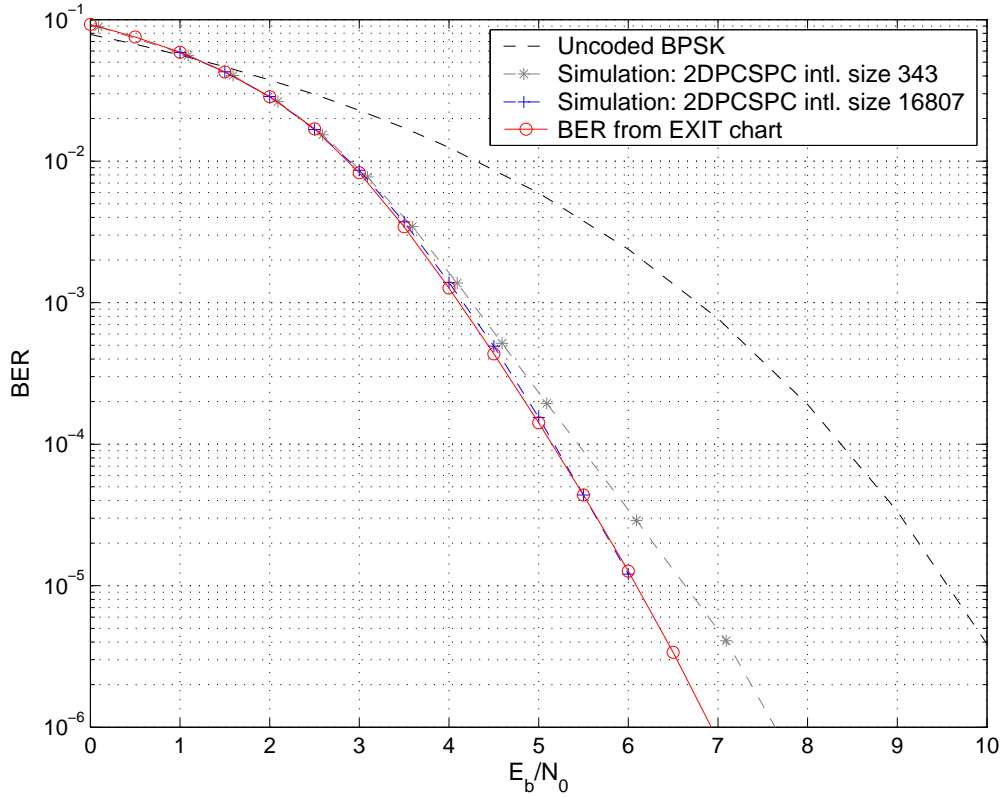


Figure 6.8. BER for a 2D PCSPC(8,7) code obtained from simulations and EXIT charts.

However, C_1^{-1} is no longer connected to the channel, but only to the output of C_2^{-1} and thus

$$I_{A(y_1)} = I_{E(x_2)}. \quad (6.60)$$

Further

$$I_{A(x_2)} = I_{E(y_1)}, \quad (6.61)$$

and

$$I_{A(x_1)} = 0, \quad (6.62)$$

since that input is not connected. Finally we get

$$I_{E(y_1)} = f_{y_1}(0, I_{E(x_2)}), \quad (6.63)$$

$$I_{E(x_2)} = f_{x_2} \left(I_{E(y_1)}, J \left(\sqrt{\frac{8r_{C_2D}^{-1} E_b}{N_0}} \right) \right). \quad (6.64)$$

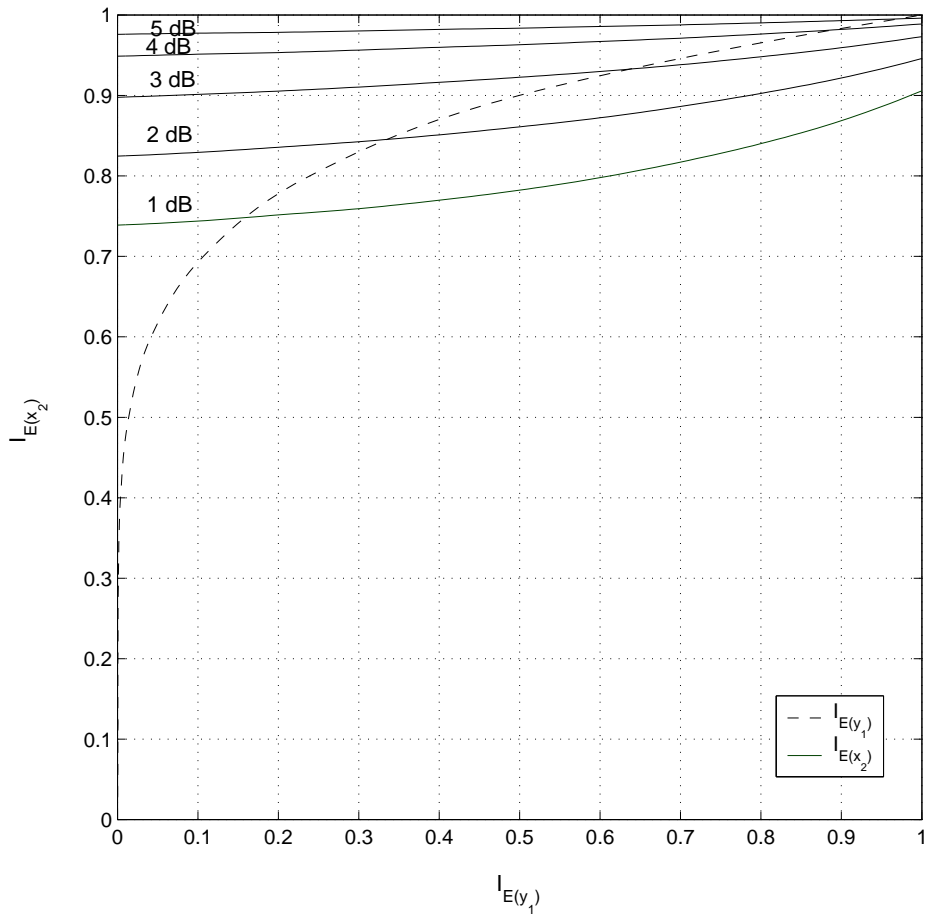


Figure 6.9. EXIT chart for a 2D serially concatenated SPC(8,7) code.

Here the EXIT functions in (6.63) and (6.64) are the ones showed in Figure 6.5 since the model of Figure 2.7 is used. Note that $I_{E(x_2)}$ depends both on $I_{E(y_1)}$ and on E_b/N_0 , whereas $I_{E(y_1)}$ is independent of E_b/N_0 . The EXIT chart for the 2D serially concatenated SPC(8,7) code is plotted in Figure 6.9 for a range of E_b/N_0 values. Since $I_{E(y_1)}$ is independent of E_b/N_0 , it has only one curve in Figure 6.9.

Returning to the system model and the notation of Figure 2.6 for a 2D SCSPC code, the decision statistics for $D(\mathbf{x})$ is

$$D(\mathbf{x}) = C(\mathbf{x}_0) + E(\mathbf{x}_1) + \Pi_1^{-1}(E(\mathbf{x}_2^{x_0})), \quad (6.65)$$

and hence the BER can be calculated as

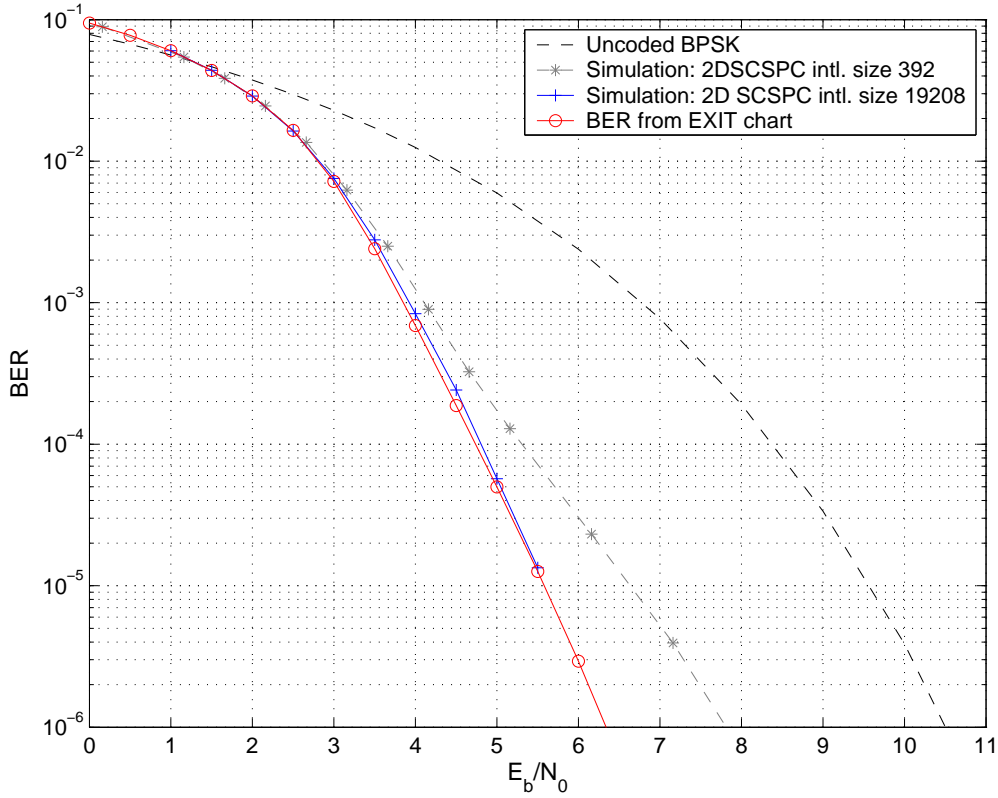


Figure 6.10. BER for 2D SCSPC(8,7) code obtained from simulations and EXIT charts.

$$\begin{aligned}
 P_b &\approx Q\left(\frac{\sigma_{D(x)}}{2}\right) = Q\left(\frac{J^{-1}(I_{D(x)})}{2}\right) = \\
 &= Q\left(\frac{\sqrt{\sigma_{A(x_0)}^2 + \sigma_{E(x_1)}^2 + \sigma_{E(x_2^{x_0})}^2}}{2}\right) = Q\left(\frac{\sqrt{\frac{8r_{C_{2D}} E_b}{N_0} + J^{-1}(I_{E(x_1)})^2 + J^{-1}(I_{E(x_2^{x_0})})^2}}{2}\right). \quad (6.66)
 \end{aligned}$$

Note that $I_{E(x_2^{x_0})} = I_{E(x_2)}$ since $I_{E(x_2)}$ denotes the average MI on the vector \mathbf{x}_2 and hence all parts of the vector will have the same average. In the serial case the average MI of the extrinsics needed to calculate the BER are not the ones that are plotted in the EXIT chart in Figure 6.9.

In Figure 6.10 we have plotted the BER curve obtained from the EXIT chart of a 2D serially concatenated SPC(8,7) code and compared it to simulations of the same code using two different sizes of the interleaver. As can be seen in Figure 6.10 the BER from EXIT charts are quite accurate also in this example, especially when the interleaver is sufficiently large.

Even for a relatively high BER the average MI on the decision statistics is close to one. Hence, it can be numerically challenging to get accurate estimates of very low BERs. This

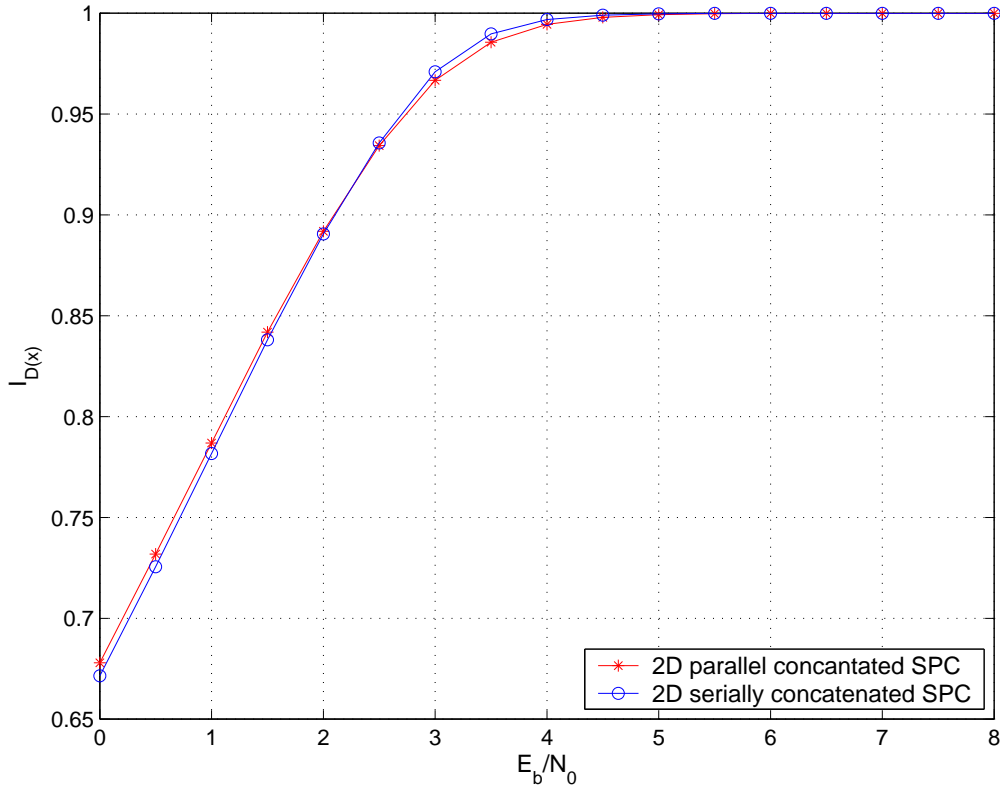


Figure 6.11. Average MI for the decision statistics $D(\mathbf{x})$ after convergence as a function of E_b/N_0 for a 2D PCSPC and a 2D SCSPC code.

can be observed in Figure 6.11 where we have plotted the corresponding decisions statistics for the BER curves obtained in Figure 6.8 and Figure 6.10.

6.3.2.1 EXIT charts for multidimensional concatenated codes

For multidimensional concatenated codes of arbitrary dimensions, i.e., codes constructed by a concatenation of two or more codes, the above equations can be generalized. In the parallel case, the average MI on the prior information on \mathbf{y}_l that is received from the channel is still

$$I_{A(y_l)} = J \left(\sqrt{\frac{8r_{C_{VD}} E_b}{N_0}} \right), \quad \forall l = 1, 2, \dots, U, \quad (6.67)$$

for each decoder, where U is the number of dimensions. However, as prior information on \mathbf{x} we have both the information received from the channel plus the extrinsic information received from all the other component decoders. Thus

$$I_{A(x_l)} = J \left(\sqrt{\frac{8r_{C_{lD}^{\parallel}} E_b}{N_0} + \sum_{\substack{i=1 \\ i \neq l}}^U J^{-1} (I_{E(x_i)})^2} \right), \quad (6.68)$$

which in turn yields

$$I_{E(x_l)} = f_{x_l} (I_{A(x_l)}, I_{A(y_l)}) = f_{x_l} \left(J \left(\sqrt{\frac{8r_{C_{lD}^{\parallel}} E_b}{N_0} + \sum_{\substack{i=1 \\ i \neq l}}^U J^{-1} (I_{E(x_i)})^2} \right), J \left(\sqrt{\frac{8r_{C_{lD}^{\parallel}} E_b}{N_0}} \right) \right). \quad (6.69)$$

for $l = 1, 2, \dots, U$. For 2D codes it was possible to plot the decoder transfer characteristics in a two-dimensional EXIT chart to visualize the decoding trajectory, but for multidimensional codes this visualization is more difficult. Therefore a technique presented in [99, 104] that projects the multidimensional EXIT chart onto a 2D EXIT chart will be described here.

First we assume that all EXIT functions are monotonically non-decreasing – an assumption which seems reasonable for any useful code and that also can be confirmed by visual inspection of the example codes used in this thesis. This assumption implies that the extrinsic MI for each component code will converge regardless of which order the component decoders are activated, as long as sufficient activations are allowed [99, 104].

Next, we need to decide onto which two dimensions we want to make the projection. Since the parallel concatenated SPC(8,7) is symmetric in all dimensions, we simply select the first two dimensions. The projection can now be done by first setting the extrinsic MI for all component codes equal to zero. Then for each value $0 \leq I_{E(x_1)} \leq 1$ we activate all the other decoders $l = 2, 3, \dots, U$ using (6.69) until the decision statistics has converged. Then we again reset all the extrinsic MI and do the same thing for each value $0 \leq I_{E(x_2)} \leq 1$ where all the other decoders $l = 1, 3, \dots, U$ are activated until the decision statistics has converged. A detailed algorithm for EXIT chart projections can be found in [30].

In Figure 6.12 the EXIT chart for a 3D parallel concatenated SPC(8,7) code is shown. A staircase shaped trajectory bounded by the decoder transfer characteristic can still be drawn into the projected EXIT chart, but the number of activations can no longer be visualized this way. This is due to the fact that a vertical step now represents an arbitrary number of activations of decoders C_1^{-1} and C_3^{-1} , until the decision statistics has converged. Similarly, a horizontal step represents an arbitrary number of activations of decoders C_2^{-1} and C_3^{-1} , until the decision statistics has converged.

In Figure 6.13 and Figure 6.14 we have plotted the EXIT chart for the 4D and 5D parallel concatenated SPC(8,7) codes, for a range of E_b/N_0 values. As can be seen there are no tunnels opening up and hence the only way to converge to a low BER, i.e., to reach the upper right corner, is to increase the SNR.

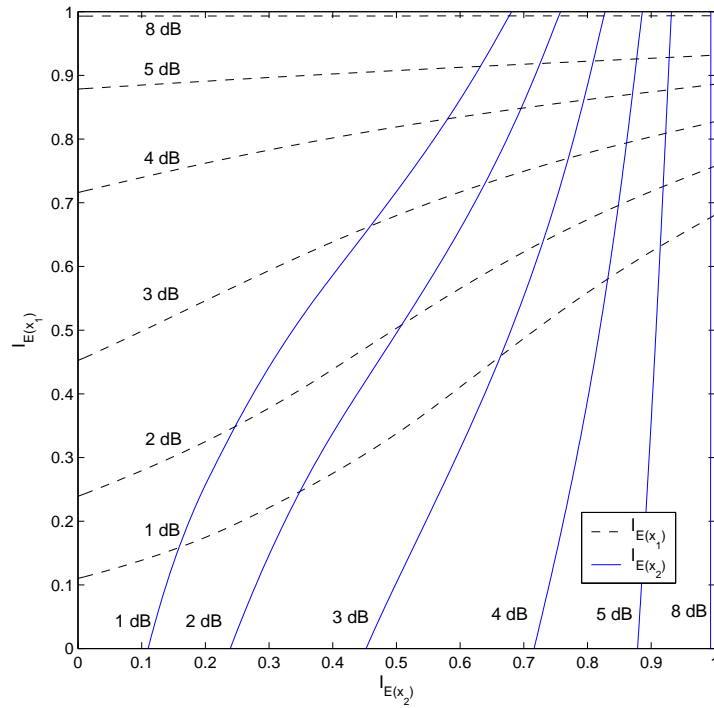


Figure 6.12. EXIT chart for a 3D parallel concatenated SPC(8,7) code.

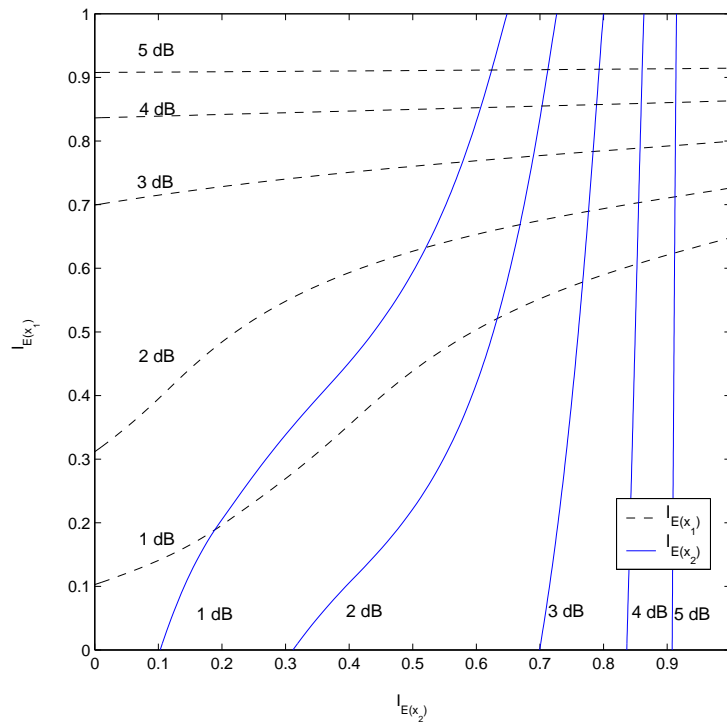


Figure 6.13. EXIT chart for a 4D parallel concatenated SPC(8,7) code.

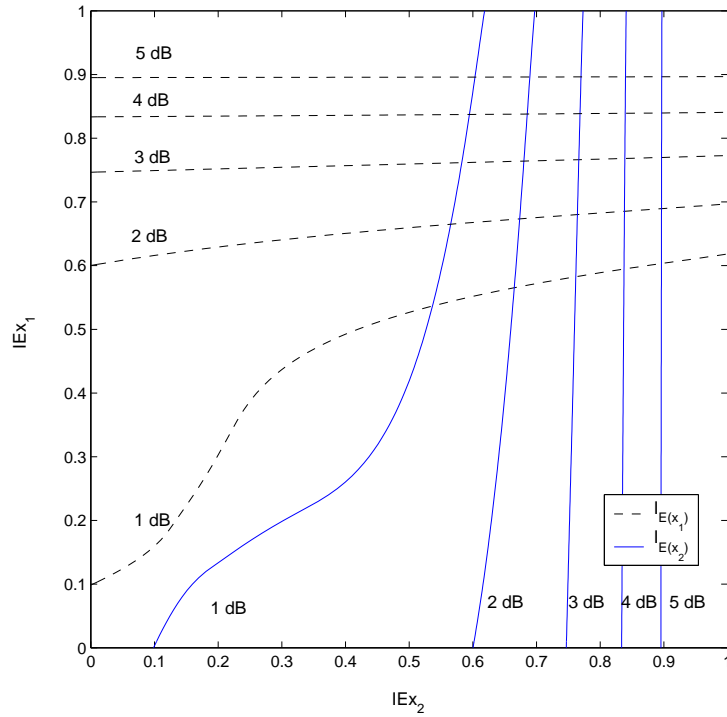


Figure 6.14. EXIT chart for 5D parallel concatenated SPC(8,7) code.

The BER can still be found using the projected EXIT chart, but now

$$P_b \approx Q \left(\frac{\sqrt{\frac{8r_{c_{UD}} E_b}{N_0} + \sum_{l=1}^U J^{-1}(I_{E(x_l)})^2}}{2} \right). \quad (6.70)$$

In Figure 6.15 the BER for parallel concatenated SPC(8,7) codes with 2-5D are shown, as attained both from simulations using large interleavers as described in Chapter 4, and from EXIT charts. As can be seen the EXIT chart is very accurate in this case when the interleavers are sufficiently large and there are no waterfalls.

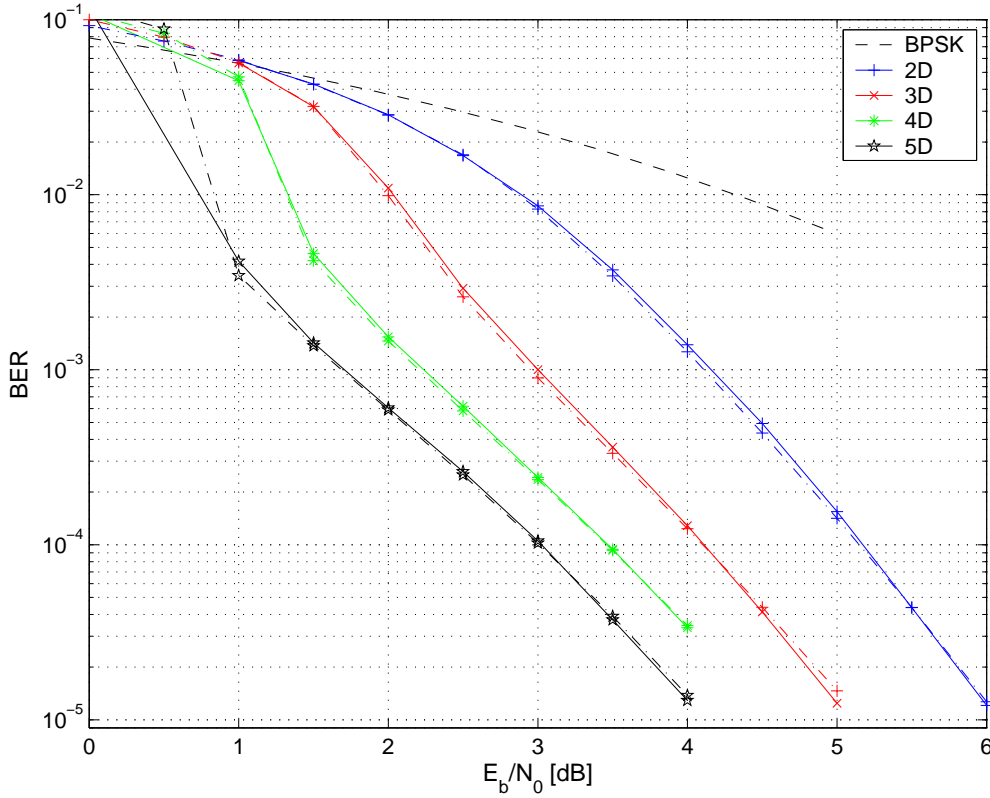


Figure 6.15. BER obtained from simulations (solid lines) and EXIT chart (dash-dotted lines) for 2D, 3D, 4D and 5D PCSPC(8,7) codes.

When constructing the EXIT charts for multidimensional serial concatenated codes we again return to the notation of the encoder depicted in Figure 2.7 in the system model. We choose the two outermost codes with $I_{E(y_1)}$ and $I_{E(x_2)}$ as the two dimensions onto which we want to make the projection. The expressions below are derived for a 3D serial concatenated code assuming that C_3^{-1} are closest to the channel. It is, however, straightforward, to extend to $U > 3$. Consequently,

$$I_{A(y_3)} = J \left(\sqrt{\frac{8r_{C_{3D}^{-1}} E_b}{N_0}} \right) \quad (6.71)$$

and

$$I_{A(x_3)} = I_{E(y_2)} \quad (6.72)$$

are the average MIs for the priors of C_3^{-1} . For C_2^{-1} we have

$$I_{A(y_2)} = I_{E(x_3)}, \quad (6.73)$$

$$I_{A(x_2)} = I_{E(y_1)}, \quad (6.74)$$

and finally for C_1^{-1} we have

$$I_{A(y_1)} = I_{E(x_2)}, \quad (6.75)$$

$$I_{A(x_1)} = 0. \quad (6.76)$$

This yields the average MI on the extrinsic outputs as

$$I_{E(y_1)} = f_{y_1}(0, I_{E(x_2)}), \quad (6.77)$$

$$I_{E(x_2)} = f_{x_2}(I_{E(y_1)}, I_{E(x_3)}), \quad (6.78)$$

$$I_{E(x_3)} = f_{x_3}\left(I_{E(y_2)}, J\left(\sqrt{\frac{8r_{\text{C3D}} E_b}{N_0}}\right)\right), \quad (6.79)$$

$$I_{E(y_2)} = f_{y_2}(I_{E(y_1)}, I_{E(x_3)}). \quad (6.80)$$

Finally $I_{E(y_1)}$ and $I_{E(x_2)}$ are plotted in the EXIT chart after convergence have been reached.

In Figure 6.16 the EXIT chart is plotted for a 3D serially concatenated SPC(8,7) code. We can see that for 2 dB there is a tunnel opening almost all the way up to the upper right corner. In Figure 6.17 the EXIT chart for a 4D serially concatenated SPC(8,7) code are given. For 1.5 dB there is already a quite big tunnel. Finally in Figure 6.18 the EXIT chart for the 5D serially concatenated SPC(8,7) code is depicted. The 1 dB curve in Figure 6.18 can exemplify the pinch-off region since there is no tunnel and the decoder transfer characteristics intersect at a low average MI resulting in a high BER. The second scenario represented by the 1.2 dB curve in Figure 6.18, is when a small tunnel is just opening up so that convergence to a low BER is slow but possible. Finally the third case exemplified by the curve marked 1.5 dB is where a big tunnel is open, yielding fast convergence to a low BER.

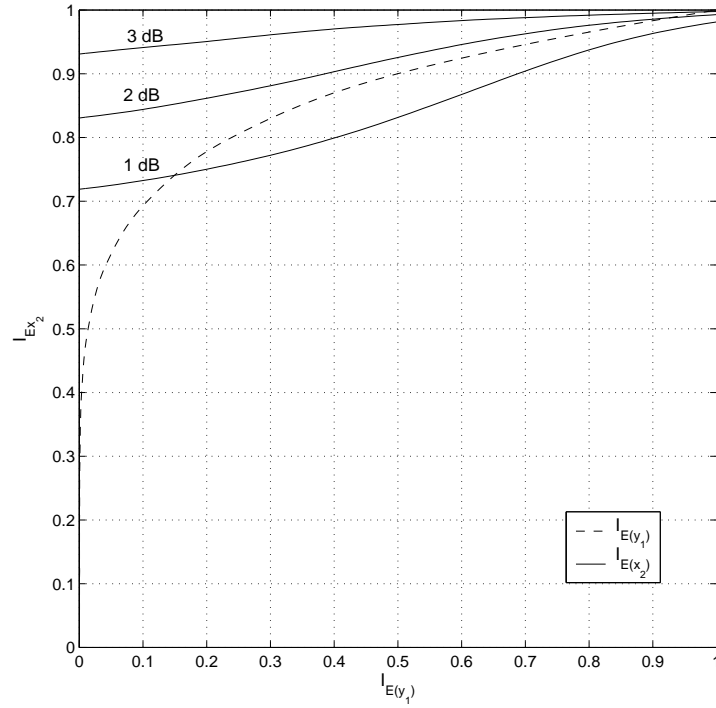


Figure 6.16. EXIT chart for 3D serially concatenated SPC(8,7) code.

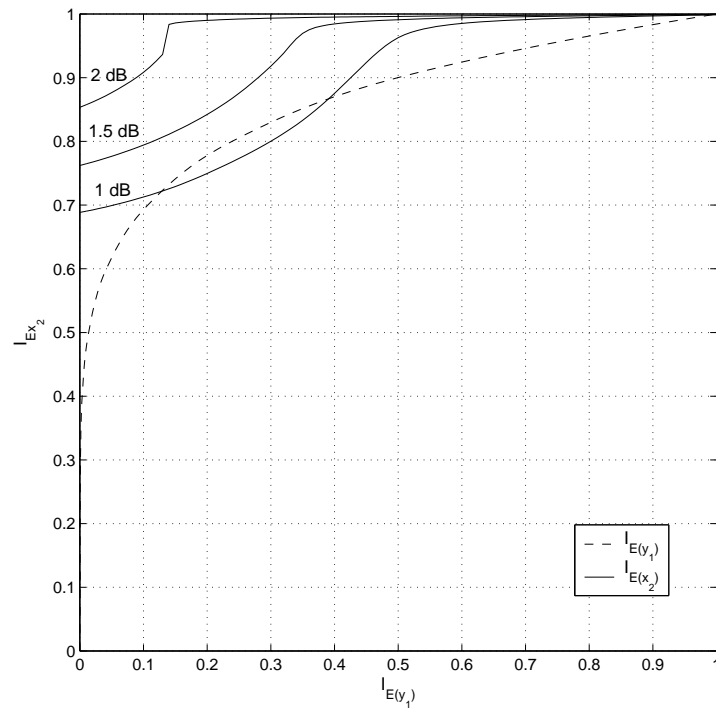


Figure 6.17. EXIT chart for 4D serially concatenated SPC(8,7) code.

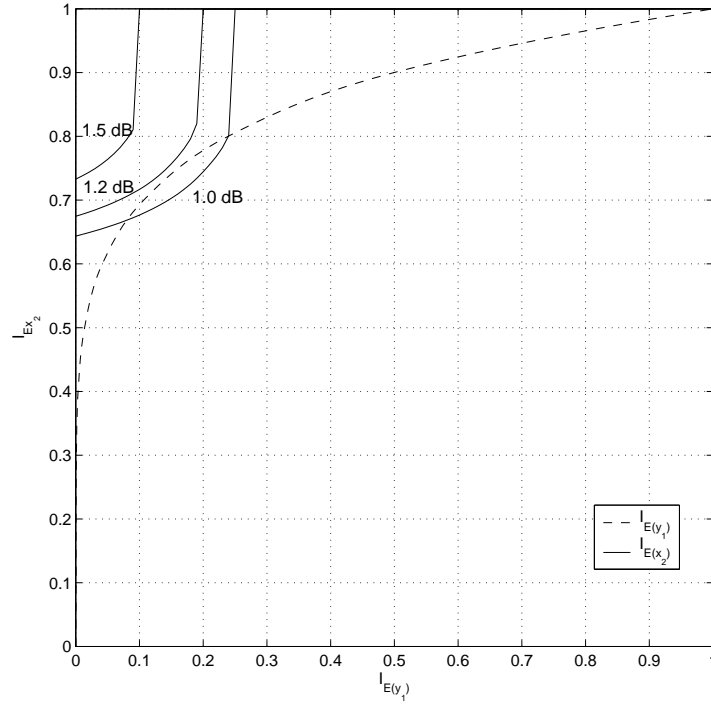


Figure 6.18. EXIT chart for 5D serially concatenated SPC(8,7) code.

We can also see that the BER curves confirm this. The BER for a 3D serially concatenated code can be expressed as

$$P_b \approx Q\left(\frac{\sigma_{D(x)}}{2}\right) = Q\left(\frac{J^{-1}(I_{D(x)})}{2}\right) = Q\left(\frac{\sqrt{\sigma_{A(x_0)}^2 + \sigma_{E(x_1)}^2 + \sigma_{E(x_2^{30})}^2 + \sigma_{E(x_3^{30})}^2}}{2}\right) \quad (6.81)$$

$$= Q\left(\frac{\sqrt{\frac{8r_{C_{3D}}^{\perp} E_b}{N_0} + J^{-1}(I_{E(x_1)})^2 + J^{-1}(I_{E(x_2)})^2 + J^{-1}(I_{E(x_3)})^2}}{2}\right).$$

In Figure 6.19 the BER for serially concatenated SPC(8,7) codes with 2-5D are shown, as attained both from simulations using large interleavers as described in Chapter 4, and from EXIT charts. As can be seen the EXIT chart is less accurate in this case when we have waterfall regions for the case of 3D-5D. We can however confirm that for 1 dB the 5D curve is above the waterfall region and at 1.2 dB we are in the waterfall region. The BER floor region for serially concatenated codes is generally more difficult to reach by simulations, since it is often fairly low.

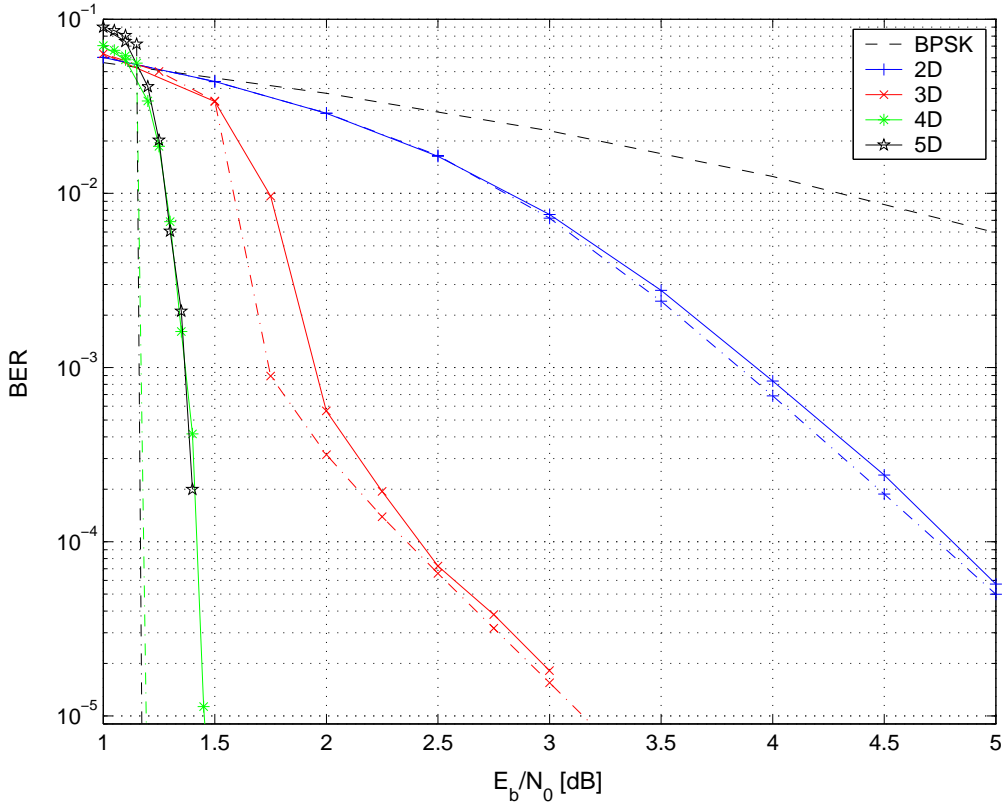


Figure 6.19. BER obtained from simulations (solid lines) and EXIT chart (dash-dotted lines) for 2D, 3D, 4D and 5D SCSPC(8,7) codes.

6.3.3 Puncturing

Thus far we have assumed unpunctured codes. However, most of the codes used in this thesis will be punctured to allow for incremental redundancy through rate compatible codes. For punctured concatenated codes, some generalizations to the above equations are needed. Starting with parallel concatenated codes the average MI of the prior information on \mathbf{y}_l that is received from the channel for each decoder is now

$$I_{A(\mathbf{y}_l)} = \delta_l J \left(\sqrt{\frac{8r'_{c_{lD}} E_b}{N_0}} \right), \quad \forall l = 1, 2, \dots, U, \quad (6.82)$$

where $0 \leq \delta_l \leq 1$ is the fraction of bits from \mathbf{y}_l that are transmitted. Note that when no coded bits are punctured $\delta_l = 1$, which has been the case in previous sections. Also note that the rate of the code changes when it is punctured, hence the prime introduced on $r'_{c_{lD}}$. As prior information on \mathbf{x} we still have both the information that is received from the channel and the extrinsic information received from all the other component decoders. Thus

$$I_{A(x_i)} = J \left(\sqrt{J^{-1} \left(\delta_0 J \left(\sqrt{\frac{8r'_{c_{LD}} E_b}{N_0}} \right) \right)^2 + \sum_{\substack{i=1 \\ i \neq l}}^U J^{-1} (I_{E(x_i)})^2} \right), \quad (6.83)$$

since puncturing is only made on the prior information received from the channel. Note that since we have puncturing, we can no longer use the variance on $\sigma_{C(x_0)} = \sigma_{A(x_0)}$ directly so we need to keep the J -functions according to (6.83). Finally, we get [30]

$$\begin{aligned} I_{E(x_i)} &= f_{x_i} (I_{A(x_i)}, I_{A(y_i)}) \\ &= f_{x_i} \left(J \left(\sqrt{J^{-1} \left(\delta_0 J \left(\sqrt{\frac{8r'_{c_{LD}} E_b}{N_0}} \right) \right)^2 + \sum_{\substack{i=1 \\ i \neq l}}^U J^{-1} (I_{E(x_i)})^2} \right), \delta_l J \left(\sqrt{\frac{8r'_{c_{LD}} E_b}{N_0}} \right) \right). \end{aligned} \quad (6.84)$$

Since the rate of the code changes when the code is punctured, it is convenient to express the code rate, $r'_{c_{LD}}$, as a function of δ_l and the rates of the component codes as

$$r'_{c_{LD}} = \left(\sum_{l=0}^U \frac{\delta_l}{r_{C_l}} \right)^{-1}, \quad (6.85)$$

where r_{C_l} is the code rate of the individual component codes used. Note that r_{C_0} denotes the rate of the systematic bits \mathbf{x}_0 and is thus always equal to one. Similarly, δ_0 denotes the fraction of the systematic bits that are unpunctured. The BER is now given by

$$P_b \approx Q \left(\frac{\sqrt{J^{-1} \left(\delta_0 J \left(\sqrt{\frac{8r'_{c_{LD}} E_b}{N_0}} \right) \right)^2 + \sum_{l=1}^U J^{-1} (I_{E(x_l)})^2}}{2} \right). \quad (6.86)$$

With the above expressions we can construct EXIT charts for punctured multi-dimensional parallel concatenated codes. Further, for a given code rate, we can make a search to find the optimal values of $\Delta = [\delta_0, \delta_1, \dots, \delta_U]$ that yields the best possible decoder transfer characteristics so that fast convergence to a low BER takes place at the lowest possible value of E_b/N_0 [30]. To do this we first need to determine what we mean by low BER, i.e., we need to establish a target BER. Having done this we can determine at what E_b/N_0 the search should start. In order to maintain a notation similar to [30] we define

$$\gamma_b \triangleq \frac{E_b}{N_0}, \quad (6.87)$$

$$\gamma_s \triangleq \frac{E_s}{N_0} = \frac{r_C E_b}{N_0}. \quad (6.88)$$

We know that for an AWGN channel we have $P_b = Q(\mu/\sigma)$ and for uncoded BPSK we have $P_b = Q(\sqrt{2\gamma_s}) = Q(\sqrt{2\gamma_b})$. Since a coded system is better than uncoded BPSK for most values of γ_b , we can use this as a reference. Hence, we start the search at

$$\gamma_b = \frac{Q^{-1}(P_b^*)^2}{2}, \quad (6.89)$$

where P_b^* is the selected target bit error rate⁵. The algorithm searching for optimal puncturing ratios then goes through each value of $\Delta = [\delta_0, \delta_1, \dots, \delta_U]$ that yields the required code rate according to (6.85). Starting with the first valid Δ , we update the extrinsic MI for all the component decoders until the decision statistics has converged. If the converged decision statistics yields a BER that is lower or equal to P_b^* , the corresponding Δ vector and the current value of γ_b are saved. In addition, we lower γ_b by a small number, $\varepsilon_\gamma > 0$, and check to see if the chosen Δ vector still results in a decision statistics corresponding to $P_b \leq P_b^*$. If so, γ_b is lowered again. If, on the other hand, the resulting BER is not lower than P_b^* , the next valid Δ vector is tried. In short, we first pick a valid Δ vector and find the lowest value of γ_b resulting in a $P_b \leq P_b^*$. Then, we try the next valid Δ vector to see if this will result in an even lower value on γ_b . Hence [30]

$$[\gamma_b^*, \Delta^*] = \arg \min_{[\gamma_b, \Delta]} g(\gamma_b, \Delta) \quad (6.90)$$

where

$$g(\gamma_b, \Delta) = \begin{cases} \gamma_b & P_b \leq P_b^*, \\ \infty & \text{otherwise.} \end{cases} \quad (6.91)$$

A detailed algorithm for this procedure can be found in [30].

For a 2D PCSPC(8,7) code, the γ_b needed for convergence to P_b^* , when the optimal values of Δ are used, is plotted as a function of the code rate in Figure 6.20. Curves corresponding to four different values of P_b^* are plotted together with the C^* that can be used as a reference. Note that it is possible to achieve rates above capacity if the target BER is sufficiently high, since C^* assumes an arbitrary low BER.

Since the lowest possible code rate for a 2D PCSPC(8,7) is 7/9, the curves stop here. There is a knee on all the curves with a low target BER, which is due to a dimension crossing at code rate 7/8. These curves also show similar behaviors within each dimension, i.e., the curves have similar shape both below and above the knee. Also, it can be noted that the γ_b required for convergence is actually decreasing when going from a code rate of 0.98 to one for curves with a low target BER. The reason for this is the same as in the simulations, i.e., not much is gained from transmitting only a few parity bits in a new

⁵ Note the target error rate selected when obtaining puncturing ratios, P_b^* , is not the same as P_t , which is the target error rate for the IR-HARQ system.

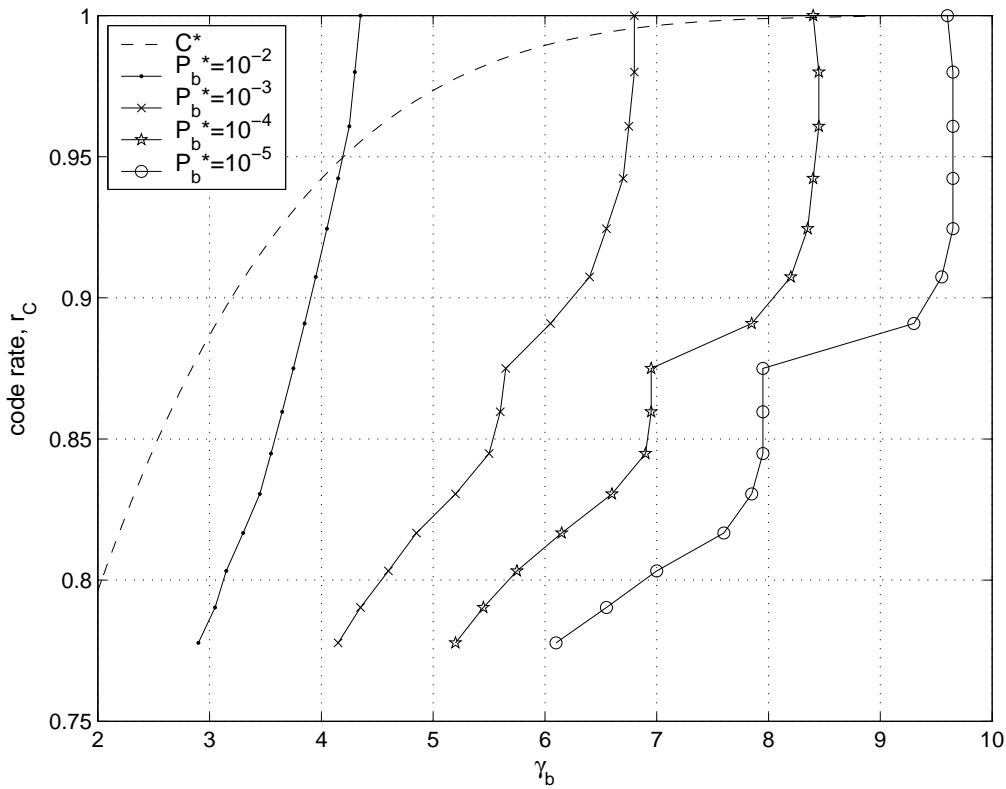


Figure 6.20. The γ_b -values required for convergence to P_b^* as a function of the code rate, r_c , for a 2D PCSPC(8,7) code that uses optimal values of Δ . As a reference C^* is also plotted.

dimension and in fact we may lose more in bit energy than we gain by the reduced code rate.

The optimal values of Δ for different code rates are plotted in Figure 6.21 for $P_b^* = 10^{-5}$. As can be seen, it is preferable to puncture some systematic bits for low code rates. However, for a code rate higher than 0.83 dimension-wise puncturing of parity bits is best. We can also see from Figure 6.21 that, apart from the puncturing of information bits for low code rates, the optimal puncturing ratios are rate compatible, since the curves are all decreasing.

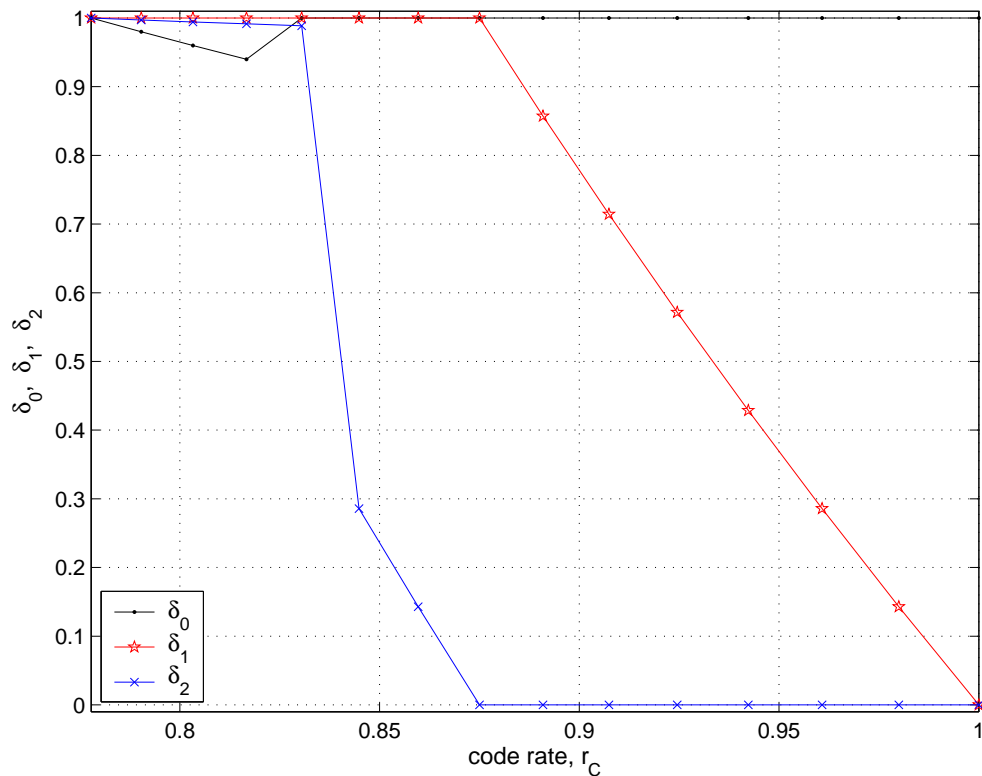


Figure 6.21. Optimal puncturing ratios for the 2D PCSCP(8,7) code when the target BER is 10^{-5} .

In Figure 6.22 the γ_b is again plotted as a function of the code rate, but for both a 2D and a 3D parallel concatenated SPC(8,7) code. The curves for the 3D codes go all the way down to rate 7/10, whereas the 2D codes stop at 7/9. The performance for a 3D code is much better than, and at worst equally good as a 2D code. Only when we have code rates close to one, is the performance of the 2D and 3D codes equal. It is also at these code rates that no systematic bits are punctured. Hence, whenever the code rate allows sufficient parity bits to be included, so that systematic bits can be punctured, this improves performance.

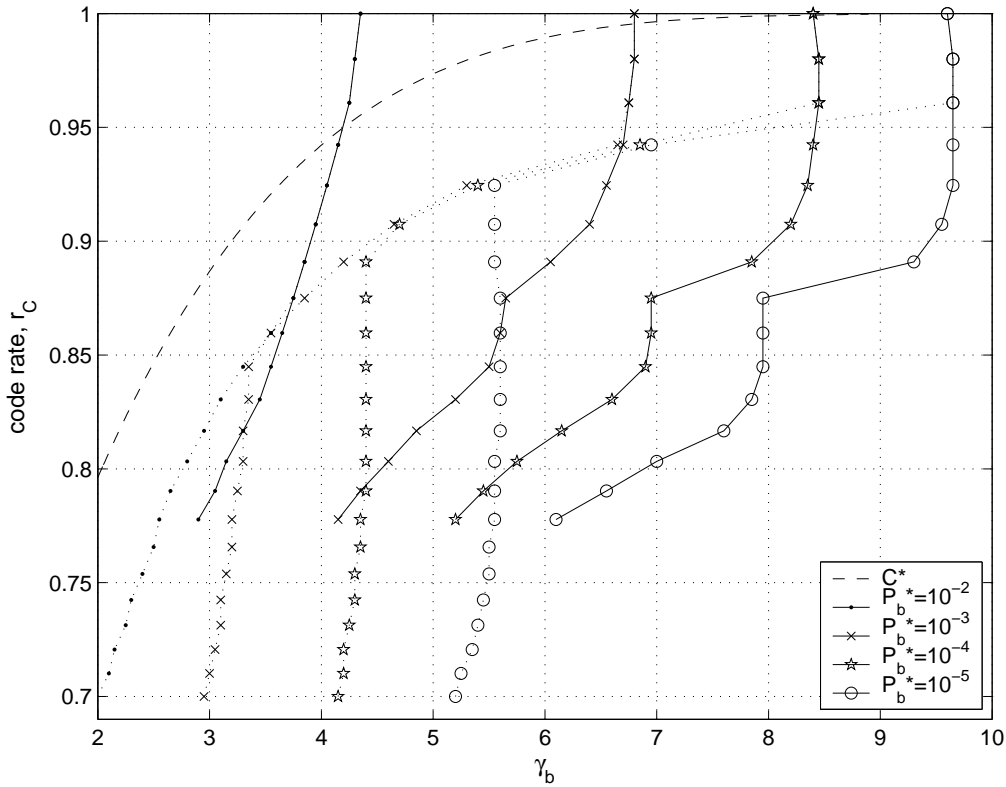


Figure 6.22. The γ_b -values required for convergence to P_b^* as a function of the code rate r_c for a 2D (solid lines) and a 3D (dotted lines) PCSPC(8,7) code that uses optimal values of Δ . C^* is plotted as a reference.

In Figure 6.23 the optimal values of Δ for different code rates are plotted for $P_b^* = 10^{-2}$. Clearly, it is preferable to puncture systematic bits for low code rates. As soon as all the systematic bits are restored for higher code rates, dimension-wise puncturing of parity bits is best. The optimal puncturing ratios are not rate compatible, since the curves are increasing when going from a lower to a higher code rate. This can be seen even more clearly in Figure 6.24 where the optimal values of Δ for different code rate are plotted, but now for $P_b^* = 10^{-5}$. In this case, even more systematic bits are punctured. Note that when $P_b^* = 10^{-5}$ we are generally operating at a higher γ_b .

Recall that for SPC codes, when two bits pertaining to the same component codeword are punctured, this leads to the local invertibility being lost. Depending on the interleavers, the invertibility may or may not be restored by the component codes in the remaining dimensions. Since EXIT charts assume infinite interleavers, the probability of pairing up punctured bits in the same component code is relatively small, since the frame length of the component codes is small compared to the interleaver size. At a low code rate, most of the parity bits are included, resulting in sufficient parity bits so that each component frame of length seven has three parity bits with a high probability. This allows for puncturing as many as three bits from a component codeword without losing the invertibility. Together

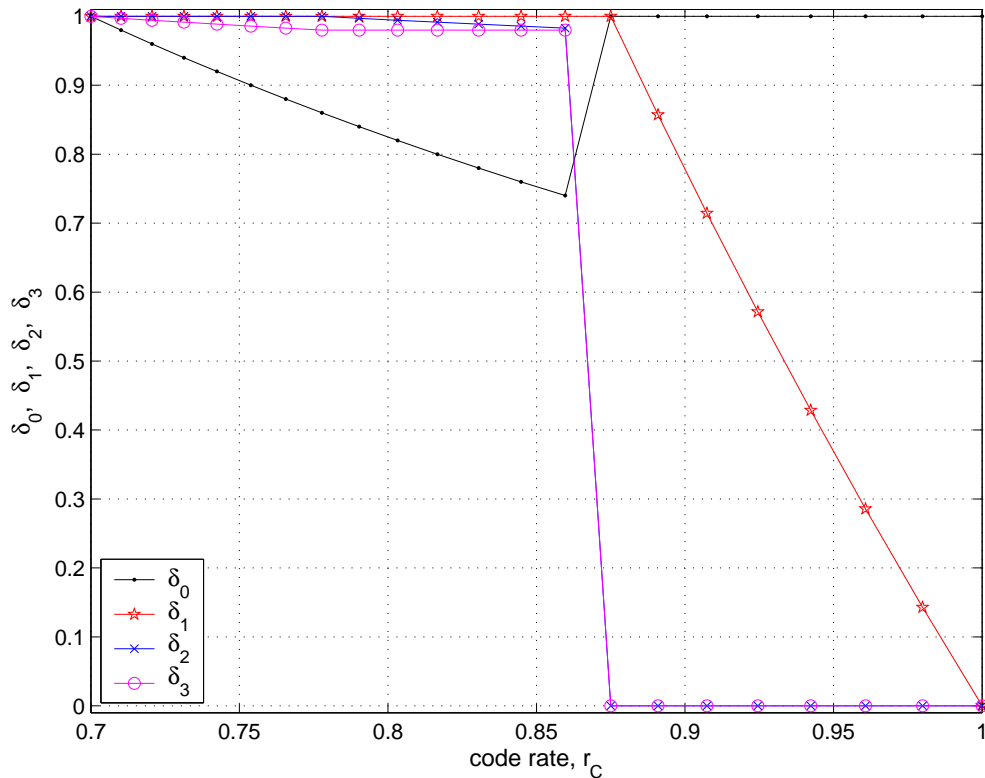


Figure 6.23. Optimal puncturing ratios for the 3D PCSCP(8,7) code when the target BER is 10^{-2} .

with the assumption of infinite interleaves, this explains why EXIT chart analysis suggests puncturing of systematic bits for low code rates.

Consequently, with knowledge of the particular random interleavers used, the puncturing pattern can be adapted so that the local invertibility is guaranteed. In this case, systematic bits should be punctured to increase the performance. However, if we want to guarantee the local invertibility regardless of which interleavers are used, puncturing of systematic bits should be avoided. Hence, we consider three different scenarios when searching for optimal puncturing ratios:

1. Optimal search with no constraints on Δ . This is what is done in (6.90) and reported in Figure 6.22.
2. Optimal search with the constraint that $\delta_0 = 1$, i.e., no systematic bits are punctured.
3. Optimal search for rate compatible puncturing ratios, i.e., when going from a low code rate to a higher code rate, the values in Δ are not allowed to increase.

Finally, we would like to see how dimension-wise puncturing compares to the optimal puncturing ratios.

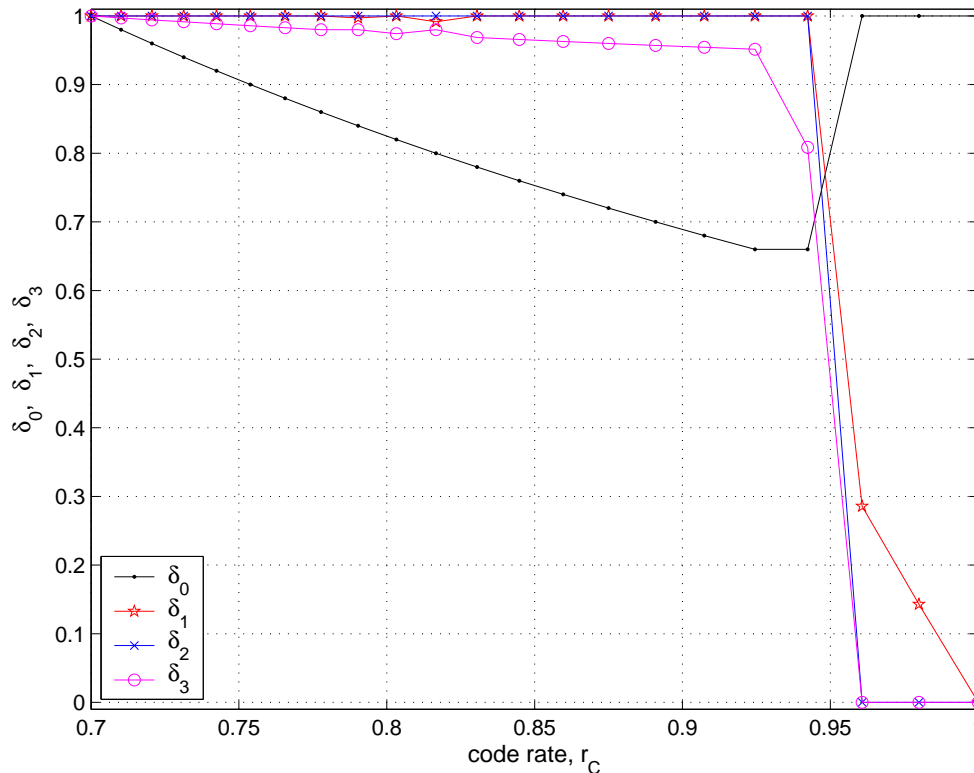


Figure 6.24. Optimal puncturing ratios for the 3D PCSCP(8,7) code when the target BER is 10^{-5} .

In Figure 6.25 the optimal search of scenario 1 and 2 are reported. We can see that when systematic bits are not allowed to be punctured, as in scenario 2, the performance of a 3D PCSPC(8,7) code is compatible to that of the 2D PCSPC(8,7), except for low code rates, where the 2D code, using scenario 1, punctures some systematic bits. The optimal values on Δ when scenario 2 is considered are basically the same as dimension-wise puncturing. When plotting the performance of dimension-wise puncturing in Figure 6.25 it is indistinguishable from scenario 2. The reason for getting δ -values closely, but not exactly corresponding to dimension-wise puncturing, when considering scenario 2, is most likely due to the fact that several puncturing patterns may result in similar performance. However, since the performance of dimension-wise puncturing is the same as that of scenario 2, we can conclude that dimension-wise puncturing is optimal under the constraint that $\delta_0 = 1$.

When searching for rate compatible puncturing ratios, we start from the lowest possible code rate and start puncturing, i.e., start decreasing the δ -values, making sure that none of them are allowed to be increased once decreased. This rate compatible search results in puncturing ratios with a performance that follows the optimal curves from scenario one and hence punctures more and more information bits as the code rate increases. However, when

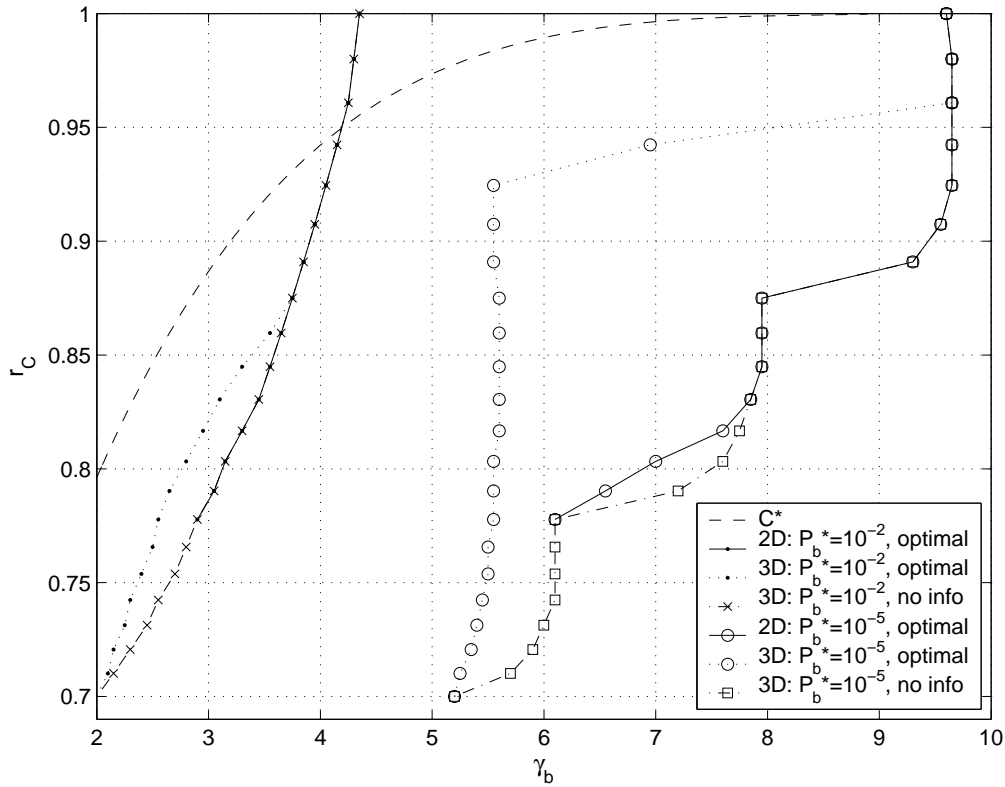


Figure 6.25. The γ_b -values required for convergence to P_b^* as a function of the code rate r_c for a 2D (solid lines) and a 3D PCSPC(8,7) code that uses optimal values of Δ (dotted lines) and optimal values of Δ with the constraint that $\delta_0 = 1$ (dash-dotted lines).

reaching a high enough code rate it is no longer possible to find a Δ for which $P_b \leq P_b^*$. This typically occurs when the code rate is at levels where the optimal scenario changes from e.g., $\delta_0 = 0.66$ to $\delta_0 = 1$ for $P_b^* = 10^{-5}$ as can be seen in Figure 6.24 at a code rate of around 0.96.

It may be argued that for an IR-HARQ scheme, we would prefer to optimize the performance at high code rates since that may increase the average code rate of the IR-ARQ scheme. Hence, the search could alternatively start at the highest possible code rate. However, the best puncturing pattern for a 3D PCSPC(8,7) code of rate one is to only transmit the systematic bits, i.e., $\Delta = [1, 0, 0, 0]$. Consequently, this search results in dimension-wise puncturing. The only way to get a different puncturing pattern is if the search starts at a code rate lower than one, implying that a longer packet is sent in the initial transmission. That does, however, require knowledge about the IR packet lengths. Consequently, we may conclude that dimension-wise puncturing is a good rate compatible puncturing pattern for the 3D PCSPC(8,7) code. Recall that, due to the very simple structure of SPC codes, it is irrelevant which particular parity bit from a certain dimension that is punctured. Hence, the obtained puncturing ratios translated directly to puncturing patterns for SPC codes.

When considering punctured serially concatenated codes we keep the encoder depicted in Figure 2.6 in the system model, since that enables individual puncturing of each of the outputs \mathbf{x} and \mathbf{y}_l for $l = 1, 2, \dots, U$. When constructing EXIT charts it was easier to look at the equivalent system model in Figure 2.7, but here we are interested in the behavior after convergence when individual puncturing is applied. The expressions below are derived for a 3D serial concatenated code assuming that C_3^{-1} is closest to the channel. However, extending to $U > 3$ is straightforward. For notation simplicity, we make the following definition

$$I_{E(x_1)} \oplus I_{E(x_2)} \triangleq J \left(\sqrt{J^{-1} (I_{E(x_1)})^2 + J^{-1} (I_{E(x_2)})^2} \right). \quad (6.92)$$

Recall from (2.18) that $A(\mathbf{x}_1)$ receives input from the channel and from the two other component codes. Further, δ_0 is the fraction of elements in $A(\mathbf{x}_1)$ that has information from $C(\mathbf{x}_0)$, $E(\mathbf{x}_2)$ and $E(\mathbf{x}_3)$. Thus, $(1 - \delta_0)$ is the fraction on elements in $A(\mathbf{x}_1)$ that has information only from $E(\mathbf{x}_2)$ and $E(\mathbf{x}_3)$. Consequently, using (6.42) to calculate the average MI yields

$$I_{A(x_1)} = \delta_0 (I_{C(x_0)} \oplus I_{E(x_2)} \oplus I_{E(x_3)}) + (1 - \delta_0) (I_{E(x_2)} \oplus I_{E(x_3)}), \quad (6.93)$$

and similarly from (2.19)

$$I_{A(y_1)} = \delta_1 (I_{C(y_1)} \oplus I_{E(x_2)} \oplus I_{E(x_3)}) + (1 - \delta_1) (I_{E(x_2)} \oplus I_{E(x_3)}). \quad (6.94)$$

Recall that $\mathbf{x}_2 = \Pi_1([\mathbf{x}_0, \mathbf{y}_1])$ which means that 7/8 of the elements in $A(\mathbf{x}_2)$ comes from the systematic bits, \mathbf{x}_0 , and 1/8 of the elements come from parity bits, \mathbf{y}_1 , since the SPC(8,7) component code is used. Further, δ_0 of the elements coming from the systematic bits has information from $C(\mathbf{x}_0)$, $E(\mathbf{x}_1)$ and $E(\mathbf{x}_3)$, whereas $(1 - \delta_0)$ of the elements has information only from $E(\mathbf{x}_1)$ and $E(\mathbf{x}_3)$. The same reasoning can be made for the 1/8 of the elements coming from parity bits, but here with δ_1 instead. Using (2.20) we have

$$\begin{aligned} I_{A(x_2)} &= \delta_0 \frac{7}{8} \cdot (I_{C(x_0)} \oplus I_{E(x_1)} \oplus I_{E(x_3)}) + (1 - \delta_0) \frac{7}{8} (I_{E(x_1)} \oplus I_{E(x_3)}) + \\ &+ \delta_1 \frac{1}{8} (I_{C(y_1)} \oplus I_{E(y_1)} \oplus I_{E(x_3)}) + (1 - \delta_1) \frac{1}{8} (I_{E(y_1)} \oplus I_{E(x_3)}), \end{aligned} \quad (6.95)$$

and similarly from (2.21)

$$I_{A(y_2)} = \delta_2 (I_{C(y_2)} \oplus I_{E(x_3)}) + (1 - \delta_2) I_{E(x_3)}. \quad (6.96)$$

The reason for adding the MIs this way is that we have multiplexers in the serial case and hence we are trying to merge two vectors, or rather the MIs of two vectors. Finally, for C_3^{-1} we have $\mathbf{x}_3 = \Pi_2([\mathbf{x}_0, \mathbf{y}_1, \mathbf{y}_2])$ and using (2.22) together with the above reasoning yields

$$\begin{aligned}
I_{A(x_3)} &= \delta_0 \frac{7}{8} \cdot \frac{7}{8} (I_{C(x_0)} \oplus I_{E(x_1)} \oplus I_{E(x_2)}) + (1 - \delta_0) \frac{7}{8} \cdot \frac{7}{8} (I_{E(x_1)} \oplus I_{E(x_2)}) + \\
&+ \delta_1 \frac{7}{8} \cdot \frac{1}{8} (I_{C(y_1)} \oplus I_{E(y_1)} \oplus I_{E(x_2)}) + (1 - \delta_1) \frac{7}{8} \cdot \frac{1}{8} (I_{E(y_1)} \oplus I_{E(x_2)}) + \\
&+ \delta_2 \frac{1}{8} (I_{C(y_2)} \oplus I_{E(y_2)}) + (1 - \delta_2) \frac{1}{8} I_{E(x_2)}.
\end{aligned} \tag{6.97}$$

and similarly from (2.23)

$$I_{A(y_3)} = \delta_3 I_{C(y_3)}. \tag{6.98}$$

Note that

$$I_{C(x_0)} = I_{C(y_1)} = I_{C(y_2)} = I_{C(y_3)} = J \left(\sqrt{\frac{8r'_{C_{3D}} E_b}{N_0}} \right). \tag{6.99}$$

It can be shown that $r'_{C_{3D}}$ can be calculated using (6.85) with $r_{C_0} = 1$ and $r_{C_l} = n_{C_l} (k_{C_l} / n_{C_l})^l$ when the same component code C_l with parameters (k_{C_l}, n_{C_l}) is used in all dimensions $l = 1, 2, \dots, U$.

For a 2D SCSPC(8,7) code, the γ_b needed for convergence to P_b^* , when the optimal values of Δ are used, is plotted as a function of the code rate in Figure 6.26. Since the lowest possible code rate for a 2D SCSPC(8,7) is $(7/8)^2$, the curves stop here. Also in the serial case we observe a knee on all the curves with a low target BER, which is due to a dimension crossing at rate 7/8. The knee for the serial case is however, a bit softer than in the parallel case.

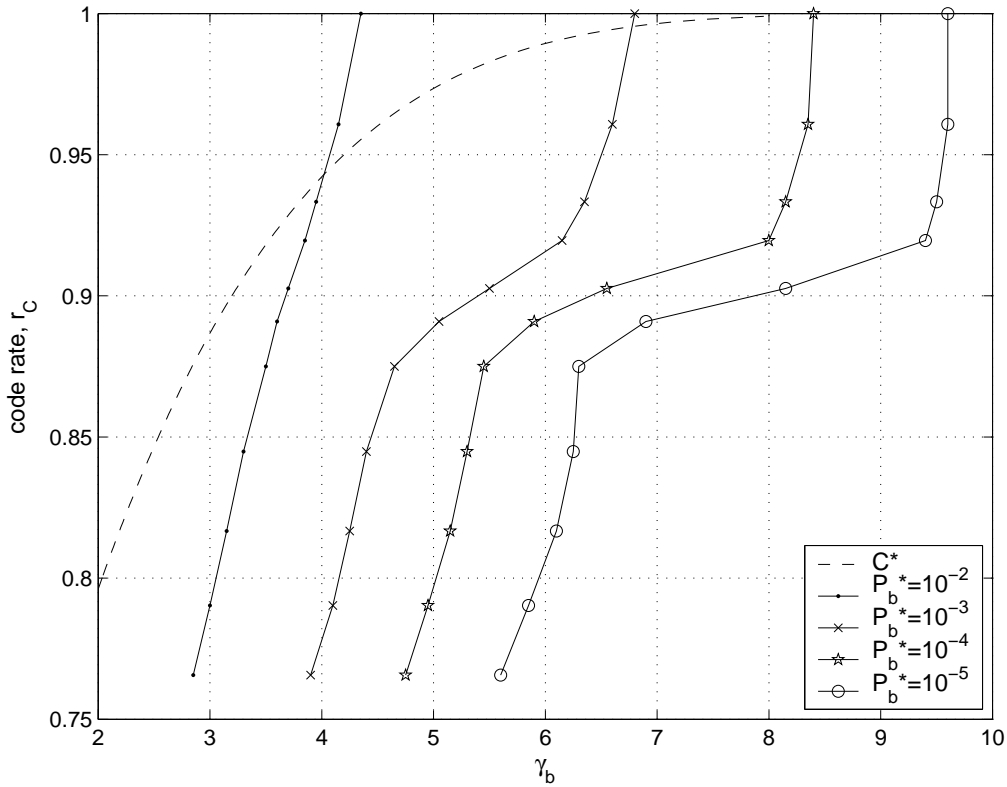


Figure 6.26. The γ_b -values required for convergence to P_b^* as a function of the code rate, r_c for a 2D SCSPC(8,7) code that uses optimal values of Δ . As a reference C^* is also plotted.

The optimal values of Δ for different code rate are plotted in Figure 6.27 for $P_b^* = 10^{-5}$. As can be seen, it is preferable to puncture some systematic bits for medium code rates. Also note that the different dimensions in a serially concatenated system are not identical, since each dimension adds more redundancy than the previous. The systematic part is represented by δ_0 , the outermost code, C_1 is represented by δ_1 and finally the innermost code C_2 is represented by δ_2 . Hence, we can determine from Figure 6.27 that puncturing should start with the parity bits in the first dimension, corresponding to the outermost code.

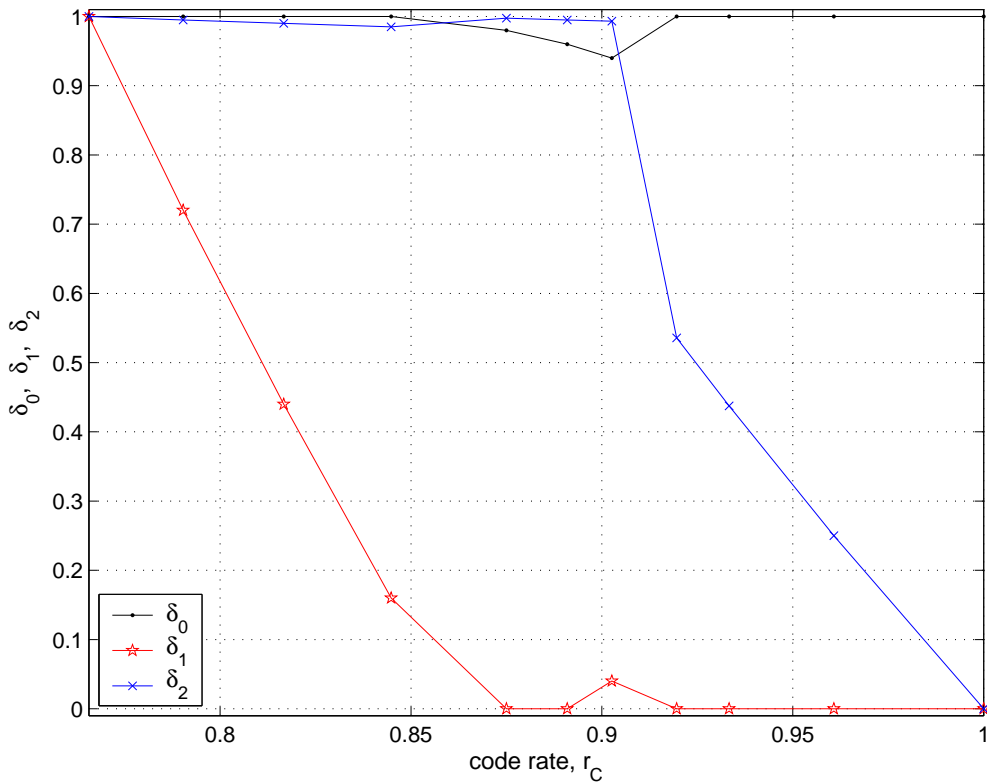


Figure 6.27. Optimal puncturing ratios for the 2D SCSCP(8,7) code when the target BER is 10^{-5} .

In Figure 6.28 the γ_b is again plotted as a function of the code rate, but for both a 2D and a 3D serially concatenated SPC(8,7) code. The curves for the 3D codes go all the way down to rate $(7/8)^3$, whereas the 2D codes stop at $(7/8)^2$. The performance for a 3D code is again much better than, and at worst equally good as a 2D code. Only when we have code rates close to one, is the performance of the 2D and 3D codes equal. Also note that puncturing decreases the γ_b required for convergence for low code rates. Punctured serially concatenated SPC codes generally perform better than punctured parallel concatenated SPC codes when optimal values of Δ are used.

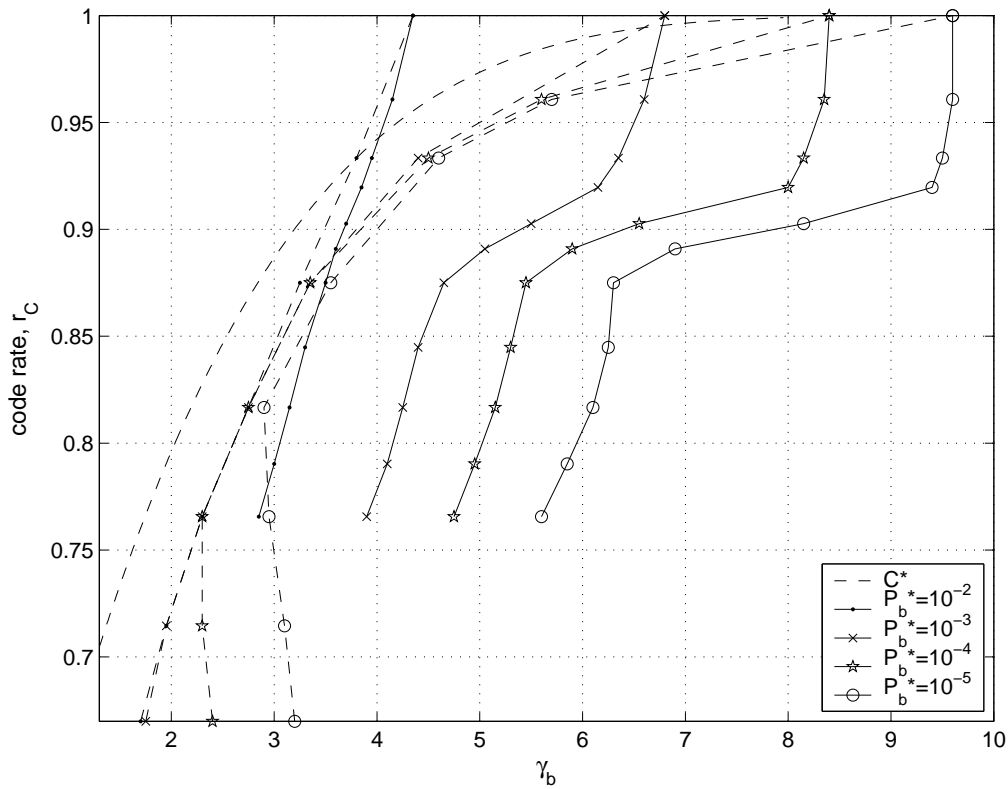


Figure 6.28. The γ_b -values required for convergence to P_b^* as a function of the code rate r_c for a 2D (solid lines) and a 3D (dashed lines) SCSPC(8,7) code that uses optimal values of Δ . C^* is plotted as a reference.

In Figure 6.29 the optimal values of Δ for different code rates are plotted for $P_b^* = 10^{-2}$. It is still preferable to puncture systematic bits for low to medium code rates, but more importantly, the innermost code should be punctured last. As soon as all the systematic bits are restored for higher code rates, dimension-wise puncturing of parity bits starting with the outermost code is best.

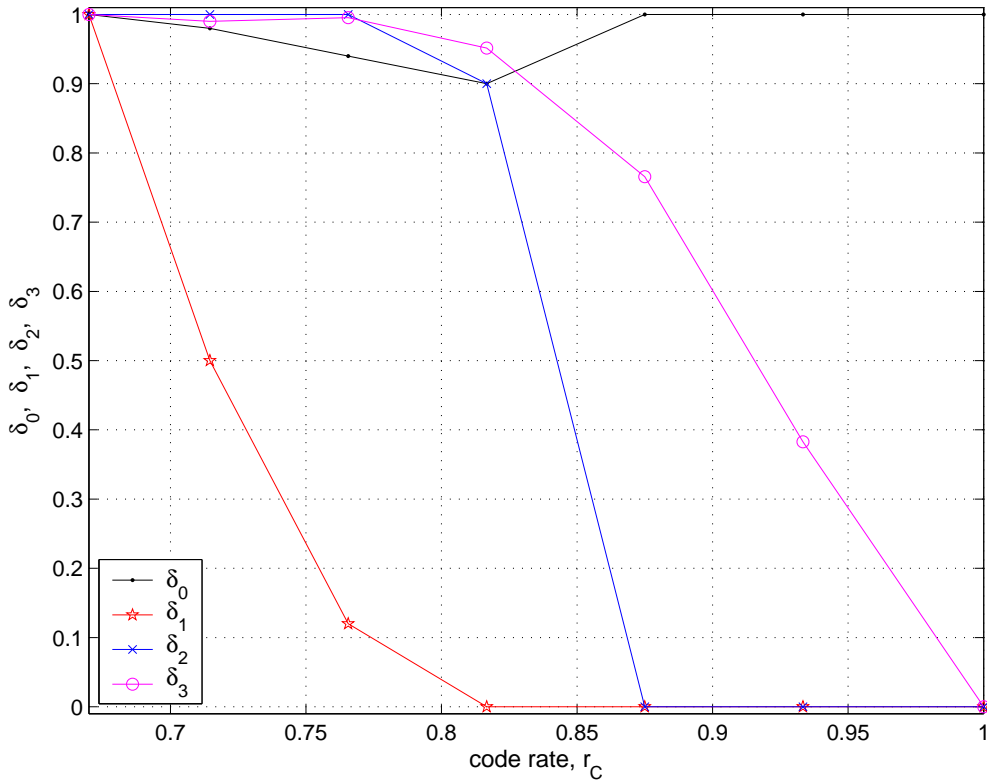


Figure 6.29 Optimal puncturing ratios for the 3D SCSCP(8,7) code when the target BER is 10^{-2} .

In Figure 6.30 the optimal values of Δ for different code rates are plotted, but now for $P_b^* = 10^{-5}$. In this case, even more systematic bits are punctured. Recall that puncturing of systematic bits should be avoided if we want to guarantee the local invertibility regardless of which interleavers are used. Hence, we consider the scenario when searching for optimal puncturing ratios while not allowing systematic bits to be punctured. The result is reported in Figure 6.31. As can be seen the optimal puncturing ratio with no constraints performs much better than the puncturing ratio not allowing systematic bits to be punctured. The performance of a 2D SCSPC code using optimal puncturing ratios is also plotted as a reference. For medium code rates, when the 2D code punctures systematic bits, it is superior to the corresponding 3D code. For high and low code rates, however, the 3D scheme is superior. The optimal puncturing ratios, under the constraint that $\delta_0 = 1$, are found to be dimension-wise puncturing starting from the outermost code.

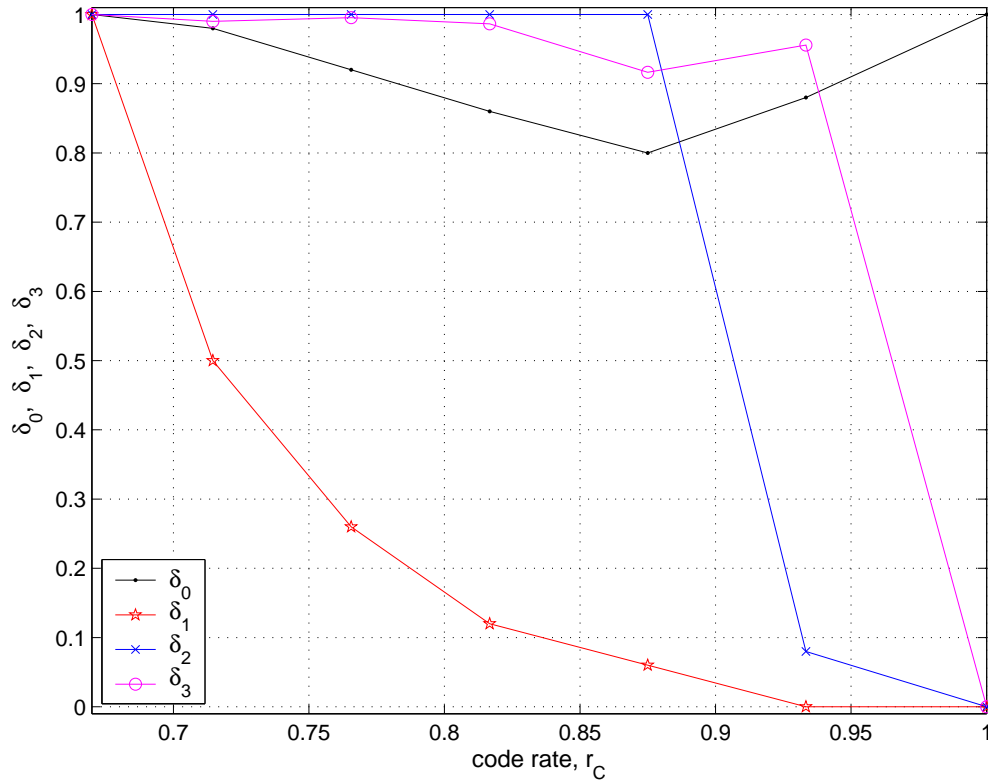


Figure 6.30. Optimal puncturing ratios for the 3D SCSCP(8,7) code when the target BER is 10^{-5} .

Rate compatible puncturing implies that going from a low code rate to a higher code rate, the Δ -values are not allowed to increase. Dimension-wise puncturing starting from the outermost code, which we term backwards dimension-wise puncturing, implies that the Δ -values are as reported in Figure 6.32. This, in turn implies that a scheme using this pattern starts transmitting parity bits from the innermost code until all its parity bits have been sent. Thereafter, transmission continues with the intermediate code, closest to the innermost code.

Backwards dimension-wise puncturing was found to be optimal for serially concatenated SPC codes under the constraint that $\delta_0 = 1$. This seems reasonable since the inner codes contain enough information to activate the outermost decoder regardless of if its parity bits are punctured or not. This implies, however, that the decoding complexity is not reduced for this puncturing pattern since all decoders have to be activated as soon as parity bits pertaining to the innermost decoder arrive. For parallel concatenated SPC codes, dimension-wise puncturing reduces the decoding complexity since only the component decoders that it has received parity bits for so far needs to be activated. Hence, even if the mother code is based on three component codes concatenated in parallel, only one or two component decoders may need to be activated for the first few transmissions.

A puncturing pattern that does reduce complexity for serial codes is dimension-wise

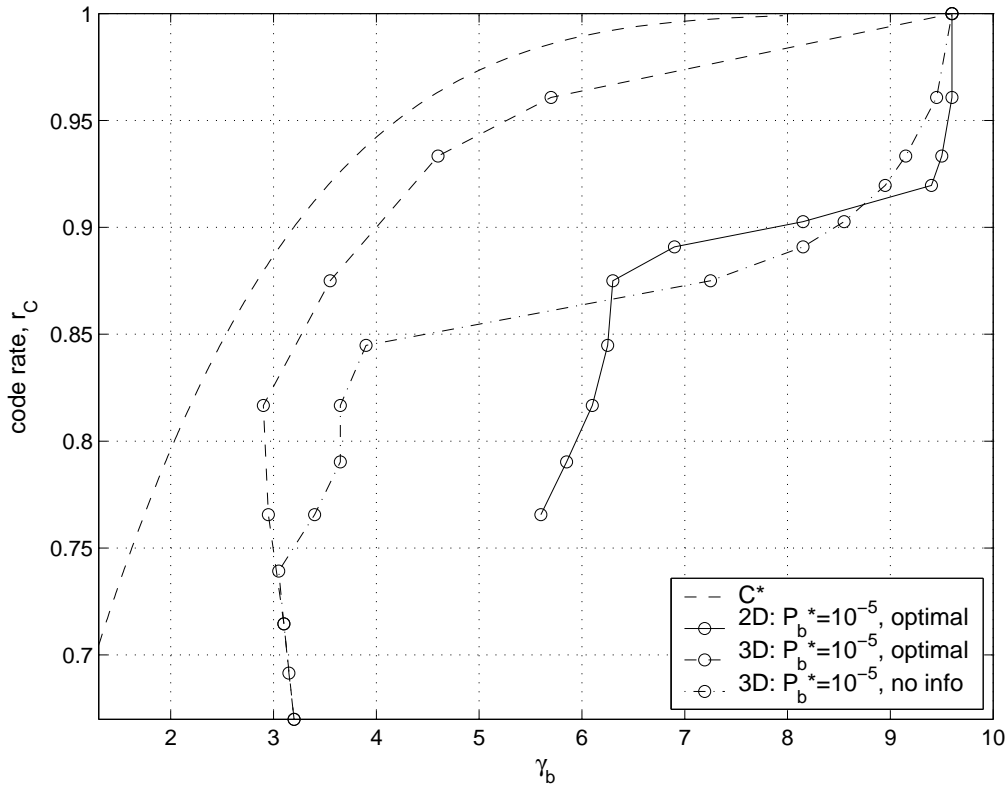


Figure 6.31. The γ_b -values required for convergence to P_b^* as a function of the code rate r_c for a 2D (solid line) and a 3D SCSPC(8,7) code that uses optimal values of Δ (dashed line) and optimal values of Δ with the constraint that $\delta_0 = 1$ (dash-dotted line).

puncturing starting with the innermost code, as depicted in Figure 6.33. Therefore, it is interesting to compare the performance between the two alternative rate compatible puncturing strategies for serially concatenated SPC codes. In Figure 6.34, γ_b is plotted as a function of the code rate, for a 3D SCSPC(8,7) code using both dimension-wise puncturing and backwards dimension-wise puncturing. Dimension-wise puncturing for a 3D PCSPC(8,7) code is also plotted as a reference. For high code rates the serial and the parallel code using dimension-wise puncturing are the same since the outermost code is identical. For lower code rates the serial code is superior. Best performance does, however, the serial system using backwards dimension-wise puncturing provide. However, this is performance in terms of γ_b required for convergence to a target BER. Performance in terms of decoding complexity is quite poor for backwards dimension-wise puncturing since all component decoders has to be activated as soon as the first transmission arrives. Consequently, the puncturing pattern should be chosen according to the specific application requirements.

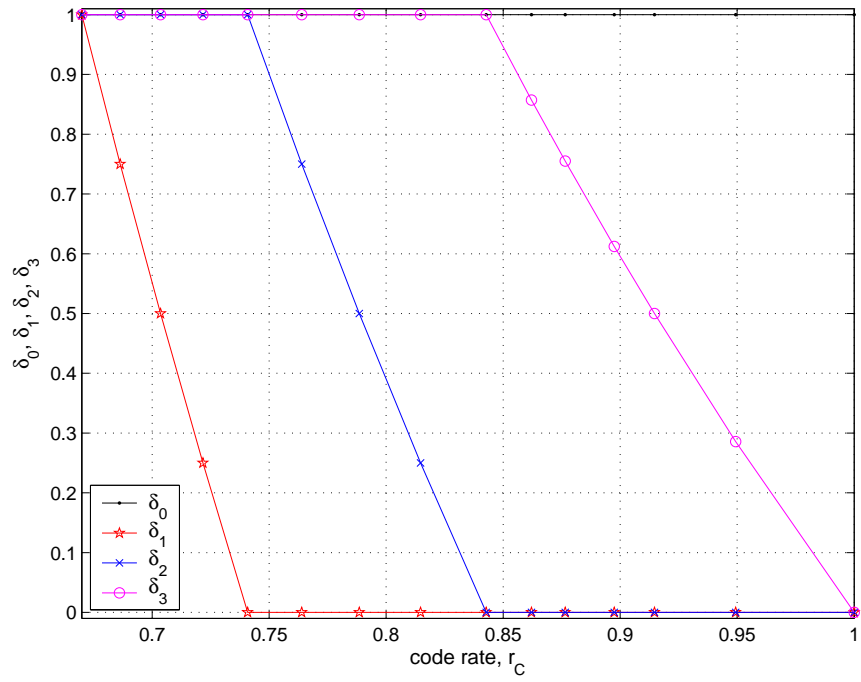


Figure 6.32. Backwards dimension-wise puncturing.

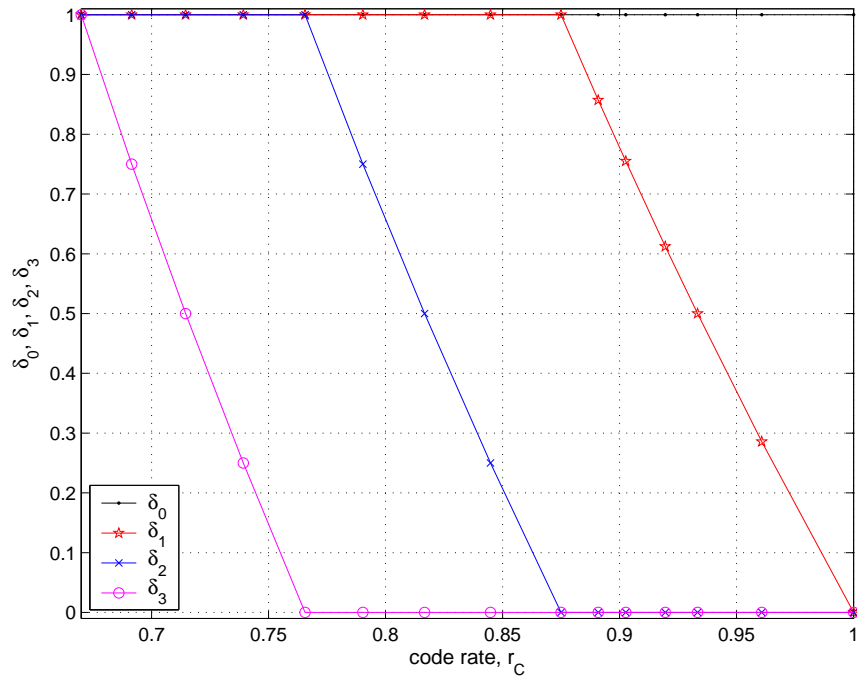


Figure 6.33. Dimension-wise puncturing.

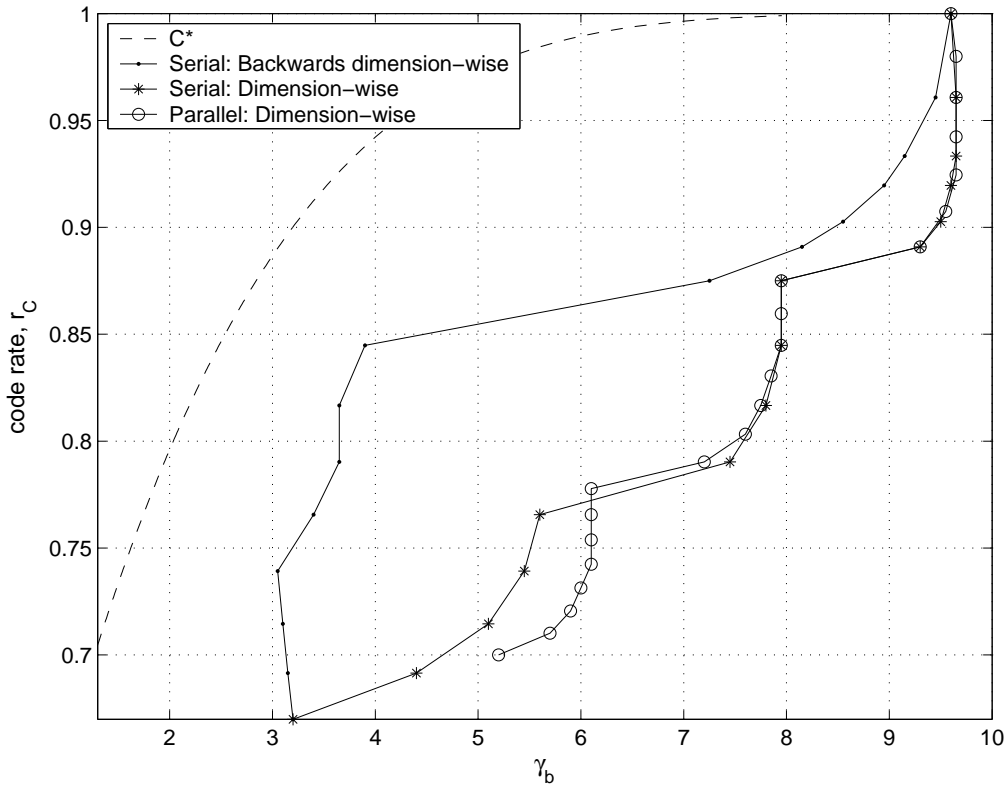


Figure 6.34. The γ_b -values required for convergence to $P_b^* = 10^{-5}$ as a function of the code rate r_c for 3D serial codes that uses dimension-wise and backwards dimension-wise puncturing. The corresponding 3D parallel code using dimension-wise puncturing is also plotted as a reference.

The same problem that occurred when searching for good rate compatible puncturing patterns using bounds also occurs when searching for good rate compatible puncturing ratios using EXIT charts. The chosen values of the code rates, i.e. the particular rate compatible code family chosen, are very important. If we choose a different set of code rates in the EXIT chart search that may result in different rate compatible puncturing ratios. For example if $r_c = 1$ is not included in the search, and we instead want to optimize the rate compatible puncturing ratios when the first packet contains 49 parity bits, this correspond to $r_c = 0.875$. According to e.g., Figure 6.30 the Δ -values for $0.67 \leq r_c \leq 0.875$ are rate compatible since they are all decreasing. Hence, rate compatible puncturing ratios including punctured systematic bits would now instead be obtained. Therefore dimension-wise puncturing is only guaranteed to be the best for the rate compatible code family including all possible rates.

However, when optimal packet lengths are obtained for a particular rate compatible code family, given a specific puncturing pattern, the EXIT chart search may be used to examine if there is a better rate compatible puncturing pattern for this particular set of code rates.

Chapter 7

Performance Results

The bounds on FER, using dimension-wise puncturing patterns obtained in Chapter 5, are used to optimize the packet lengths in an IR-HARQ scheme to maximize the average code rate. Performance results are presented for both parallel and serially concatenated SPC codes. It is concluded that the more retransmissions allowed, the higher average code rate can be obtained. Hence, transmitting one code bit at a time yields an achievable upper bound on the average code rate. This way each information frame is accepted following the transmission of a minimum number of parity bits. The analytical results are also compared to Monte-Carlo simulations using the optimal packet lengths and low complexity puncturing patterns. The average code rate obtained in the simulations corresponds well with the average code rate obtained analytically.

The dimension-wise puncturing is chosen for these examples and for serially concatenated codes the puncturing starts with the innermost code. These puncturing patterns have been shown in previous chapters to be good rate compatible puncturing patterns, but mainly they have the benefit of reducing the decoder complexity since only those constituent decoders for which parity bits have been received needs to be activated.

7.1 Parallel Concatenated Scheme

We first consider the 3D PCSPC (8,7) example code with $k = 7^3$ as the mother code. Assume that this mother will provide the required FER at an acceptable SNR. The lowest possible code rate a scheme based on this mother code can have is then $r_{\text{C}_b} = k/n = 0.7$ and $n - k = 49 + 49 + 49 = 147$. For simplicity, and in order to get a lower bound on the achievable performance, we assume PED. Hence, $P_{ARQ}(n_i)$ is equal to the FER after transmission i . Since we have PED, the FER in accepted frames is zero and the number of rejected frames following the last allowed transmission determines the total FER of the IR-HARQ scheme. Dimension-wise puncturing is applied and bounds on FER from Chapter 5

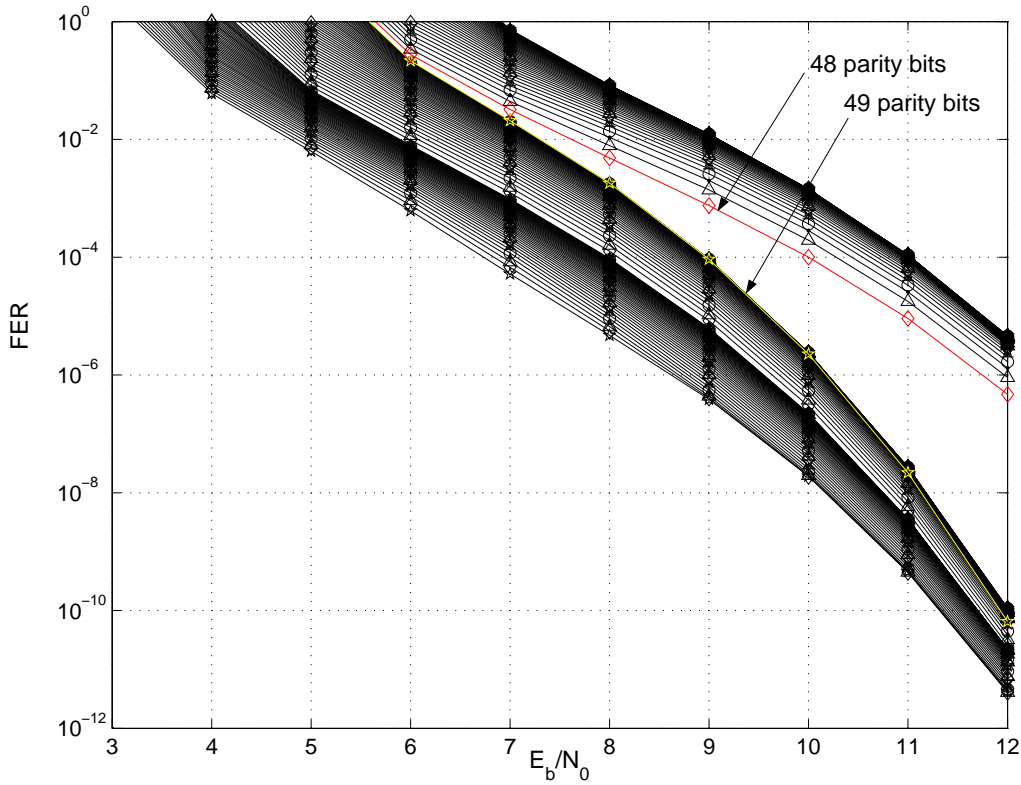


Figure 7.1. FER versus E_b/N_0 obtained using bounds for different packet lengths of a 3D PCSPC(8,7) mother code. Dimension-wise puncturing is used.

are obtained for every possible code rate from 0.7 up to one as shown in Figure 7.1.

Essentially, all the curves in Figure 7.1 are grouped in three groups. In the upper group, ending with the curve marked ‘48 parity bits’, all curves have $d_{min} = 1$ since they still contain some “uncoded” component codewords. As soon as all the parity bits from the first dimension are included, $d_{min} = 2$ and we see that as the second group beginning with the grey curve marked ‘49 parity bits’. Up to 7 dB the grey curve marked ‘49 parity bits’ has the worst performance in the second group (i.e., among the curves that include bits from the second dimension), but at e.g., 12 dB some of the curves in the second group are worse. This is because d_{min} is the dominating factor for high SNR and since all the curves in the second group have $d_{min} = 2$ the ‘49 parity bits’-curve is no longer the worst since it uses the least amount of redundancy. The redundancy introduced by the other curves in the second group is not used to increase d_{min} , but it does reduce the multiplicity of codewords at d_{min} and consequently they are better at low SNR.

Based on these bounds we can plot the FER as a function of the packet length for a fixed E_b/N_0 according to Figure 7.2. Note that these results only apply to the specific puncturing pattern chosen. A different puncturing scheme would result in a different set of bounds in Figure 7.1. However, given this particular puncturing pattern, Figure 7.2 shows the FER as a function of the number of parity bits for different E_b/N_0 . Note that since we

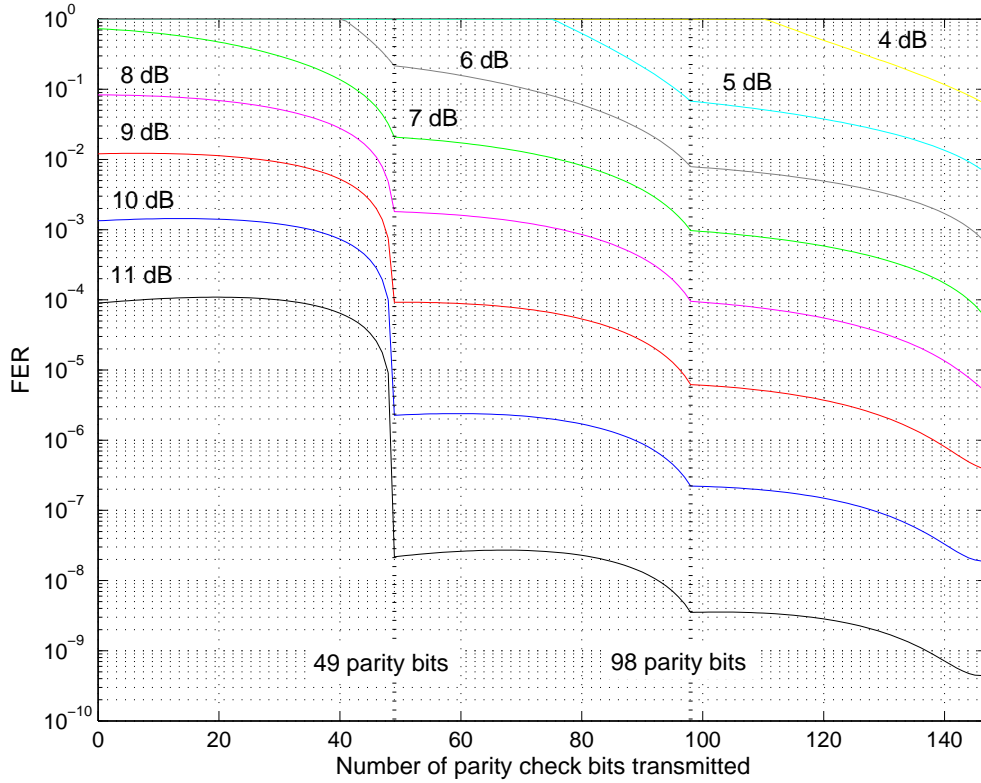


Figure 7.2. FER for 3D PCSPC(8,7) for different E_b/N_0 as a function of the number of parity bits used in the decoding process.

use PED $P_{ARQ}(n_i)$ corresponds to the FER.

Variations in code strength are observed as the number of parity bits increases in each dimension. At the boundaries between dimensions we observe significant differences in the rate of improvement. This corresponds to the three groups in Figure 7.1. Depending on which parity bits are transmitted, the FER is affected in different ways. For example, the first few bits from a new dimension in our concatenated SPC code have less impact than the last few from the same dimension. This is true for every dimension. The values 49 and 98 marked with dotted lines in Figure 7.2 represent all the bits from the first and second dimension respectively. Looking at the FER versus packet lengths at e.g., 7dB and 11 dB, they do not have the same characteristics. At 7 dB it is always better to include more redundancy, whereas at 11 dB, that may not be the case. Looking closely at the curve for 11 dB, we can see that the FER is actually increasing when more parity bits are included in the beginning of the new dimension, before it starts to reduce again towards the end of the dimension. This is in accordance with the bounds in Figure 7.1.

Now we want to solve the maximization problem in (3.10), with constraints on c_i and c_M given in (3.9). For $M=2$ this is a one-dimensional optimization problem. An exhaustive search yields the optimal value for c_1 , which at $E_b/N_0 = 5$ dB is 94 parity bits resulting in $r_C^{M=2} = 0.7731$. This implies that c_1 should contain all of the 49 parity bits from

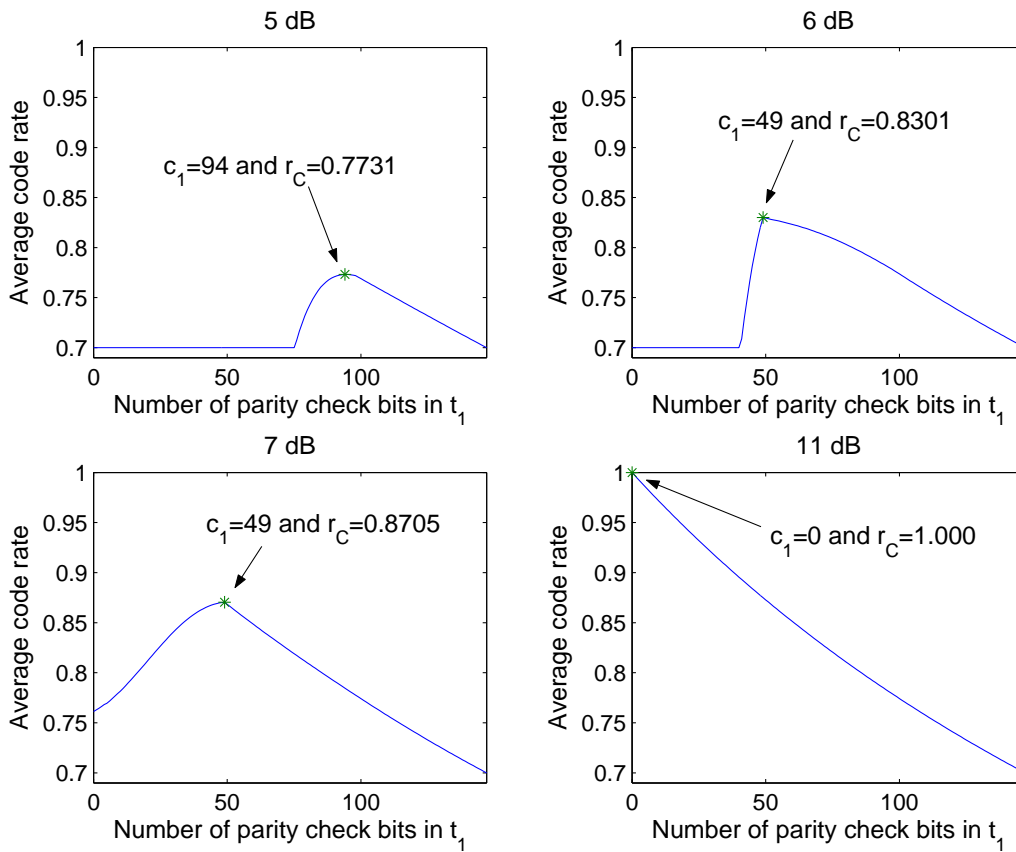


Figure 7.3 Average code rate versus the number of parity bits in c_1 for $M = 2$ at $E_b/N_0 = 5, 6, 7$ and 11 dB.

the first dimension and any 45 parity bits of the 49 available in the second dimension. In Figure 7.3 the optimal values of c_1 are shown for $E_b/N_0 = 5, 6, 7$ and 11 dB.

For $M = 3$ the optimization problem is two-dimensional, since the average code rate is a function of c_1 and c_2 as shown in Figure 7.4 for $E_b/N_0 = 5$ dB. Note that the function is only defined in the area where $c_1 + c_2 \leq 147$, i.e., smaller or equal to the maximum number of parity bits, $n - k$. Also it is interesting to see that the first portion of the function is flat, which is likely to be due to the fact that the FER is equal to 1 for many packet lengths at 5 dB.

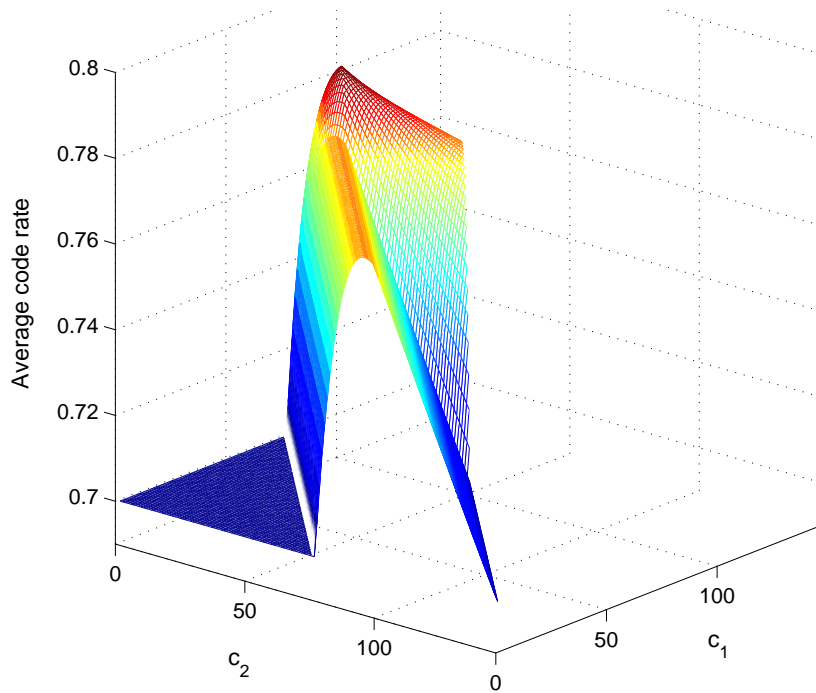


Figure 7.4. Average code rate versus c_1 and c_2 for $M = 3$ at $E_b/N_0 = 5$ dB.

For $M = 1, 2, 3, 4$ the optimization may be accomplished by an exhaustive search. However, when $M > 4$ an exhaustive search is too complex and hence the Nelder-Mead simplex search method, [105] is used for solving the optimization problem.

Figure 7.5 shows the maximum average code rate as a function of E_b/N_0 for IR-HARQ schemes with $M = 1, 2, 3, 4, 5, 6, 7, 148$. The maximum average code rate is obtained by solving the maximization problem in (3.10), with constraints on c_i and c_M given in (3.9) for case 1. Case 1 represent that we always send all the available parity bits in the end if necessary, and hence the last transmission will always contain the remaining party bits of the mother code. $M = 148$ is obtained by setting $c_1 = 0$ and $c_i = 1$ for $i = 2, 3, \dots, n - k + 1$.

A non-ARQ system using the full-rate mother code is also shown as a reference in Figure 7.5. Note that the non-ARQ scheme has the same performance as an IR-HARQ scheme with $M = 1$ when case 1 is considered. We see that the maximum average code rate increases with M . If $M = n - k + 1 = 148$, we get $r_c^{M=148} = 0.7992$ for $E_b/N_0 = 5$ dB. We conclude that allowing a large number of IR transmissions leads to a finer resolution of code rates, facilitating the task of only transmitting the required amount of redundancy. Hence the optimal choice of M is to set it to the maximum value allowed. The corresponding average code rate is an achievable upper bound for $1 \leq M \leq n - k + 1$.

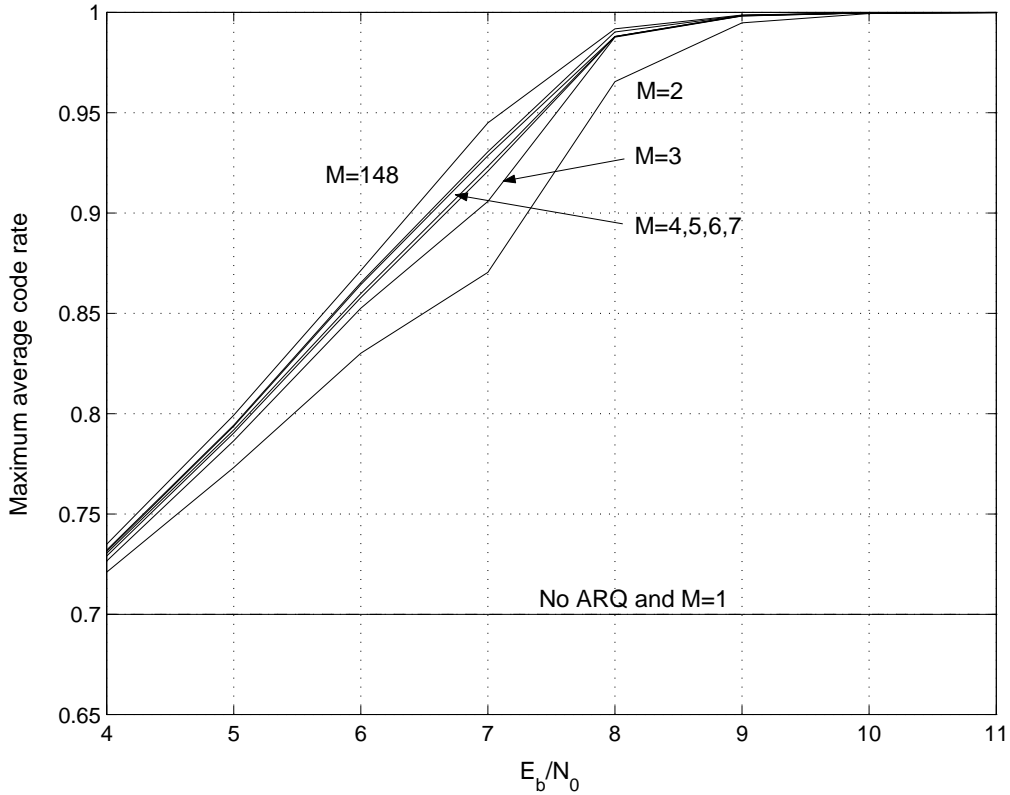


Figure 7.5. Maximum average code rate as a function of E_b/N_0 for different M , based on the 3D PCSPC(8,7) mother code. Here c_M always contains the remaining parity bits of the mother code, a.k.a., case 1.

To accommodate an error rate performance of $\text{FER} = 10^{-2}$ for a wide range of SNR, case 2 is used, i.e., letting c_M contain only the remaining parity bits needed to provide the target FER. We now get the maximum average code rate as a function of E_b/N_0 according to Figure 7.6. The curves plotted represent IR-HARQ schemes with $M = 1, 2, 3, 4, 5, 6, 7, 148$ obtained by solving the maximization problem in (3.10), with the constraints in (3.11). Note that the performance of the non-ARQ system remains the same but the performance when $M = 1$ is now improved since less redundancy is needed when E_b/N_0 increases. It can be seen that the gain for $M = 1$ and $M = 2$ is considerable as compared to Figure 7.5. As M increases, the gain diminishes and for $M = 148$, it is the same average code rate as for case 1.

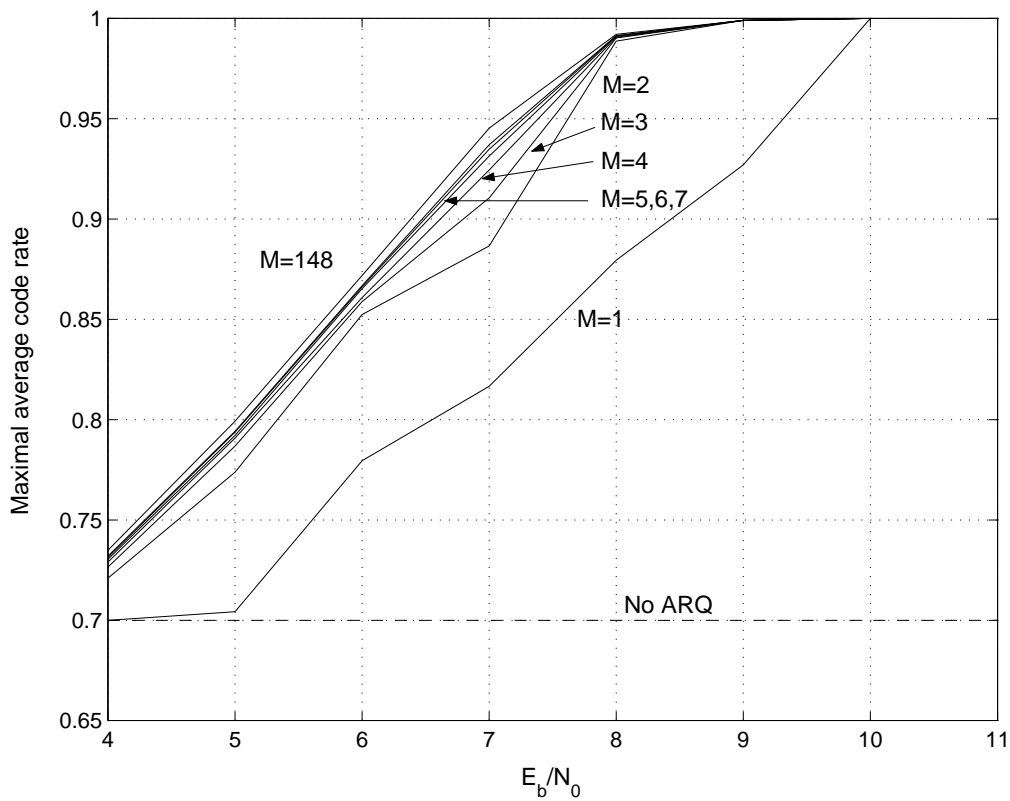


Figure 7.6. Maximum average code rate as a function of E_b/N_0 for different M , based on the 3D PCSPC(8,7) mother code. Here c_M only contains enough parity bits to obtain a FER of 10^{-2} , corresponding to case 2.

When considering case 2, but this time with a target FER of 10^{-3} , Figure 7.7 is obtained. Note that the QoS level of $P_t = 10^{-3}$ cannot be guaranteed until around 6 dB with this mother code. The horizontal dotted lines denote that the code rate is crossing a code dimension, which may explain the irregular behaviour of the code rate curve at these values.

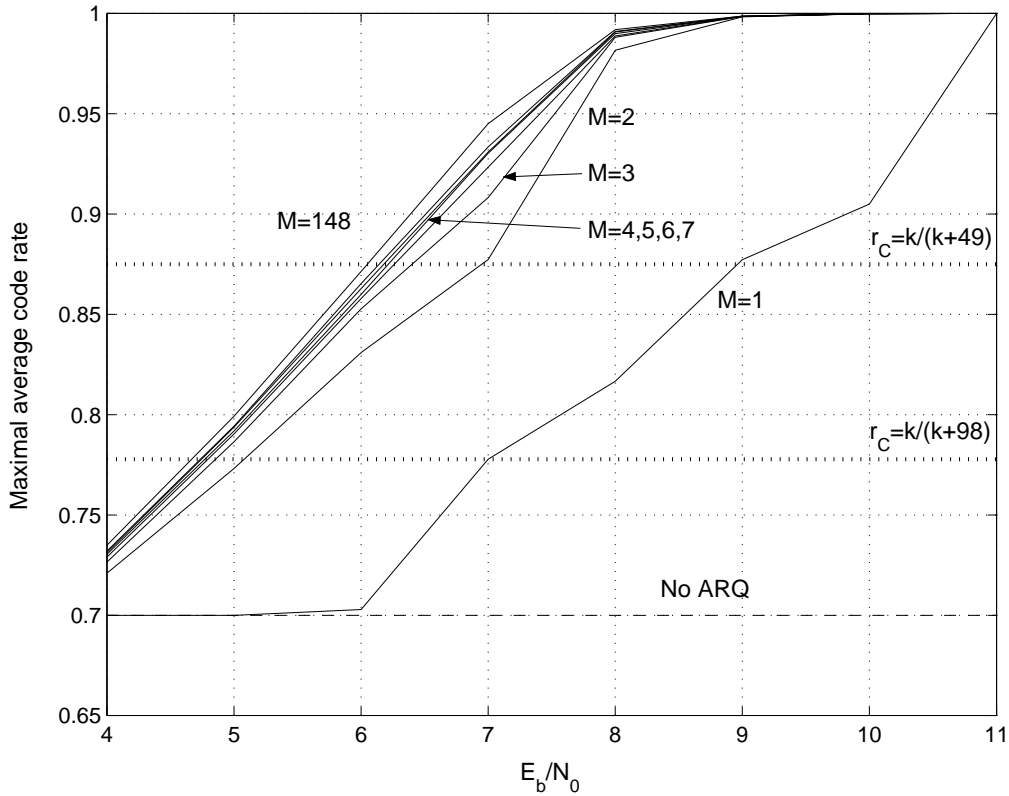


Figure 7.7. Maximum average code rate as a function of E_b/N_0 for different M , based on the 3D PCSPC(8,7) mother code. Here c_M only contains enough parity bits to obtain a FER of 10^{-3} .

We consider two different example scenarios and simulate the performance obtained when optimal packet lengths are used in the IR-HARQ scheme. The first scenario considers case 1 with $M = 7$ and the packet lengths as reported in Table 7.1. Consequently, the simulation uses different packet lengths for different E_b/N_0 and should result in the corresponding calculated average code rate. In Figure 7.8 the performance in terms of FER obtained through simulation for a scenario with $M = 7$ using case 1 is depicted. The different curves in Figure 7.8 represent the FER performance if the IR-HARQ system were to be truncated after transmission t_i , where $i = 1, 2, \dots, M$. The FER of the IR-HARQ system with $M = 7$, marked ‘After $t_{M=7}$ ’ in Figure 7.8, should follow the performance of the mother code since case 1 is considered. Therefore, the FER of the mother code is plotted as a reference in Figure 7.8. However, it can be seen that the IR-HARQ scheme has better performance than the mother code in terms of FER after allowing M transmissions.

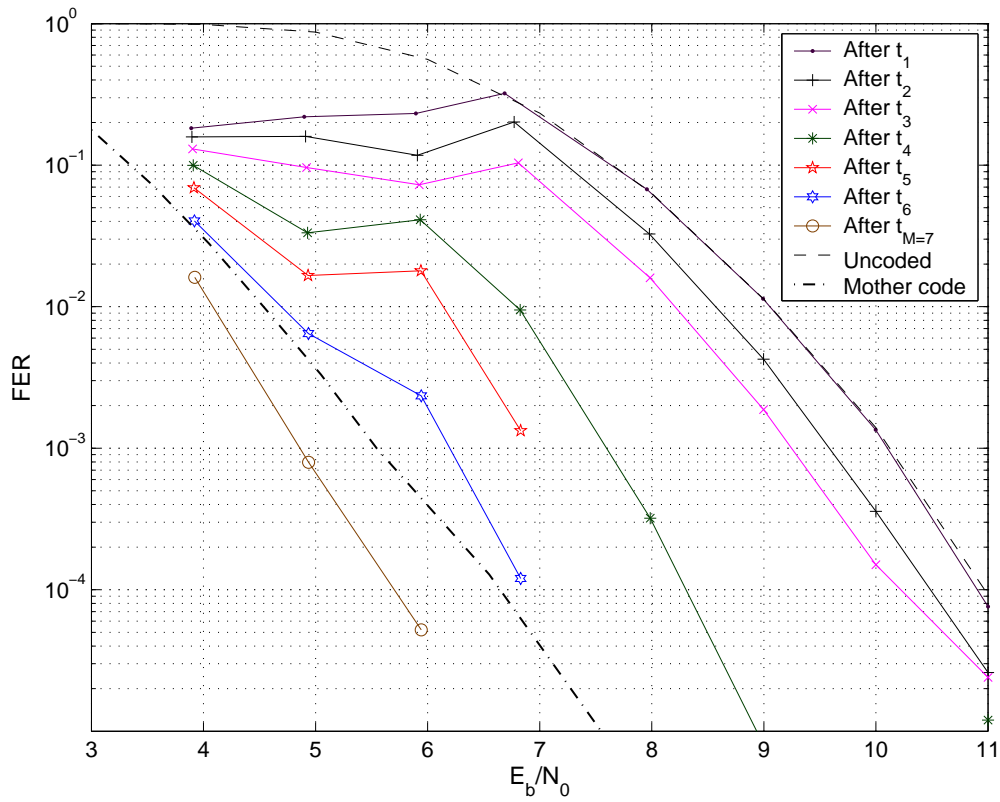


Figure 7.8. Simulated performance of a QoS case 1 with $M=7$ for the mother code 3D PCSPC(8,7).

Table 7.1. Optimal packet lengths for QoS case 1 with $M=7$ using the 3D PCSPC(8,7) mother code and dimension-wise puncturing.

E_b/N_0	r_C	n_1	n_2	n_3	n_4	n_5	n_6	n_7
4 dB	0.7320	457	460	464	469	475	482	490
5 dB	0.7942	422	427	433	441	457	473	490
6 dB	0.8653	387	392	405	416	427	441	490
7 dB	0.9305	343	364	379	392	426	441	490
8 dB	0.9902	343	369	381	392	467	478	490
9 dB	0.9985	343	374	384	392	424	441	490
10 dB	0.9998	343	378	386	392	469	479	490
11 dB	1.0000	343	380	383	387	390	392	490

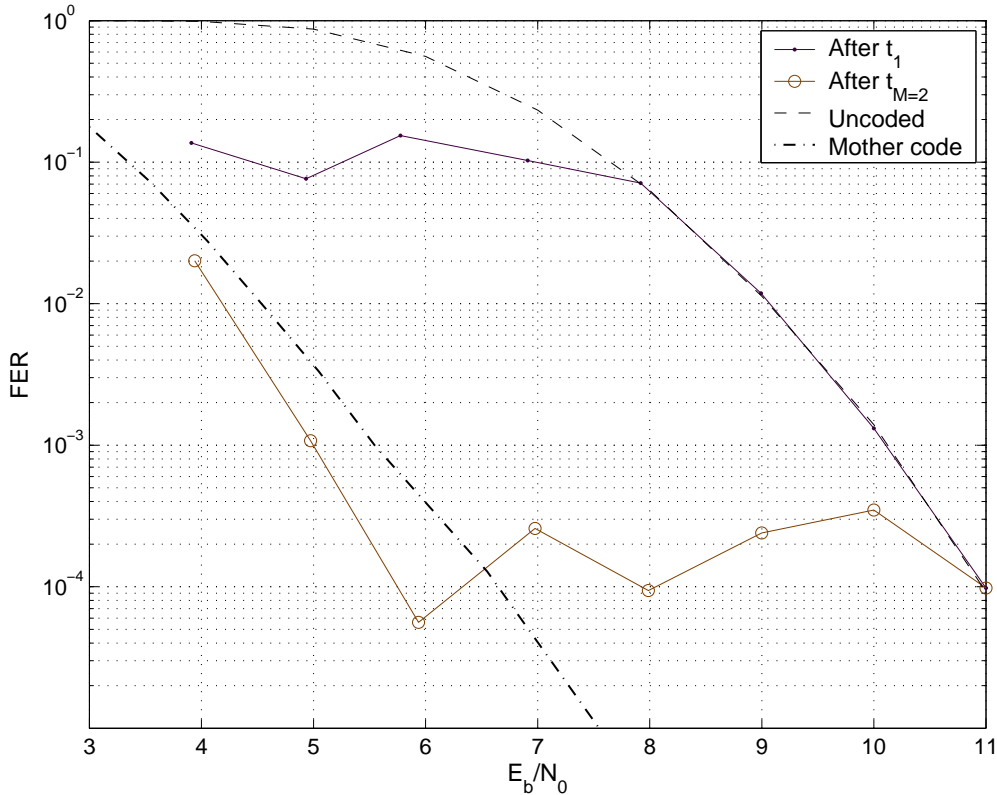


Figure 7.9. Simulated performance of a QoS case 2 with $M=2$ and FER 10^{-3} for the mother code 3D PCSPC(8,7).

This is due to the fact that the corresponding average code rate of the IR-HARQ scheme is indeed higher than the code rate of the mother code used in a non-ARQ system. All M transmissions are not always required. Since the abscissa denotes E_b/N_0 rather than E_s/N_0 , the IR-HARQ scheme is better.

The second scenario considers case 2 with $M = 2$ and FER of 10^{-3} . The corresponding optimal packet lengths are reported in Table 7.2. In Figure 7.9 the simulated performance in terms of FER is given. The FER performance of the simulated IR-HARQ scheme should follow the mother code until the FER reaches 10^{-3} . Thereafter the performance should settle at a FER of 10^{-3} until E_b/N_0 is sufficiently high so that the uncoded case provides a FER of 10^{-3} . Thereafter the performance should instead follow the uncoded case. According to Table 7.2, the full mother code is used in the last transmission until 6 dB. We can see from Figure 7.9 that the performance of the IR-HARQ scheme is better than the mother code for 4 and 5 dB, since all transmissions are not always required. For 6 dB, when the final transmission no longer contains the full mother code, the performance of the IR-HARQ schemes increases further as compared to the mother code. For 7-10 dB the performance remains relatively stable at a level somewhat lower than 10^{-3} . Since $M = 2$, there are only two available transmissions. For a higher M we would expect to achieve a more stable level

Table 7.2. Optimal packet lengths for QoS case 2 with $FER=10^{-3}$ and $M=2$ using the 3D PCSPC(8,7) mother code and dimension-wise puncturing.

E_b/N_0	r_C	n_1	n_2
4 dB	0.7210	466	490
5 dB	0.7731	437	490
6 dB	0.8310	392	488
7 dB	0.8773	383	441
8 dB	0.9817	343	420
9 dB	0.9983	343	391
10 dB	0.9999	343	379
11 dB	1.0000	343	343

once the target FER has been reached. Also, we generally get a lower FER than requested, which may be caused by the bounds being a bit pessimistic.

In Figure 7.10 the simulated average code rates for the two scenarios are compared to the calculated maximum average code rate. We can see that the average code rate obtained using simulations corresponds well with the average code rate obtained analytically. In most cases the actual average code rate is slightly higher than the one obtained analytically.

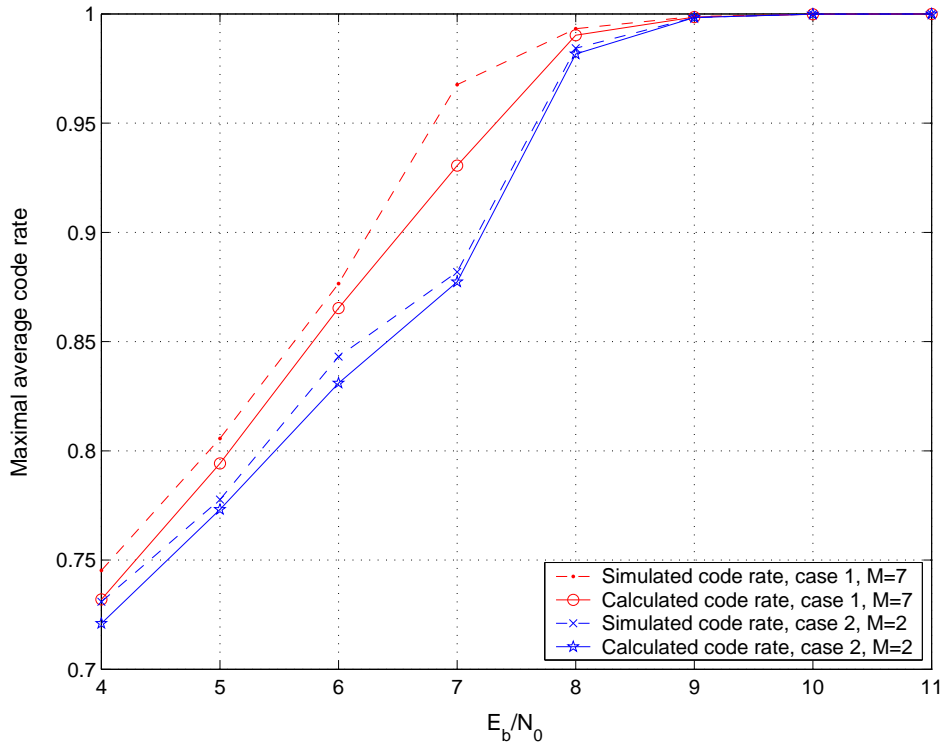


Figure 7.10. Maximum average code rate as a function of SNR for $M=7$ case 1 and $M=2$ case 2, respectively. Simulations are compared to calculations.

7.2 Serially Concatenated Scheme

In the serial case we consider the 3D SCSPC (8,7) example code with $k = 7^3$ as the mother code. The lowest possible code rate a scheme based on this mother code can have is $r_{C_{3d}} = k/n = 0.67$. Dimension-wise puncturing, starting with the innermost code is applied. Recall that this puncturing pattern was not the optimal one as determined by the EXIT chart analysis, but it was the one minimizing the decoder complexity since the innermost decoders need not be activated until their parity bits have been received. The bounds on FER from Chapter 5 are obtained for every possible code rate from 0.67 up to one as shown in Figure 7.11. All the curves in Figure 7.11 are grouped in three groups. The upper group, together with the curve marked '49 parity bits', are identical to the PCSPC(8,7) code. Thereafter, they differ from the second dimension where the serial code adds 56 parity bits as opposed to the 49 parity bits added by the parallel code. For the serial mother code we have $n - k = 49 + 56 + 64 = 169$. Thus, Figure 7.11 contains 170 curves in total, whereas Figure 7.1 only contains 148.

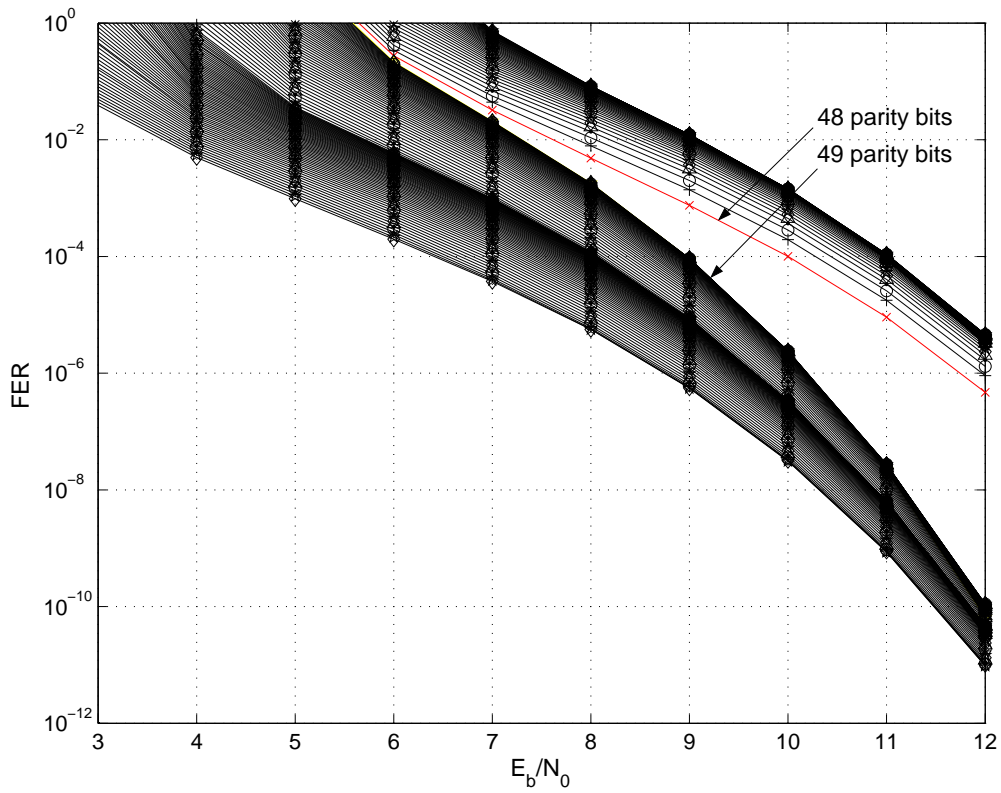


Figure 7.11. FER versus E_b/N_0 obtained using bounds for different packet lengths of a 3D SCSPC(8,7) mother code. Dimension-wise puncturing starting with the innermost code is used.

Based on these bounds we can plot the FER as a function of the packet length for a fixed E_b/N_0 according to Figure 7.12. Again note that these results only apply to the specific puncturing pattern chosen. At the boundaries between dimensions we observe significant differences in the rate of improvement, similar to the parallel case.

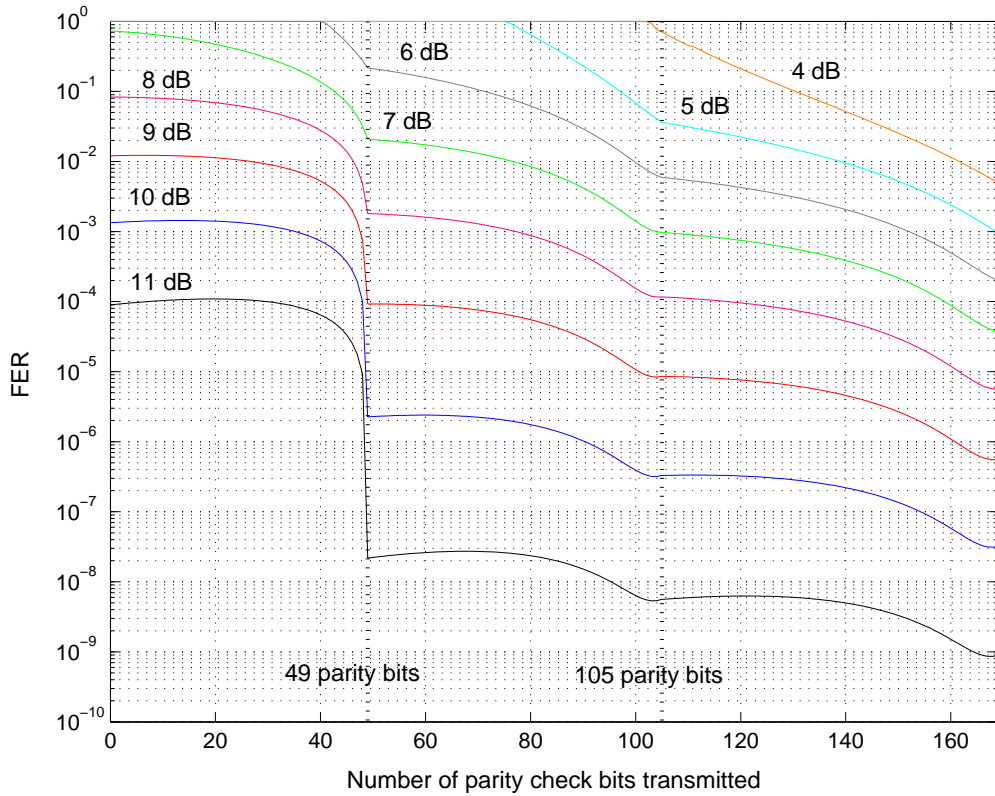


Figure 7.12. FER for the 3D SCSPC(8,7) code for different E_b/N_0 as a function of the number of parity bits used in the decoding process.

By solving the maximization problem in (3.10) using the Nelder-Mead simplex search method, with the constraints in (3.9), i.e., case 1, the optimal packet lengths and their corresponding code rates are found. In Figure 7.13 we have plotted the maximum average code rate as a function of E_b/N_0 for IR-HARQ schemes with $M = 1, 2, 3, 4, 5, 6, 7, 170$. We observe two regions where the slope of the maximum average code rate curve decreases for increasing E_b/N_0 . These regions correspond to the regions in Figure 7.12 of slow decrease of the FER for decreasing code rate. We also observe typically diminishing returns for an increasing maximum number of transmissions. For $M = 4$ the maximal average code rate is around 0.25 dB away from the upper bound on the maximal average code rate represented by allowing $M = 170$ transmissions.

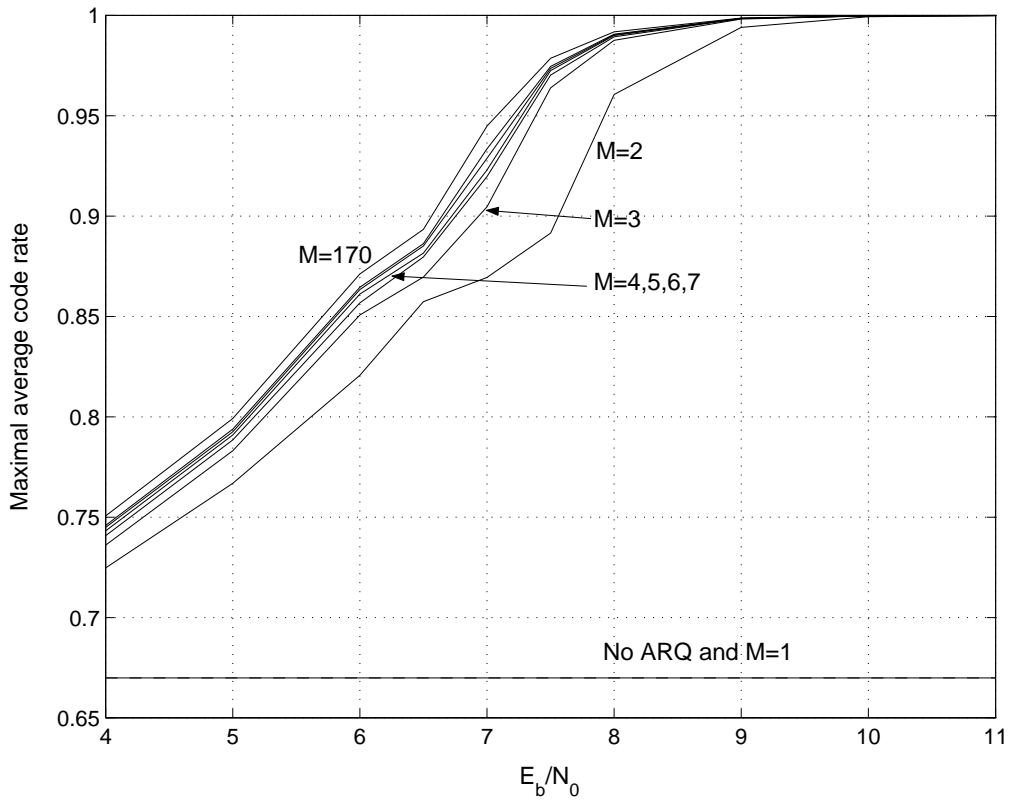


Figure 7.13. Maximum average code rate as a function of E_b/N_0 for different M , based on the 3D SCSPC(8,7) mother code. Case 1: c_M always contains the remaining parity bits of the mother code.

To accommodate an error rate performance of $\text{FER} = 10^{-2}$ for a wide range of SNR, case 2 is used and thus we let c_M contain only the remaining parity bits needed to provide a target FER of 10^{-2} . We now get the maximum average code rate as a function of E_b/N_0 according to Figure 7.14. The curves plotted represent IR-HARQ schemes with $M = 1, 2, 3, 4, 5, 6, 7, 170$ obtained by solving the maximization problem in (3.10), with the constraints in (3.11). It can be seen that the gain for $M = 1$ and $M = 2$ is considerable as compared to Figure 7.13 and also compared to the parallel case in Figure 7.6. As M increases, the gain diminishes and for $M = 170$, it is the same average code rate as for case 1.

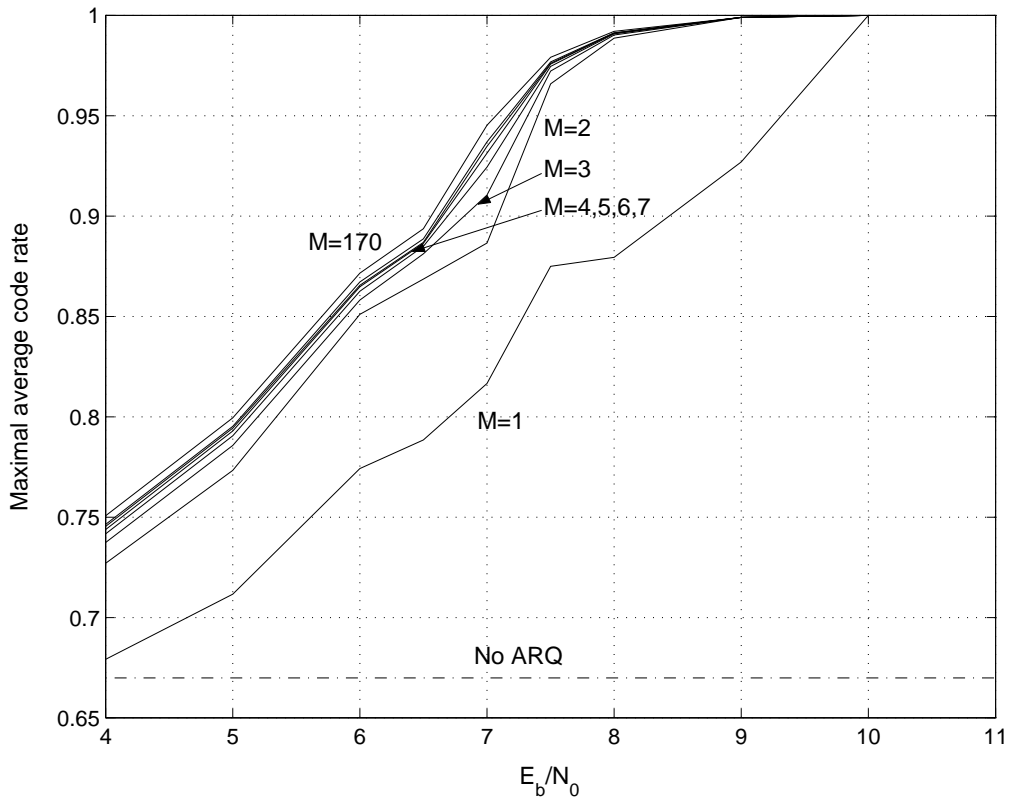


Figure 7.14. Maximum average code rate as a function of E_b/N_0 for different M , based on the 3D SCSPC(8,7) mother code. Here c_M only contains enough parity bits to obtain a FER of 10^{-2} , a.k.a. case 2.

Finally in Figure 7.15 the maximal average code rate for case 2, with a target FER of 10^{-3} is plotted. There we have also plotted the corresponding rates of the dimension borders.

An example scenario is selected and its performance is simulated when optimal packet lengths are used in the IR-HARQ scheme. The example scenario considers case 2 with $M = 5$ and a target FER of 10^{-2} . The packet lengths are reported in Table 7.3. The FER performance of the simulation should follow the mother code until the FER reaches 10^{-2} . Thereafter the performance should settle at a FER of 10^{-2} until E_b/N_0 is sufficiently high so that the uncoded case provides the required target FER. Thereafter the performance should instead follow the uncoded case. We can see from Table 7.3 that the full mother code of 512 is never used since the selected target FER is quite high.

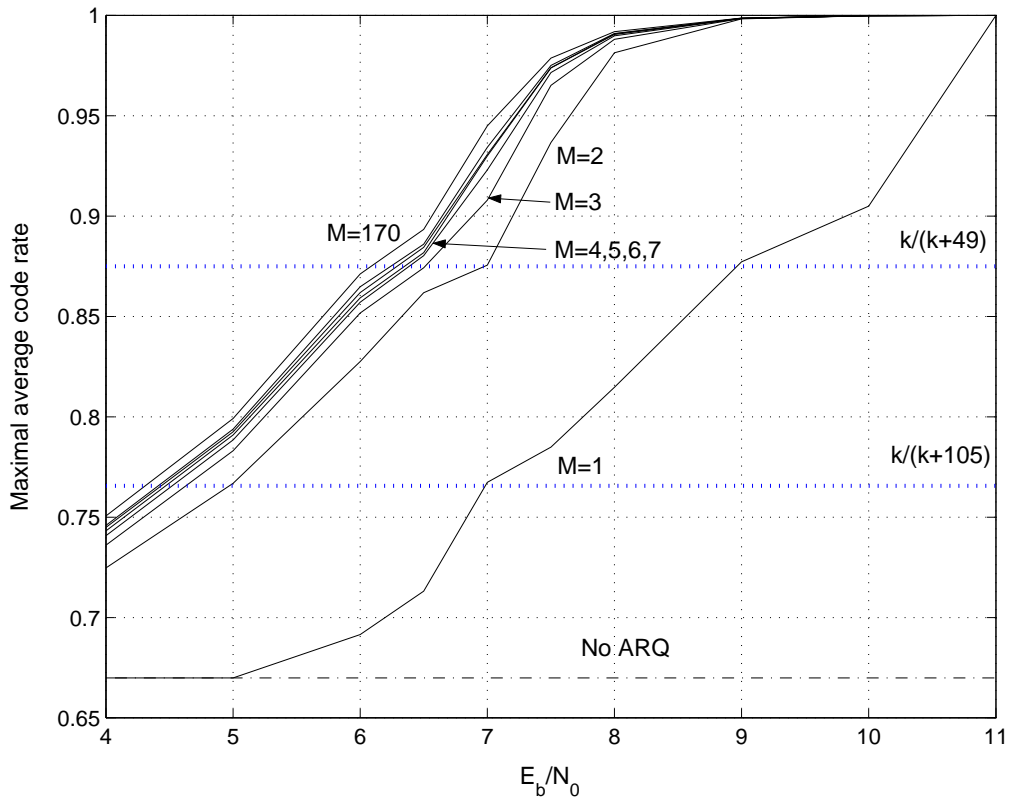


Figure 7.15. Maximum average code rate as a function of E_b/N_0 for different M , based on the 3D SCSPC(8,7) mother code. Here c_M only contains enough parity bits to obtain a FER of 10^{-3} , corresponding to case 2

Table 7.3. Optimal packet lengths for QoS case 2 with $FER=10^{-2}$ and $M=5$ using the 3D SCSPC(8,7) mother code and dimension-wise puncturing, starting from the innermost code.

E_b/N_0	r_C	n_1	n_2	n_3	n_4	n_5
4.0 dB	0.7439	450	457	466	480	505
5.0 dB	0.7930	423	429	437	448	482
6.0 dB	0.8648	387	392	408	424	443
7.0 dB	0.9314	343	362	377	392	420
8.0 dB	0.9911	343	362	372	380	390
9.0 dB	0.9991	343	359	362	366	370
10.0 dB	1.0000	343	343	343	343	343
11.0 dB	1.0000	343	343	343	343	343

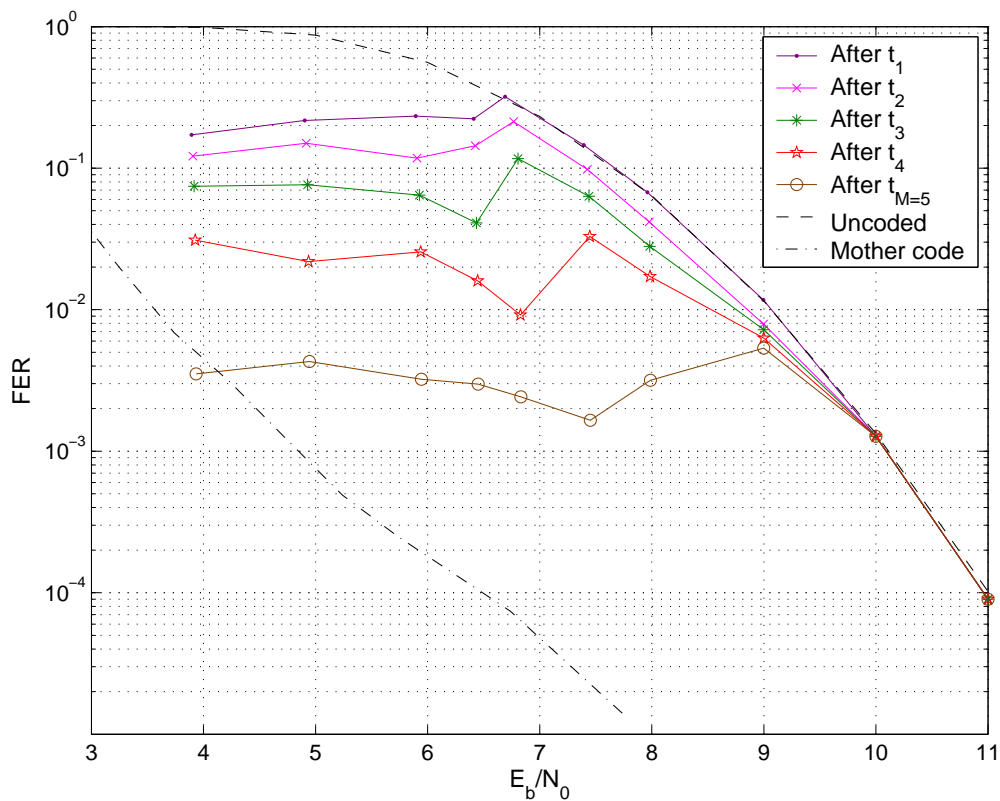


Figure 7.16. Simulated performance of a QoS case 2 with $M=5$ and $FER 10^{-2}$ for the mother code 3D SCSPC(8,7).

We can see from Figure 7.16 that the IR-HARQ scheme performs better than the mother code for 4 dB, both since all transmissions are not always required and since the entire mother code is not required to obtain the requested target FER. For 5-9 dB the performance remains quite stable at a level somewhat lower than 10^{-2} . Since $M = 5$, we have a more flexible scheme allowing frames to be accepted with less excess redundancy.

Finally in Figure 7.17, the simulated code rate is compared to the calculated. Since upper bounds on the FER are used in the optimization process, they may result in a lower bound on the code rate. This may explain why the simulated performance is superior to the calculated.

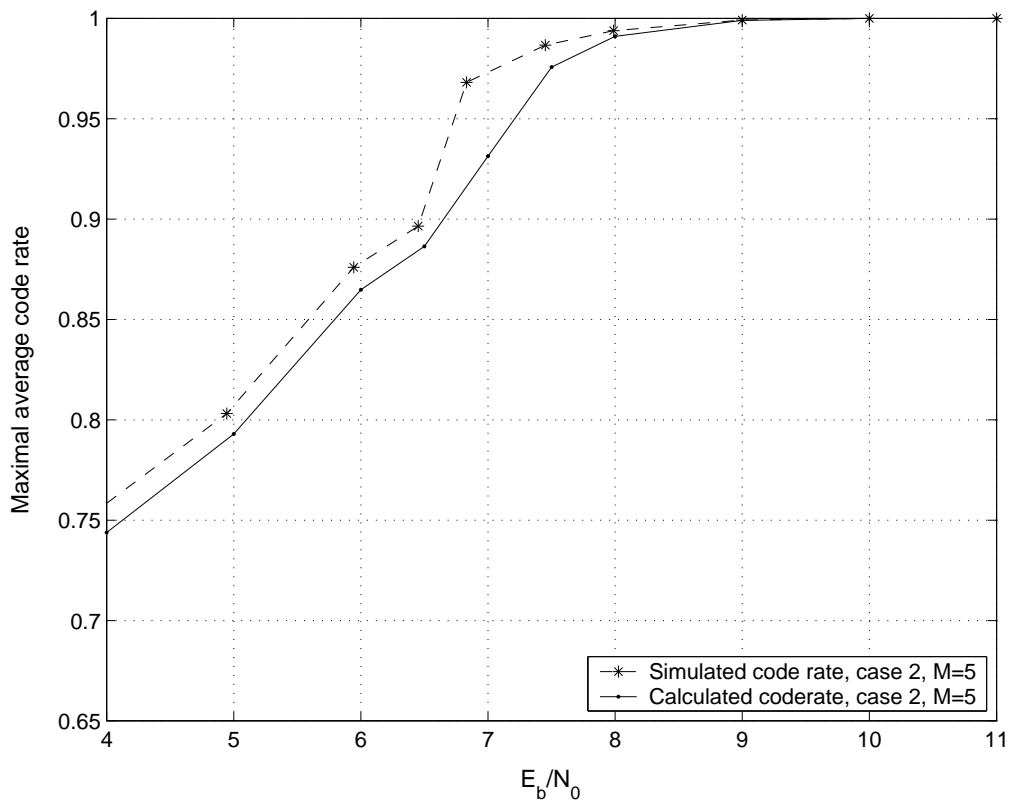


Figure 7.17. Maximum average code rate as a function of SNR for $M=5$, case 2. The simulated value is compared to calculated.

Chapter 8

Conclusions

In this chapter, the objectives of the thesis are reconsidered and contrasted against the contributions made in terms of enabling tools and techniques. It is concluded that the primary objective of developing a QoS-based adaptive coding scheme as a core component in a wireless real-time communications protocol has been successful. The contributions of the thesis work are summarized, insights are discussed and concluding remarks made. The impact of the obtained results and the relations to results in the literature are considered. The influence of the obtained results in wider perspectives is also argued and directions for future work are proposed. Future directions are partly inspired by obvious extensions of the reported work, but also by future applications and potential trends in the design of wireless communications systems.

8.1 Objectives Achieved

As stated in Chapter 1, the primary objective of this thesis is to propose good channel coding and decoding methods to be used in a wireless real-time communication system. To that end, the study has been focused on developing a foundation for efficient and reliable real-time communications protocols for critical deadline dependent communication over wireless channels. The proposed solution is based on the concept of DDC, which has proven to be a promising design approach for achieving this objective. The main idea behind the concept of DDC is to make the communication protocol deadline dependent, tailoring the channel code to the real-time constraints.

The specific aim is to provide a high level of flexibility by adopting adaptive principles in the underlying communications protocols in terms adaptive coding strategies. A novel QoS-based adaptive coding scheme is proposed that integrates physical layer error control coding, data link layer retransmission schemes, and parts of network layer QoS responsibilities. Such an approach, based on a concatenated system design, moves beyond

current network-layer protocol paradigms, resulting in a novel approach for guaranteeing QoS by exploiting the strong interplay between communication theory and networking techniques.

In a typical wireless communication system, signals transmitted over the channel are perturbed by many time-varying factors. As a consequence, each received data frame can experience different channel characteristics. In such a dynamic environment, retransmission schemes in terms ARQ strategies are highly tractable coding strategies due to their adaptive and flexible nature. A retransmission protocol implies maximizing the probability of correct delivery in a dynamically changing environment, using a minimum of resources.

Concatenated coding within ARQ protocols provides a new array of possibilities for adaptability in terms of decoding complexity and communication time versus reliability. Hence, critical reliability and timing constraints can be readily evaluated as a function of available system resources and complexity. This in turn enables quantifiable QoS and thus negotiable QoS. Service requests can therefore be accepted, rejected or re-negotiated depending on available resources.

Consequently, a central contribution of this work is the IR-CHARQ-DDC protocol and the attempt to bring together the areas of coding theory and real-time communication. The primary objective is achieved by developing an analytical design framework for the optimization of IR-CHARQ protocols, based on:

- Analytical methods for incorporating strict QoS constraints into the IR-CHARQ protocol design, in terms of mapping parameters such as deadline and reliability directly within the protocol design.
- Analytical tools for performance evaluation of IR-CHARQ protocols, in terms of bounding and convergence analysis techniques for rate-compatible code families.

These tools have made it possible to reach two target outcomes, namely

- A formal methodology for optimal IR-CHARQ protocol design, subject to strict QoS requirements;
- New protocol designs for QoS-aware wireless communication networks that makes better use of limited resources such as power and supportable computational complexity,

which, in turn, have achieved the primary objective of the thesis work. The component contributions leading to these target outcomes are discussed in more detail in the following section.

8.2 Contributions and Impact

The primary contribution of this thesis is the analytical design procedure for QoS-based IR-CHARQ protocols. The QoS parameters are here in terms of a deadline and a target FER. Based on a target FER, a suitable rate compatible code family is selected where the mother code is able to provide the required reliability. As the delay in connection with a retransmission is the most significant contribution to the overall delay in an ARQ scheme,

the deadline determines the maximum number of retransmissions allowed. Given the maximum number of transmissions allowed and the selected rate compatible code family, optimal packet lengths for each transmission and corresponding good puncturing patterns are then found to complete the design. As detailed in Chapter 3, the proposed design procedure is as follows:

1. Select a mother code to provide the required target FER.
2. Select a rate compatible coding family with corresponding puncturing patterns.
3. Select a maximum number of transmissions to provide the required target deadline.
4. Obtain the FER as a function of the codeword lengths.
5. Find optimal packet lengths that maximize the average code rate.

This design approach becomes analytical as it is based on a series of analytical tools, which are discussed in more details below. In this thesis, the approach is exemplified using rate-compatible code families based on multiple concatenated single parity check codes. However, the general approach is not limited to these code families, but with modifications can be applied to any arbitrary code family of choice.

The feature of mapping QoS requirements directly to parameters and building blocks in the underlying IR-HARQ scheme leads to a very flexible structure, able to provide several different solutions in terms of combinations of resources, achieving the same perceived QoS. This, in turn, allows for resource allocation with additional degrees of freedom in finding the most favorable ways of using the available resources, and satisfying simultaneous QoS requirements for more users in the network.

The novelty of the approach is to develop an analytical framework for optimal design of IR-HARQ schemes both in terms of IR strategy and puncturing pattern such that the average code rate is maximized. This is in contrast to previous work, where only the puncturing pattern has been optimized for maximum throughput, given a specific IR strategy. This approach provides a new theoretical perspective on IR-HARQ design.

8.2.1 Optimization of Packet Lengths

Previous work on IR-HARQ schemes has been focused on finding optimal puncturing patterns for maximizing the throughput, given an IR strategy, i.e., a maximum number of allowed transmissions and the number of parity bits to be included in each IR transmission. In contrast, we propose an analytical approach for finding the optimal IR strategy, given a puncturing pattern. Good rate-compatible coding families using suitable puncturing strategies are selected specifically for the optimized IR-HARQ schemes. The guiding principle in the design procedure is to maximize the average code rate subject to QoS constraints and subject to minimizing the use of limited resources.

The analytical approach allows for incorporating QoS constraints in terms of deadline and reliability requirements over a range of SNRs into the optimization procedure. A strict deadline enforces a maximum number of allowed transmissions, while reliability

requirements will determine the specific low rate mother code to use, as well as the number of parity bits to be included in the last allowed transmission as a function of the effective SNR. The mapping of the QoS parameters to the IR-CHARQ-DDC scheme may be done using a look-up table or, alternatively if the transmitter and receiver can be more costly, by using adaptive strategies.

Having a truncated IR-CHARQ protocol gives an upper bound on the required transmission time for each information frame. This in turn yields an average transmission time and hence slack time. This slack time may be used to run non-real-time tasks, or alternatively seen as a way to limit the interference to other nodes. Further using the IR-HARQ-DDC scheme, communication tasks may be preempted.

8.2.2 Multiple Concatenated SPC Codes

Rate-compatible coding families based on multiple concatenated codes are especially advantageous in conjunction with retransmission schemes, creating highly flexible and adaptive CHARQ systems. Systematic concatenated codes are particularly flexible as additional component codes concatenated in serial or parallel can be included directly in case lower code rates are required.

Multiple concatenated SPC codes have recently been shown to provide a good performance/complexity trade-off, making them suitable for low-complexity real-time communications systems. The performance of these codes is remarkably good given the low decoding complexity. In this thesis, both parallel and serially concatenated SPC codes have been investigated for use in IR-HARQ protocols.

With rate compatible codes, the same decoder can be used to decode all code rates. For applications requiring low decoding complexity, dimension-wise puncturing may be applied. Using dimension-wise puncturing only component decoders for which we have received parity bits need to be activated in the iterative decoding process. Not only does dimension-wise puncturing constitute a low-complexity puncturing pattern, but it is also found to have superior performance when compared to random puncturing using a Monte-Carlo simulation.

8.2.3 Performance Bounds

Analytical upper bounds are presented for both punctured and full-rate multiple parallel and serial SPC codes. Bounds for multiple parallel SPC codes with punctured information bits are also given. The bounds are found to be tight for a relatively low SNR for both BER and FER. The FER for all possible rate-compatible codes using the chosen puncturing pattern are obtained for use in the optimization of IR packet lengths, where the FER as a function of the block length is required.

In addition, the union bounds are used to select good puncturing patterns, which are chosen to give low FER for low SNRs. In order to guarantee the invertibility of the

resulting punctured codewords, only puncturing of parity bits are included in the search. For multiple parallel concatenated SPC codes, dimension-wise puncturing is found to be the best pattern.

8.2.4 EXIT Charts Analysis

The average behavior of iterative decoding strategies can be analyzed using EXIT charts, where the iterative behavior is captured by mutual information exchanges between component decoders. The convergence behavior is investigated for multiple concatenated SPC codes using EXIT charts.

Further, EXIT chart analysis is employed to find optimal puncturing ratios for each component code branch in the multiple concatenated code, given a specific overall code rate. The puncturing ratio is the ratio of punctured bits to the total number of bits in the mother codeword. The optimal puncturing ratios minimize the required SNR for which the iterative decoder will converge, and thus finds the minimum SNR for which the resulting code rate provides acceptable error rate performance. This procedure is conducted for the code rates provided by puncturing the mother code, giving a SNR interval within which a particular coding scheme is useful. Given these puncturing ratios, the corresponding BER can be approximated based on mapping from mutual information to BER. Due to the simple structure of SPC codes, the puncturing ratios translate directly into a puncturing pattern.

In order to guarantee the invertibility of the resulting punctured codewords a search is made for optimal puncturing ratios under the constraint that the systematic bits are left unpunctured. For multiple parallel concatenated SPC codes, dimensions-wise puncturing is found to be the best, confirming the results made using bounds. For multiple serial concatenated SPC codes, backwards dimension-wise puncturing, i.e., puncturing starts in the outermost code, is found to be the best.

In addition, a search for rate compatible puncturing ratios when allowing systematic bits to be punctured is made. If the search starts at the lowest possible code rate, more and more systematic bits are punctured as the code rate increases. Hence, due to the rate compatibility constraint, this search does not result in good puncturing ratios for high code rates, since no systematic bits should be punctured for rates close to one. Since we are using IR-HARQ schemes, we are interested in good performance at high code rates. Consequently, a search starting from the highest possible code rate is made. This results in a pattern where systematic bits are left unpunctured since that yields best performance at a code rate close to one. Consequently, dimension-wise puncturing for parallel SPC codes and backwards dimension-wise puncturing for serial SPC codes are the rate compatible puncturing patterns that yields the lowest convergence thresholds. Backwards dimension-wise does not, however, result in the lowest possible decoding complexity since all component decoders have to be activated even if we have only received parity bits pertaining to the outermost code.

8.3 Perspectives

Future wireless communications networks will require vast improvements in data rates and user-mobility in order to meet the increasing demands of advanced data services. The wide variety of services is increasing and has resulted in mixed communication traffic with an equally wide range of data-quality requirements. The service quality needed for each particular application can be quantified in terms of data rate, latency and reliability; quantities which are referred to as QoS parameters. For example, data services such as emailing and web browsing are sensitive to transmission errors, but relatively tolerant to transmission delay. In contrast, telephony is delay sensitive but relatively error tolerant, while many new wireless services such as video and CD-quality audio are both delay and error sensitive. The challenge is to design new wireless networks that can accommodate high data rate applications with a wide range of QoS requirements in an environment with large densities of highly mobile users.

A challenging problem for such a wireless network design is to enable the best possible use of limited network resources and to accommodate QoS sensitive high data rate applications. Optimal resource allocation and QoS control at network level are facilitated by adaptive physical and data link layers. A promising solution is a cross-layer design based on iterative signal processing in a concatenated communication system. The adaptive concatenated coding scheme proposed in this thesis has the potential of being a core component in such a design, bridging physical, data link and network layers and leading to new QoS-based cross-layer communication protocols.

The novel approach for wireless network design is to propagate methodologies of physical layer design across the disciplinary boundaries within wireless network design in a bottom-up cross-layer approach, enabling more efficient use of limited resources. This approach will also facilitate implementation of QoS enabling strategies, making guaranteed QoS possible and controllable. The underlying innovation is contained in the interdisciplinary approach, leading to cross layer protocol design. This is a first step towards expanding the level of interdisciplinary research across boundaries within the traditional communication protocol stack.

8.4 Future Work

In this thesis, communication over an AWGN channel has been considered for the analytical approach. The AWGN channel provides a tractable analytical design tool and has therefore been the focus here. However, in a wireless environment we encounter more hostile channel impairments such as multipath propagation and fading. The analytical approaches developed in this thesis are also applicable to the fading case, and thus an obvious extension of the work is to apply the framework to more realistic wireless channel models.

In addition, the protocols are intended to be used in a wireless network, where multiple

access is used to share resources among a set of active users. Due to the limitations of the multiple access scheme and/or the stochastic nature of the wireless channel, active users will invariably cause interference. Another interesting extension of the current work is to consider a multi-user environment, investigating joint design of multi-user detection and scheduling strategies.

The adaptive coding scheme could be improved further by recursive modulation incorporated in the concatenated code or by using more advanced component codes, such as e.g., the zigzag code suggested in Appendix A. Both these approaches have been shown to result in a significant interleaver gain.

The use of performance based stopping criterion for the iterative decoding process, simultaneously used as a retransmission criterion, enables instantaneous adaptation and a performance based retransmission criterion accessible in practice.

From a real-time perspective, application specific encoding of data may be applied using the unequal error protection originating from puncturing to protect some bits of elevated importance.

A more far-reaching direction of future work is to consider the perspective of wireless network design based on a concatenated system design briefly discussed above. An adaptive communication protocol involves many functionalities and technologies in addition to channel coding. Inspired by the success of concatenated approaches, the interaction between functionalities at the transmitter can be modeled as a hybrid concatenated system with multiple parallel and serial branches. The receiver then has a similar concatenated structure where components recursively interchange information. The adaptive coding component proposed in this work will integrate physical layer error control coding, data link layer retransmission schemes, and some parts of network layer QoS responsibilities, constituting the core component in the concatenated system design. However, it is an interesting challenge to formulate a tractable optimization problem for a larger-scale cross-layer design objective, leading to a high level of protocol flexibility. A potential outcome of such a research focus can be novel cross-layer protocol designs for QoS-aware wireless communications networks based on a joint concatenated systems design.

Appendix A

Multiple Parallel Concatenated Zigzag Codes

Zigzag codes, originally suggested in [106], can be viewed as a modified SPC code, a two-state convolutional code, or an LDPC code [31, 107]. They are more complex to decode than a corresponding SPC code, but less complex than turbo codes. For that reason they are suggested here as an alternative to SPC component codes. It will be shown, using EXIT functions, that the ZZ codes have a recursive structure and are therefore more robust against puncturing.

Efficient low-complexity decoding algorithms for ZZ codes have been suggested and evaluated in [31, 107]. Finally, a type-II HARQ scheme using multidimensional parallel concatenated ZZ codes is suggested in [108].

A.1 Encoding

The encoding of a ZZ code has been described in [31]. The information frame, \mathbf{x}_0 , is arranged in a matrix with L rows and J columns denoted \mathbf{x}' , according to Figure A.1. Note that $J = k_C$. Denote the j -th element in the l -th row of \mathbf{x}' as $\mathbf{x}'(l, j)$, where $l = 1, 2, \dots, L$ and $j = 1, 2, \dots, J$. The first row of the ZZ code is identical to an SPC code, and thus its parity bit is generated according to

$$\mathbf{y}'_i(1) = \left(\sum_{j=1}^J \mathbf{x}'(1, j) \right) \bmod 2, \quad (\text{A.1})$$

where $i = 1, 2, \dots, U$. Parity bits for the following rows are generated according to

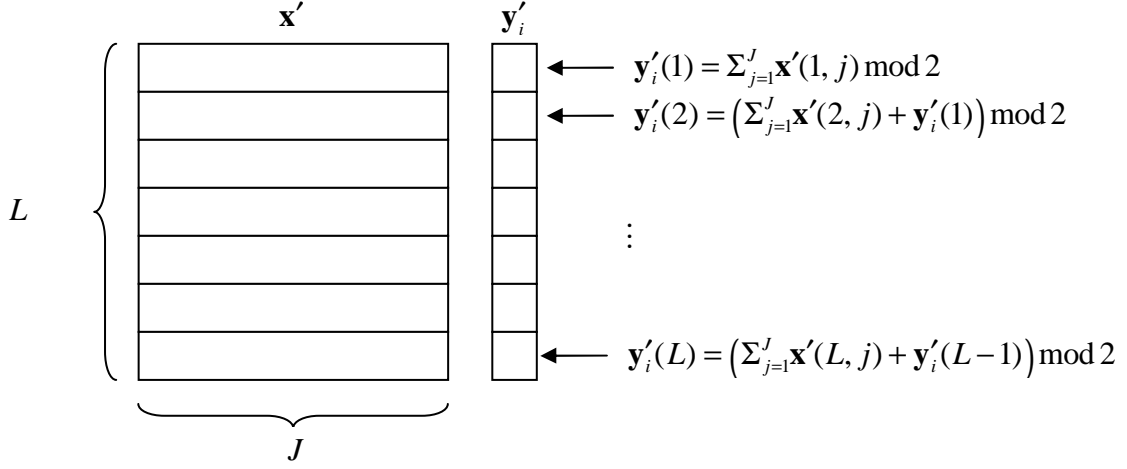


Figure A.1. Encoding of a zigzag code.

$$\mathbf{y}'_i(l) = \left(\mathbf{y}'_i(l-1) + \sum_{j=1}^J \mathbf{x}'(l, j) \right) \bmod 2, \quad \text{for } l = 2, 3, \dots, L. \quad (\text{A.2})$$

The code rate and the minimum distance of a parallel concatenated ZZ (PCZZ) code is the same as for a PCSPC code.

A.2 Decoding

The decoding algorithm derived in [31] is used to decode the ZZ codes. It is reviewed here for completeness. The received vector, \mathbf{r} , is arranged in a matrix with L rows and $J + 1$ columns denoted \mathbf{r}' . Denote the j -th element in the l -th row of \mathbf{r}' as $\mathbf{r}'(l, j)$, where $l = 1, 2, \dots, L$ and $j = 1, 2, \dots, J + 1$. First, the forward recursion, \mathbf{F} , on the parity bits is calculated according to

$$\mathbf{F}(0) = +\infty$$

$$\mathbf{F}(l) = L_c \mathbf{r}'(l, J + 1) + 2 \operatorname{arctanh} \left(\tanh \left(\frac{\mathbf{F}(l-1)}{2} \right) \prod_{j=1}^J \tanh \left(\frac{L_c \mathbf{r}'(l, j) + L(\mathbf{x}'(l, j))}{2} \right) \right), \quad (\text{A.3})$$

for $l = 1, 2, \dots, L$ where $L_c = 2\mu / \sigma_w^2$. Thereafter the backward recursion, \mathbf{B} , is evaluated as

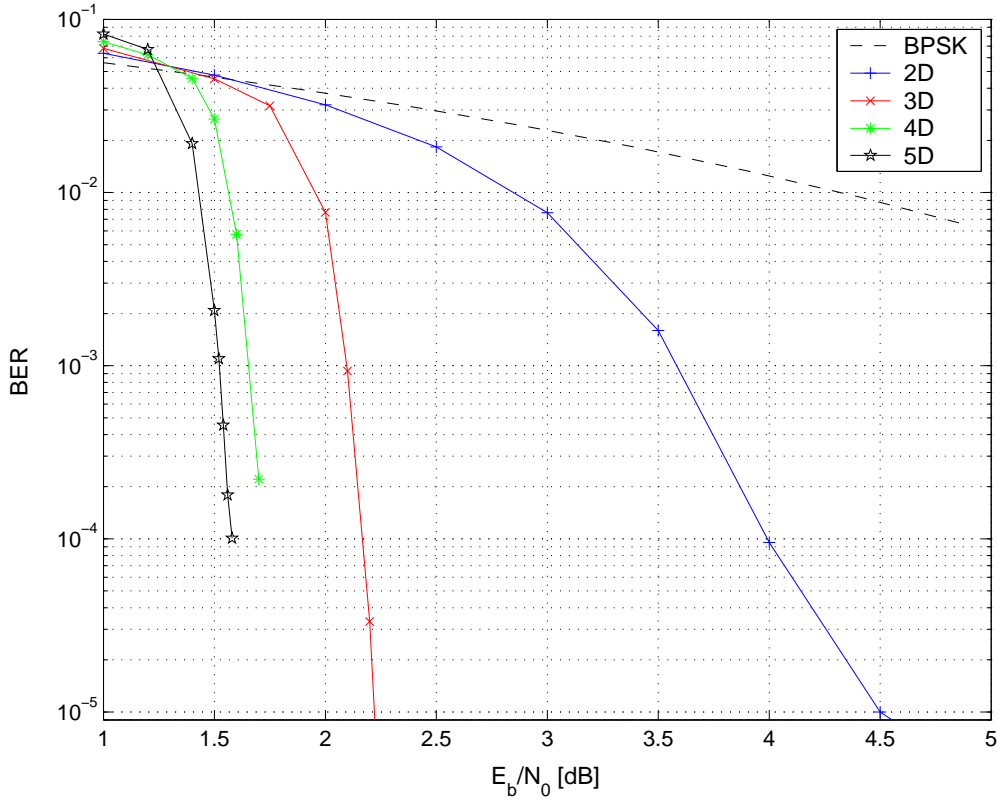


Figure A.2. PCZZ(8,7) codes with 2-5D using large interleavers.

$$\mathbf{B}(L) = L_c \mathbf{r}'(L, J+1)$$

$$\mathbf{B}(l-1) = L_c \mathbf{r}'(l-1, J+1) + 2 \operatorname{arctanh} \left(\tanh \left(\frac{\mathbf{B}(l)}{2} \right) \prod_{j=1}^J \tanh \left(\frac{L_c \mathbf{r}'(l, j) + L(\mathbf{x}'(l, j))}{2} \right) \right), \quad (\text{A.4})$$

for $l = L, L-1, \dots, 2$. Finally, the APP is given as

$$L(\mathbf{x}'(l, q) | \mathbf{r}') = L_c \mathbf{r}'(l, q) + L(\mathbf{x}'(l, q)) + 2 \operatorname{arctanh} \left(\tanh \left(\frac{\mathbf{F}(l-1)}{2} \right) \tanh \left(\frac{\mathbf{B}(l)}{2} \right) \prod_{\substack{j=1 \\ j \neq q}}^J \tanh \left(\frac{L_c \mathbf{r}'(l, j) L(\mathbf{x}'(l, j))}{2} \right) \right), \quad (\text{A.5})$$

where $q = 1, 2, \dots, J$ and $l = 1, 2, \dots, L$. Note that the forward and backward recursions are calculated once every time the respective component decoder is activated, rather than once per row.

In Figure A.2 the performance of 2-5D PCZZ codes using large interleavers is shown.

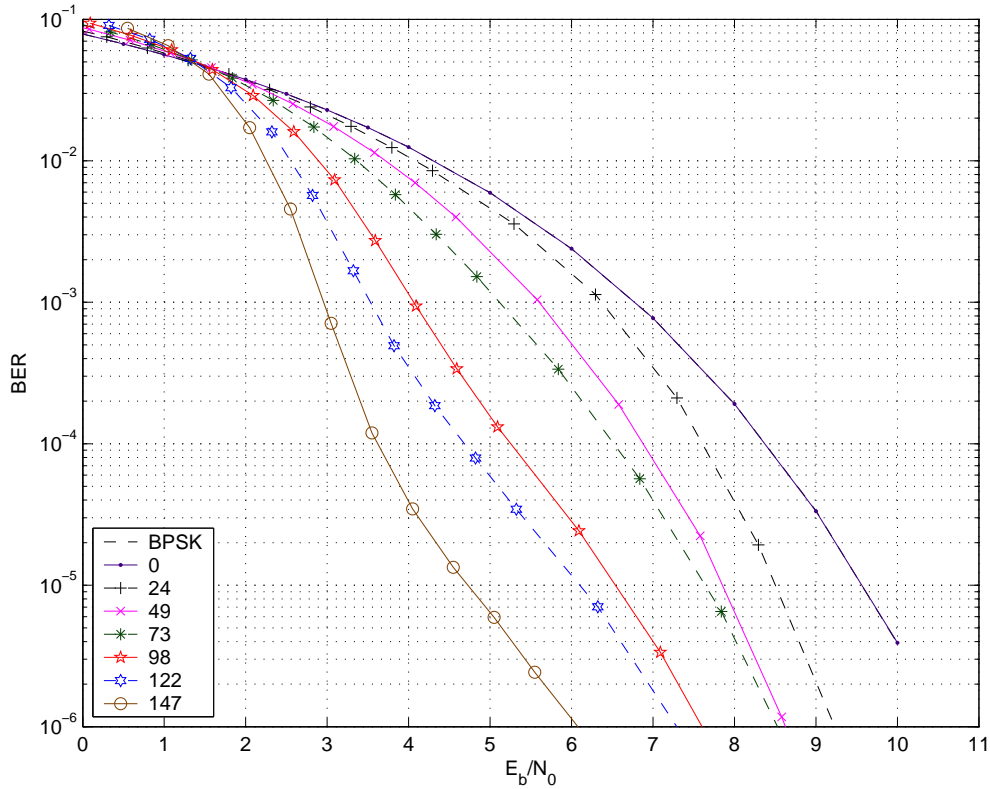


Figure A.3. Performance of the 3D PCZZ(8,7) code as a function of the number of parity bits transmitted.

These curves should be compared to the ones in Figure 4.4 and Figure 4.5 for PCSPC and SCSPC codes respectively. We can see that the performance for a PCZZ code is in the same range as the performance of a SCSPC code and in most cases better, even though its code parameters, such as code rate and the interleaver sizes is the same as for a PCSPC code

A.3 Puncturing

The performance of a 3D PCZZ(8,7) code with an information frame of size $k = 7^3 = 343$ is shown in Figure A.3. The two interleavers are of size 343 and each encoder adds 7^2 parity bits summing up to a total codeword of length $n = 7^3 + 3 \cdot 7^2 = 490$, just as in the case with the 3D PCSPC(8,7) code. Dimension-wise puncturing starting with the first row is applied, and the maximum number of parity bits is 147. The curves in Figure A.3 should be compared to Figure 4.6 for PCSPC and Figure 4.7 for SCSPC codes. We can conclude that the PCZZ code has code parameters matching the PCSPC code, but a performance that more closely matches that of the SCSPC code. Hence, even with the same amount of redundancy as that of a PCSPC code the performance of the PCZZ code is comparable to a SCSPC code. Moreover, it is interesting to notice the performance when only 24 parity bits

have been transmitted. Quite contrary to the concatenated SPC codes, a few parity bits from the first dimension make a huge difference here, since the ZZ code involves all rows of the information frame in the encoding process and hence there are no completely unprotected information bits even if puncturing is applied. The encoding and decoding complexities however are slightly higher for ZZ codes than for SPC codes and thus there is a tradeoff between energy used for transmission of additional parity bits and energy used to cater for a more complex decoder. Alternatively, since the gain seems to be greatest in the first dimension, ZZ codes may be used in only one dimension and SPC codes in the rest. This yields an additional way of providing trade off between complexity and performance. Note that in this case, dimension-wise puncturing should start with the dimensions encoded with SPC codes, so that the ZZ code is indeed left until last.

A.4 Extrinsic Information Transfer Functions

The SPC(8,7) can be seen as a special case of a ZZ(8,7) code, namely when the ZZ code has only one row. Therefore the EXIT functions for a ZZ code with only one row are also given in Figure 6.4 and Figure 6.5. The code structure of a ZZ code depends on the number of rows used in the code. In Figure A.4 the EXIT function $I_{E(x)}$ for different number of rows is shown. Note that the number of rows used in the component code is not directly related to the interleaver size, since the minimum block can be repeated a number of times until an appropriate interleaver size is achieved – similar to what is done for SPC codes. We can see from $I_{E(x)}$ in Figure 6.4 and from Figure A.4 that the more rows that are used in the ZZ component code the closer we get to $I_{E(x)} = f_x(1, I_{A(y)}) = 1$, and hence to a recursive code. Thus, we can conclude that one should always incorporate as many rows as the interleaver allows in a ZZ component code, rather than repeating a minimum block several times. 49 rows of a ZZ(8,7) code corresponds to a minimal interleaver size of $49 \cdot 7 = 343$ in a parallel concatenated system. Hence using an interleaver of size $7^3 = 343$ and letting the ZZ code span all the rows of the interleaver yields a code that is virtually recursive.

A.5 Extrinsic Information Transfer Charts

The EXIT charts for multiple parallel concatenated ZZ codes are constructed the same way as for multiple PCSPC codes, with the exception that the EXIT function from Figure A.4 is used instead. We are interested in looking at how $I_{E(x_1)}$ and $I_{E(x_2)}$ evolve when the component decoders are activated, thus

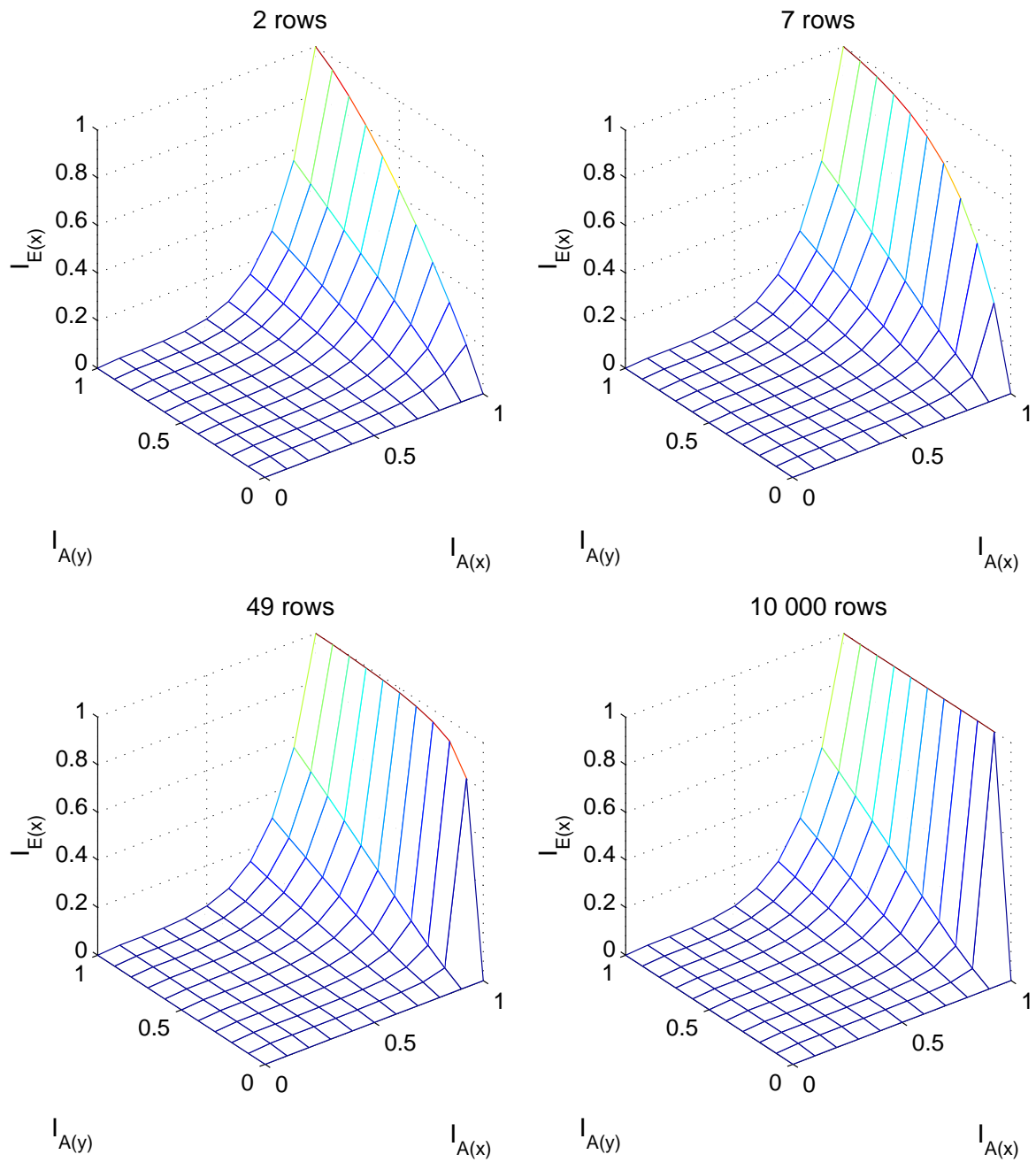


Figure A.4. The $I_{E(x)}$ EXIT function for a ZZ(8,7) code as a function of the number of rows used in the codeword.

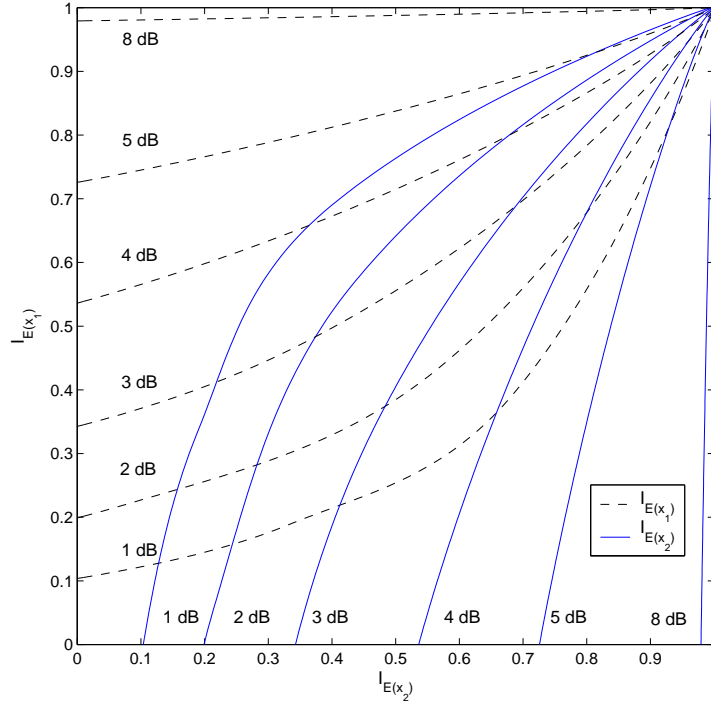


Figure A.5. EXIT chart for the 2D parallel concatenated ZZ(8,7) code.

$$I_{E(x_i)} = f_{x_i}(I_{A(x_i)}, I_{A(y_i)}) = f_{x_i} \left(J \left(\sqrt{\frac{8r_{C_{UD}} E_b}{N_0} + \sum_{\substack{i=1 \\ i \neq l}}^U J^{-1}(I_{E(x_i)})^2} \right), J \left(\sqrt{\frac{8r_{C_{UD}} E_b}{N_0}} \right) \right), \quad (\text{A.6})$$

where $f_{x_i}(\cdot)$ is the EXIT function with 49 rows from Figure A.4. The EXIT function with 49 rows is chosen to mimic the example code plotted in Figure A.3. In Figure A.5 to Figure A.8 we have plotted the EXIT charts for 2D, 3D, 4D and 5D PCZZ(8,7) codes respectively.

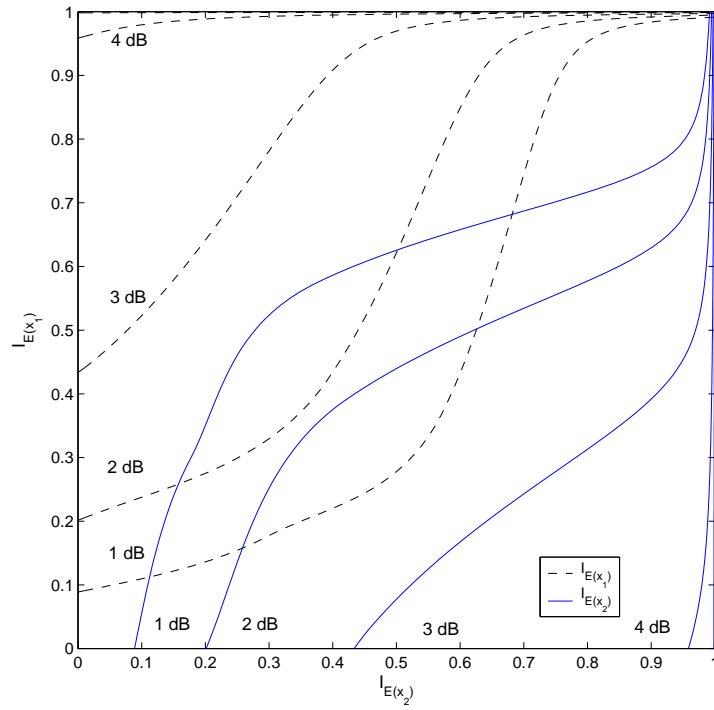


Figure A.6. EXIT chart for the 3D parallel concatenated ZZ(8,7) code.

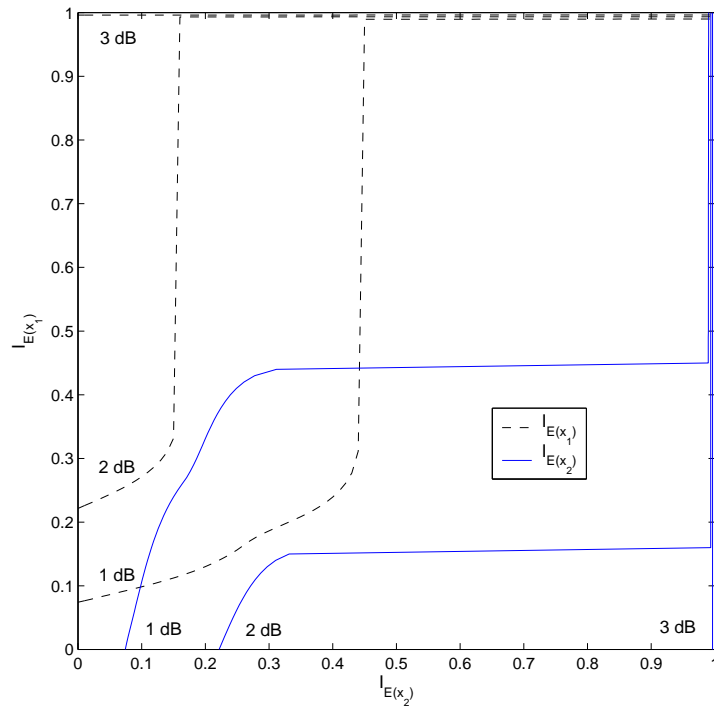


Figure A.7. EXIT chart for the 4D parallel concatenated ZZ(8,7) code.

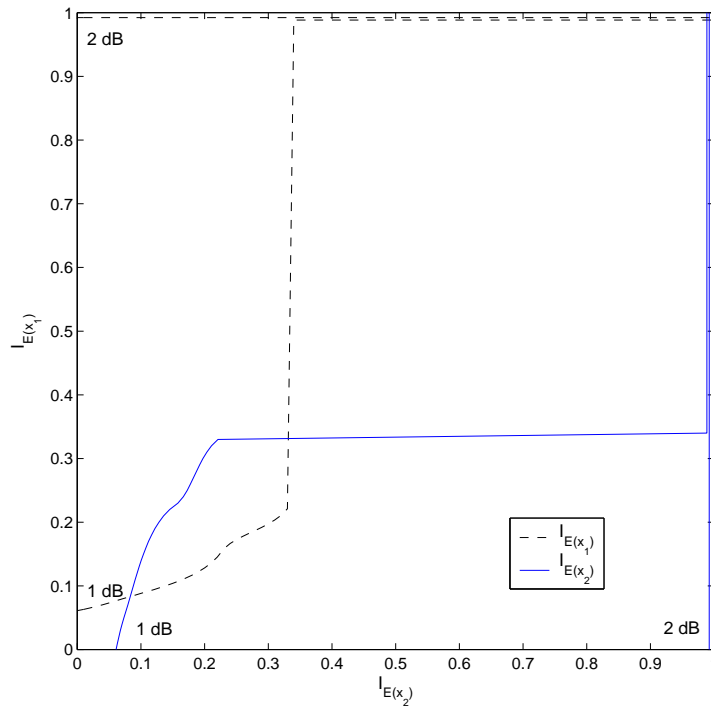


Figure A.8. EXIT chart for the 5D parallel concatenated ZZ(8,7) code.

The good performance of the ZZ codes can clearly be visualized in the EXIT charts. Then the EXIT charts are used to estimate the BER of PCZZ codes, the similar problem as with SCSPC codes occur, namely that the BER estimation is less accurate in the waterfall region. This is reported in Figure A.9.

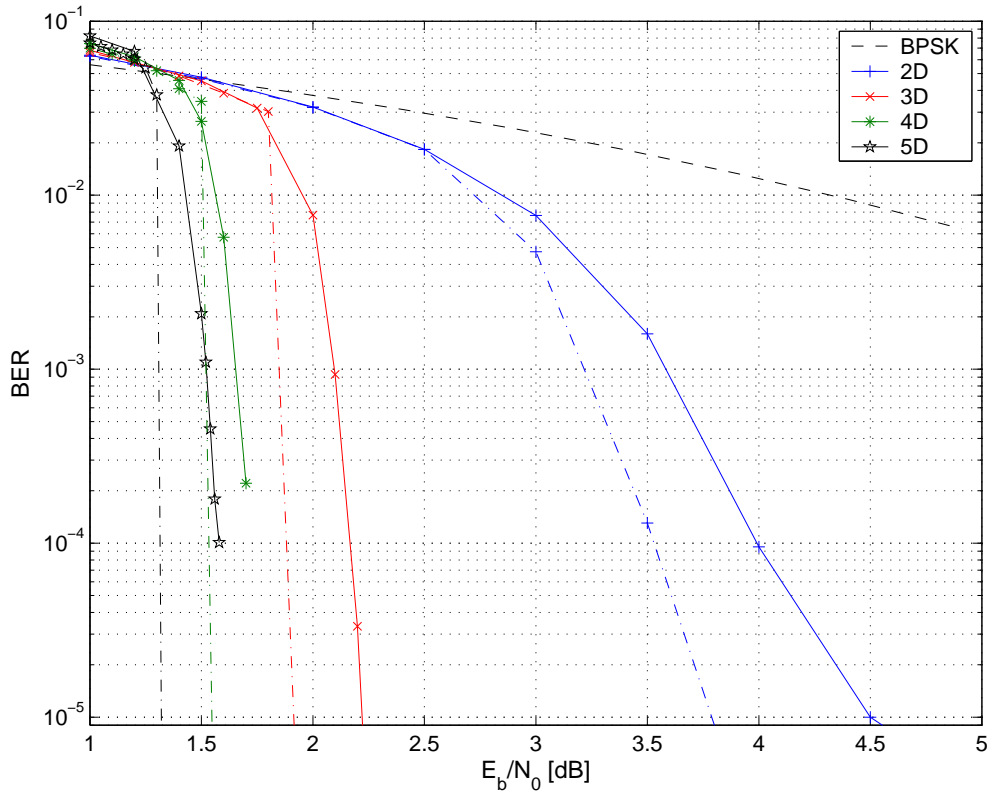


Figure A.9. BER obtained from simulations (solid lines) and EXIT chart (dash-dotted lines) for 2D, 3D, 4D and 5D PCZZ(8,7) codes.

A.6 Puncturing Ratios Obtained Using EXIT Charts

When searching for optimal puncturing ratios for PCZZ codes, the same reasoning and equations as used for PCSPC codes in Chapter 6 can be applied. The only difference is that the EXIT function in Figure A.4 for 49 rows is used. For the 2D and the 3D PCZZ(8,7) codes, the γ_b needed for convergence to P_b^* , when the optimal values of Δ are used are reported in Figure A.10. Curves corresponding to four different values of P_b^* are plotted together with the C^* that can be used as a reference. Recall that it is possible to achieve rates above capacity if the target BER is sufficiently high, since C^* assumes an arbitrary low BER. We can see that the performance of PCZZ codes is much better than PCSPC codes and also better than SCSPC codes. The 2D curves benefit from the recursive structure supplied in each dimension and are thus much better than the 2D SCSPC code. For the 3D case the performance of the PCZZ code is still better than the SCSPC code, but the difference is reduced. The performance improvement is most noticeable for high code rates.

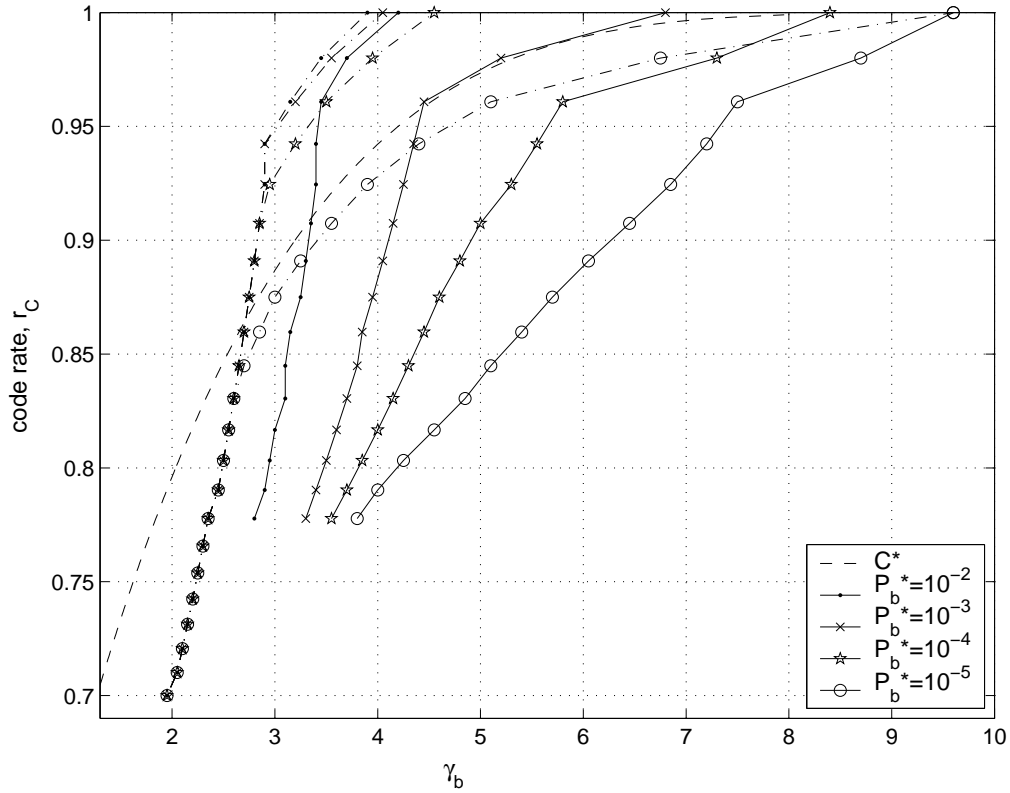


Figure A.10. The γ_b -values required for convergence to P_b^* as a function of the code rate r_c for a 2D (solid lines) and a 3D (dotted lines) PCZZ(8,7) code that use optimal values of Δ . C^* is plotted as a reference.

The optimal values of Δ for different code rates of the 3D PCZZ code are plotted in Figure A.11 for $P_b^* = 10^{-5}$. As can be seen, puncturing of systematic bits is still made, but it not to the same extent as for SPC codes.

Using ZZ codes, the puncturing ratios do not translate as easily into puncturing patterns as for SPC codes since the different parity bits in a dimension may have different importance. This may also explain why the puncturing ratios in Figure A.11 are less regular than for SPC codes. Hence, the results apply assuming that random puncturing within each dimension is used.

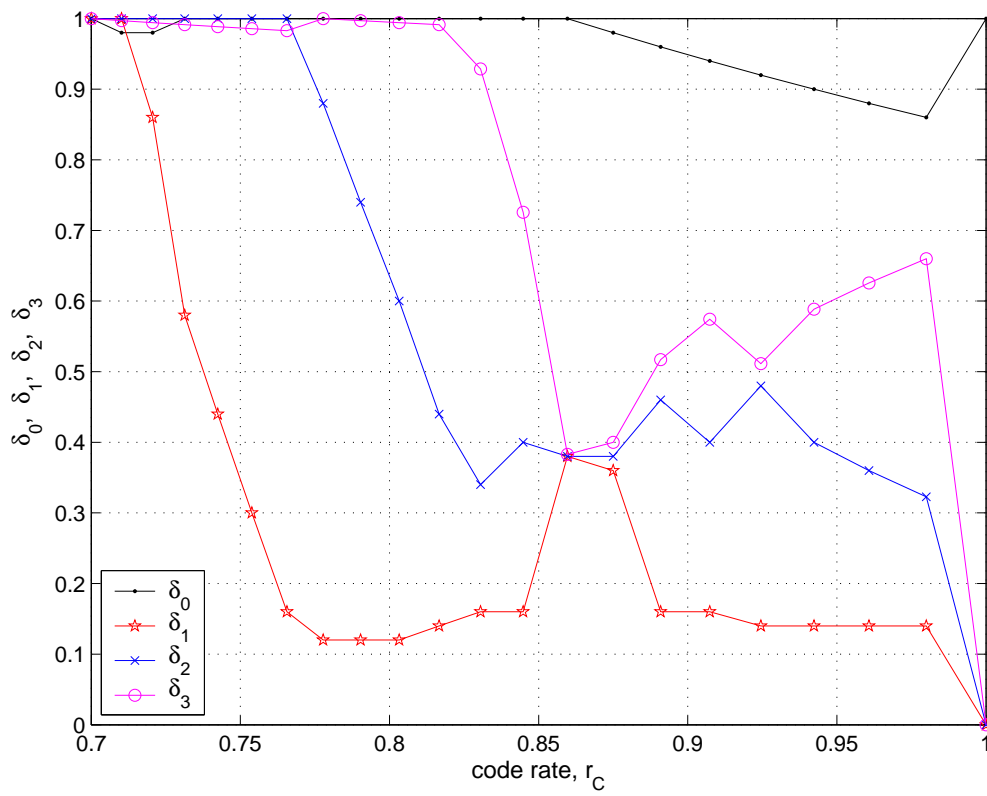


Figure A.11. Optimal puncturing ratios for the 3D PCZZ(8,7) code when the target BER is 10^{-5} .

References

- [1] C. M. Krishna and K. G. Shin, *Real-Time Systems*, McGraw-Hill, Singapore, 1997.
- [2] C. Ekelin, "An Optimization Framework for Scheduling of Embedded Real-Time systems," Ph.D. thesis, Department of Computer Engineering, Chalmers University of Technology, Göteborg, May 2004.
- [3] A. S. Tanenbaum, *Computer Networks*, 3 ed., Prentice-Hall, Inc., New Jersey, 1996.
- [4] N. Malcom and W. Zhao, "The timed-token protocol for real-time communications," *Computer*, vol. 27, no. 1, pp. 35-41, January 1994.
- [5] C. Venkatramani and T. Chiueh, "Supporting real-time traffic on the Ethernet," in *Proc. IEEE Real-Time Systems Symposium*, San Juan, Puerto Rico, December 1994, pp. 282-286.
- [6] H. Hoang, M. Jonsson, U. Hagström, and A. Kallerdahl, "Switched real-time Ethernet with earliest deadline first scheduling - protocols and traffic handling," in *Proc. Workshop on Parallel and Distributed Real-Time Systems*, Fort Lauderdale, FL, April 2002.
- [7] S. Chakrabarti and A. Mishra, "QoS issues in ad hoc wireless networks," *IEEE Communications Magazine*, vol. 39, no. 2, pp. 142-148, February 2001.
- [8] S. Ackmer, U. Bilstrup, and L. Svalmark, "Routing Protocol for Wireless Real-Time Multihop Networks," Master's Thesis CCA 9906, Halmstad University, Halmstad, Sweden, January 1999.
- [9] C. Heegard and S. B. Wicker, *Turbo Coding*, Kluwer Academic Press, 1999.
- [10] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379-423 and pp. 623-656, October 1948.
- [11] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Englewood Cliffs, N.J., 1983.
- [12] S. Lin, D. J. Costello, Jr., and M. J. Miller, "Automatic-repeat-request error control schemes," *IEEE Communications Magazine*, vol. 22, no. 12, pp. 5-16, December 1984.
- [13] J. M. Wozencraft and M. Horstein, "Digitalised communication over two-way channels," in *Proc. Fourth London Symposium on Information Theory*, London, U.K., September 1960.
- [14] P. Sindhu, "Retransmission error control with memory," *IEEE Transactions on Communications*, vol. 25, no. 5, pp. 423-429, May 1977.

- [15] D. Chase, "Code combining - a maximum-likelihood decoding approach for combining an arbitrary number of noisy packets," *IEEE Transactions on Communications*, vol. 33, no. 5, pp. 385-393, May 1985.
- [16] E. Uhlemann, "Hybrid ARQ using serially concatenated block codes for real-time communication - an iterative decoding approach," Lic. Eng. thesis, Chalmers University of Technology, Gothenburg, Sweden, October 2001.
- [17] E. Uhlemann, T. M. Aulin, L. K. Rasmussen, and P.-A. Wiberg, "Packet combining and doping in concatenated hybrid ARQ schemes using iterative decoding," in *Proc. IEEE Wireless Communications and Networking Conference*, New Orleans, LA, March 2003, pp. 849-854.
- [18] D. M. Mandelbaum, "An adaptive-feedback coding scheme using incremental redundancy," *IEEE Transactions on Information Theory*, vol. 20, no. 3, pp. 388-389, May 1974.
- [19] D. N. Rowitch and L. B. Milstein, "On the performance of hybrid FEC/ARQ systems using rate compatible punctured turbo (RCPT) codes," *IEEE Transactions on Communications*, vol. 48, no. 6, pp. 948-959, June 2000.
- [20] E. Malkamäki and H. Leib, "Performance of truncated type-II hybrid ARQ schemes with noisy feedback over block fading channels," *IEEE Transactions on Communications*, vol. 48, no. 9, pp. 1477-1487, September 2000.
- [21] C. Ladas, R. M. E. Amiee, M. Mahdavi, and G. A. Manson, "Class based selective-ARQ scheme for high performance TCP and UDP over wireless links," in *Proc. International Workshop on Mobile and Wireless Communications Network*, September 2002, pp. 311-315.
- [22] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: turbo codes," in *Proc. International Conference on Communications*, Geneva, Switzerland, May 1993, pp. 1064-1070.
- [23] G. D. Forney, Jr., *Concatenated Codes*, M.I.T. Press, Cambridge, MA, 1966.
- [24] A. C. Reid, T. A. Gulliver, and D. P. Taylor, "Convergence and errors in turbo-decoding," *IEEE Transactions on Communications*, vol. 49, no. 12, pp. 2045-2051, December 2001.
- [25] D. N. Rowitch and L. B. Milstein, "Rate compatible punctured turbo (RCPT) codes in a hybrid FEC/ARQ system," in *Proc. Communications Theory Mini-Conference of GLOBECOM '97*, Poenix, AZ, November 1997, pp. 55-59.
- [26] D. M. Rankin and T. A. Gulliver, "Single parity check product codes," *IEEE Transactions on Communications*, vol. 49, no. 8, pp. 1354-1362, August 2001.
- [27] S. Benedetto and G. Montorsi, "Unveiling turbo codes: some results on parallel concatenated coding schemes," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 409-428, March 1996.

- [28] 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*, vol. 44, no. 3, pp. 909-926, May 1998.
- [29] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Transactions on Communications*, vol. 49, pp. 1727-1737, October 2001.
- [30] F. Brännström, "Convergence Analysis and Design of Multiple Concatenated Codes," Ph.D. thesis, Department of Computer Engineering, Chalmers University of Technology, Göteborg, March 2004.
- [31] L. Ping, X. Huang, and N. Phamdo, "Zigzag codes and concatenated zigzag codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 800-807, February 2001.
- [32] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, Prentice Hall Inc., Englewood Cliffs, NJ, 1995.
- [33] J. M. Wozencraft and I. M. Jacobs, *Principles of Communication Engineering*, John Wiley & Sons, New York, NY, 1965.
- [34] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 429-445, March 1996.
- [35] L. K. Rasmussen, "Trellis Coded, Adaptive Rate Hybrid-ARQ Protocols over AWGN Channels and Slowly Fading Rician Channels," Ph.D. dissertation, School of Electrical Engineering, Georgia Institute of Technology, Atlanta, GA, September 1993.
- [36] H. Yamamoto and K. Itoh, "Viterbi decoding algorithm for convolutional codes with repeat request," *IEEE Transactions on Information Theory*, vol. 26, no. 5, pp. 540-547, September 1980.
- [37] B. A. Harvey and S. B. Wicker, "Packet Combining Systems Based on the Viterbi Decoder," *IEEE Transactions on Communications*, vol. 42, no. 2/3/4, pp. 1544-1557, February/March/April 1994.
- [38] G. D. Forney, Jr., "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268-278, March 1973.
- [39] S. B. Wicker, "High-reliability data transfer over land mobile radio channel using interleaved hybrid-ARQ error control," *IEEE Transactions on Vehicular Technology*, vol. 39, no. 1, pp. 48-55, February 1990.
- [40] S. B. Wicker, "Reed-Solomon error control coding for data transmission over Rayleigh fading channels with feedback," *IEEE Transactions on Vehicular Technology*, vol. 41, no. 2, pp. 124-133, May 1992.

- [41] L. K. Rasmussen, M. J. Bartz, and S. B. Wicker, "A comparison of trellis coded and Reed-Solomon coded hybrid-ARQ protocols over slowly fading Rayleigh channels," *Kluwer Journal Wireless Personal Communications*, vol. 2, no. 4, pp. 393-413, 1996.
- [42] T. L. Tapp, A. A. Luna, X. Wang, and S. B. Wicker, "Extended Hamming and BCH soft decision decoders for mobile data applications," *IEEE Transactions on Communications*, vol. 47, no. 3, pp. 333-337, March 1999.
- [43] E. Uhlemann, T. M. Aulin, L. K. Rasmussen, and P.-A. Wiberg, "Deadline dependent coding - a framework for wireless real-time communication," in *Proc. International Conference on Real-Time Computing Systems and Applications*, Cheju Island, South Korea, December 2000, pp. 135-142.
- [44] G. Benelli, "An ARQ scheme with memory and sort error detectors," *IEEE Transactions on Communications*, vol. 33, no. 3, pp. 285-288, March 1985.
- [45] S. Lin and P. S. Yu, "A hybrid ARQ scheme with parity retransmission for error control of satellite channels," *IEEE Transactions on Communications*, vol. 30, no. 7, pp. 1701-1719, July 1982.
- [46] S. B. Wicker and M. J. Bartz, "Type-II hybrid-ARQ protocols using punctured MDS codes," *IEEE Transactions on Communications*, vol. 42, no. 2/3/4, pp. 1431-1440, February/March/April 1994.
- [47] E. Uhlemann, T. M. Aulin, L. K. Rasmussen, and P.-A. Wiberg, "Hybrid ARQ based on serially concatenated block codes using iterative decoding for real-time communication," in *Proc. Radiovetenskap och Kommunikation*, Stockholm, Sweden, June 2002, pp. 517-521.
- [48] J. G. Proakis, *Digital Communications*, 3rd ed., McGraw-Hill, New York, NY, 1995.
- [49] S. Kallel, "Complementary punctured convolutional (CPC) codes and their applications," *IEEE Transactions on Communications*, vol. 43, no. 6, pp. 2005-2009, June 1995.
- [50] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Transactions on Communications*, vol. 36, no. 4, pp. 389-400, April 1988.
- [51] J.-F. Cheng, Y.-P. Wang, and S. Parkvall, "Adaptive incremental redundancy," in *Proc. IEEE Vehicular Technology Conference Fall 2003*, Orlando, FL, October 2003, pp. 737-741.
- [52] B. Stender and M. Weckerle, "A Semi Implicit Incremental Redundancy Scheme," in *Proc. IST Mobile & Wireless Communications Summit*, Aveiro, Portugal, June 2003, pp. 583-587.
- [53] G. D. Forney, Jr., "Coset codes II: Binary lattices and related codes," *IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 1152-1187, September 1988.

- [54] K. R. Narayanan and G. L. Stüber, "A novel ARQ technique using the turbo coding principle," *IEEE Communications Letters*, vol. 1, no. 2, pp. 49-51, March 1997.
- [55] W.-C. Chan, E. Geraniotis, and V. D. Nguyen, "An adaptive hybrid FEC/ARQ protocol using turbo codes," in *Proc. IEEE International Conference on Universal Personal Communications Record*, San Diego, CA, October 1997, pp. 541-545.
- [56] J. S. Sadowsky, "A maximum likelihood decoding algorithm for turbo codes," in *Proc. IEEE Global Telecommunications Conference, GLOBECOM '97*, Phoenix, AZ, November 1997, pp. 929-933.
- [57] C.-F. Law, C. H. Lau, and T.-M. Ko, "A modified adaptive hybrid FEC/ARQ protocol using turbo codes with incremental redundancy transmission," in *Proc. IEEE Vehicular Technology Conference*, Amsterdam, Netherlands, September 1999, pp. 1670-1674.
- [58] J. Hámorský, U. Wachsmann, J. B. Huber, and A. Cižmár, "Hybrid automatic repeat request scheme with turbo codes," in *Proc. International Symposium on Turbo Codes & Related Topics*, Brest, France, September 1997, pp. 247-250.
- [59] J. Hámorský and U. Wachsmann, "Criteria for minimum bit error rate decoding of turbo codes," in *Proc. Kleinheubacher Tagung*, Kleinheubach, Germany, October 1995, pp. 607-616.
- [60] M. E. Buckley and S. B. Wicker, "A neural network for predicting decoder error in turbo decoders," *IEEE Communications Letters*, vol. 3, no. 5, pp. 145 -147, May 1999.
- [61] M. E. Buckley and S. B. Wicker, "The design and performance of a neural network for predicting turbo decoding error with application to hybrid ARQ protocols," *IEEE Transactions on Communications*, vol. 48, no. 4, pp. 566-576, April 2000.
- [62] Z. Chi, Z. Wang, and K. K. Parhi, "High throughput low energy FEC/ARQ technique for short frame turbo codes," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '00*, Istanbul, Turkey, June 2000, pp. 2653-2656.
- [63] P. Coulton, C. Tanriover, B. Wright, and B. Honary, "Simple hybrid type-II ARQ technique using soft output information," *Electronics Letters*, vol. 36, no. 20, pp. 1716-1717, September 2000.
- [64] E. Visotsky, V. Tripathi, and M. Honig, "Optimum ARQ design: A dynamic programming approach," in *Proc. IEEE International Symposium on Information Theory*, Yokohama, Japan, June 2003, pp. 451.
- [65] V. Tripathi, E. Visotsky, R. Peterson, and M. Honig, "Reliability-based type II hybrid ARQ schemes," in *Proc. IEEE International Conference on Communications*, May 2003, pp. 2899-2903.
- [66] A. S. Barbulescu and S. S. Pietrobon, "Rate compatible turbo codes," *Electronics Letters*, vol. 31, pp. 535-536, Mars 1995.

- [67] P. Jung, J. Plechinger, M. Doetsch, and F. M. Berens, "A pragmatic approach to rate compatible punctured turbo-codes for mobile radio applications," in *Proc. International Conference on Advances in Communications and Control*, Corfu, Greece, June 1997.
- [68] P. Jung and J. Plechinger, "Performance of rate compatible punctured turbo-codes for mobile radio applications," *Electronics Letters*, vol. 33, no. 25, pp. 2102-2103, December 1997.
- [69] J. Li and H. Imai, "Performance of hybrid-ARQ protocols with rate compatible turbo codes," in *Proc. International Symposium on Turbo Codes & Related Topics*, Brest, France, September 1997, pp. 188-191.
- [70] T. Ji and W. E. Stark, "Concatenated punctured turbo Reed-Solomon codes in a hybrid FEC/ARQ DS/SSMA data network," in *Proc. IEEE Vehicular Technology Conference*, Houston, TX, May 1999, pp. 1678 -1682.
- [71] R. Mantha and F. R. Kschischang, "A capacity-approaching hybrid ARQ scheme using turbo codes," in *Proc. IEEE Global Telecommunications Conference, GLOBECOM '99*, Rio de Janeiro, Brazil, December 1999, pp. 2341-2345.
- [72] T. Ji and W. E. Stark, "Rate compatible product codes," in *Proc. 21st Century Military Communications Conference, MILCOM 2000*, Los Angeles, CA, October 2000, pp. 412-416.
- [73] Y. Wu and M. C. Valenti, "An ARQ technique using related parallel and serial concatenated convolutional codes," in *Proc. IEEE International Conference on Communication, ICC'00*, New Orleans, LA, June 2000, pp. 1390-1394.
- [74] N. Chandran and M. C. Valenti, "Hybrid ARQ using serial concatenated convolutional codes over fading channels," in *Proc. IEEE Vehicular Technology Conference, VTC'01*, Rhodes, Greece, May 2001, pp. 1410-1414.
- [75] A. Banerjee, D. J. Costello, Jr., and T. E. Fuja, "Bandwidth efficient hybrid ARQ schemes using turbo codes," in *Proc. IEEE International Symposium on Information Theory*, Sorrento, Italy, June 2000, pp. 188.
- [76] A. Banerjee, D. J. Costello, Jr., and T. E. Fuja, "Diversity combining techniques for bandwidth-efficient turbo ARQ systems," in *Proc. IEEE International Symposium on Information Theory, ISIT'01*, Washington D.C., June 2001, pp. 213.
- [77] J. Hámorský and L. Hanzo, "Performance of the turbo hybrid automatic repeat request system type-II," in *Proc. IEEE Information Theory and Networking Workshop, ITW '99*, Metsovo, Greece, July 1999, pp. 51.
- [78] Ö. F. Açikel and W. E. Ryan, "Punctured turbo-codes for BPSK/QPSK channels," *IEEE Transactions on Communications*, vol. 47, no. 9, pp. 1315-1323, September 1999.
- [79] S. Benedetto, R. Garello, and G. Montorsi, "A search for good convolutional codes to be used in the construction of turbo codes," *IEEE Transactions on Communications*, vol. 46, no. 9, September 1998.

- [80] F. Babich, G. Montorsi, and F. Vatta, "Design of rate-compatible punctured turbo (RCPT) codes," in *Proc. IEEE International Conference on Communications, ICC'02*, New York City, NY, April 2002, pp. 1701-1705.
- [81] R. Liu, P. Spasojevic, and E. Soljanin, "Punctured turbo code ensembles," in *Proc. IEEE Information Theory Workshop*, Paris, France, March 2003.
- [82] R. Liu, P. Spasojevic, and E. Soljanin, "On the role of puncturing in hybrid ARQ schemes," in *Proc. IEEE International Symposium on Information Theory*, Yokohama, Japan, June 2003, pp. 449.
- [83] E. Uhlemann, L. K. Rasmussen, A. J. Grant, and P.-A. Wiberg, "Optimal incremental-redundancy strategy for type-II hybrid ARQ," in *Proc. IEEE International Symposium on Information Theory*, Yokohama, Japan, June 2003, pp. 448.
- [84] E. Uhlemann, L. K. Rasmussen, A. J. Grant, and P.-A. Wiberg, "Optimal type-II concatenated hybrid ARQ using single parity check codes," in *Proc. International Symposium on Turbo Codes & Related Topics*, Brest, France, September 2003, pp. 587-590.
- [85] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 33, no. 6, pp. 457-458, March 1997.
- [86] D. M. Rankin and T. A. Gulliver, "Randomly interleaved SPC product codes," in *Proc. IEEE International Symposium on Information Theory*, Sorrento, Italy, June 2000, pp. 88.
- [87] L. Ping, S. Chan, and K. L. Yeung, "Iterative decoding of multi-dimensional concatenated single parity check codes," in *Proc. IEEE International Conference on Communications*, Atlanta, GA, June 1998, pp. 131-135.
- [88] J. S. K. Tee and D. P. Taylor, "Multiple parallel concatenated single parity-check codes," in *Proc. IEEE International Conference on Communications*, Helsinki, Finland, June 2001, pp. 60-64.
- [89] J. S. K. Tee and D. P. Taylor, "Multiple serially concatenated single parity-check codes," in *Proc. IEEE International Conference on Communications*, New Orleans, LA, June 2000, pp. 613-617.
- [90] J. S. K. Tee, D. P. Taylor, and P. A. Martin, "Multiple serial and parallel concatenated single parity-check codes," *IEEE Transactions on Communications*, vol. 51, no. 10, pp. 1666-1675, October 2003.
- [91] J. M. Shea and T. F. Wong, "Reduced-complexity decoding for concatenated codes based on rectangular parity-check codes and turbo codes," in *Proc. IEEE Global Telecommunications Conference*, San Antonio, TX, November 2001, pp. 1031-1035.
- [92] J. M. Shea and T. F. Wong, "Concatenated codes based on multidimensional parity-check codes and turbo codes," in *Proc. IEEE Military Communications Conference*, Washington, D.C., October 2001, pp. 1152-1156.

- [93] J. S. K. Tee and D. P. Taylor, "Iterative decoding of systematic binary algebraic block codes," in *Proc. IEEE Global Telecommunications Conference*, San Francisco, CA, November 2000, pp. 842-846.
- [94] L. Ping, "Turbo-SPC codes," *IEEE Transactions on Communications*, vol. 49, no. 5, pp. 754-759, May 2001.
- [95] M. Ferrari and S. Bellini, "Existence and uniqueness of the solution for turbo decoding of parallel concatenated single parity check codes," *IEEE Transactions on Information Theory*, vol. 49, no. 3, pp. 722-726, March 2003.
- [96] P. Elias, "Error free coding," *IRE Transactions on Information Theory*, vol. 4, pp. 29-37, September 1954.
- [97] S. Dolinar and D. Divsalar, "Weight distributions for turbo codes using random and nonrandom permutations," *TDA Progress Report 42-122*, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, pp. 56-65, August 1995.
- [98] D. M. Rankin and T. A. Gulliver, "Asymptotic performance of product codes," in *Proc. IEEE International Conference on Communications*, Vancouver, B.C., Canada, June 1999, pp. 431-435.
- [99] F. Brännström, L. K. Rasmussen, and A. J. Grant, "Optimal scheduling for iterative decoding," in *Proc. IEEE International Symposium on Information Theory*, Yokohama, Japan, June 2003, pp. 350.
- [100] X. Huang, N. Phamdo, and L. Ping, "BER bounds on parallel concatenated single parity check arrays and zigzag codes," in *Proc. IEEE Global Telecommunications Conference*, Rio de Janeiro, Brazil, December 1999, pp. 2436-2440.
- [101] I. Land and P. Hoeher, "Partially systematic rate 1/2 turbo codes," in *Proc. International Symposium on Turbo Codes and Related Topics*, Brest, France, September 2000, pp. 287-290.
- [102] S. S. Pietrobon, "On punctured serially concatenated turbo codes," in *Proc. Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, November 2001, pp. 265-269.
- [103] F. Babich, G. Montorsi, and F. Vatta, "Rate-compatible punctured serial concatenated convolutional codes," in *Proc. IEEE Global Telecommunications Conference, GLOBECOM '03*, San Francisco, CA, December 2003, pp. 2062-2066.
- [104] F. Brännström, L. K. Rasmussen, and A. J. Grant, "Optimal scheduling for multiple serially concatenated codes," in *Proc. International Symposium on Turbo Codes and Related Topics*, Brest, France, September 2003, pp. 383-386.
- [105] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, pp. 308-313, 1965.
- [106] L. Ping and N. Phamdo, "Zigzag codes and concatenated zigzag codes," in *Proc. IEEE Information Theory and Networking Workshop*, Metsovo, Greece, July 1999, pp. 70.

- [107] X. Wu, Y. Xue, and H. Xiang, "On concatenated zigzag codes and their decoding schemes," *IEEE Communications Letters*, vol. 8, no. 1, pp. 54-56, January 2004.
- [108] K. S. Chan, L. Ping, and S. Chan, "Adaptive type II hybrid ARQ scheme using zigzag code," *Electronics Letters*, vol. 35, no. 24, pp. 2102-2104, November 1999.