Practical Anytime Codes

LEEFKE GROSJEAN

Doctoral Thesis
Stockholm, Sweden 2016

KTH School of Electrical Engineering
SE-100 44 Stockholm
SWEDEN

ISBN 978-91-7595-937-5

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av teknologie doktorsexamen i telekommunication den 13 Maj 2016 klockan 10:00 i Sal F3, Lindstedtsvägen 26, Stockholm.

Tryck: Universitetsservice US AB

Abstract

The demand of an increasingly networked world is well reflected in modern industrial control systems where communication between the different components of the system is more and more taking place over a network. With an increasing number of components communicating and with hardware devices of low complexity, the communication resources available per communication link are however very limited. Yet, despite limited resources, the control signals transmitted over the link are still required to meet strict real-time and reliability constraints. This requires entirely new approaches in the intersection of communication and control theory. In this thesis we consider the problem of stabilizing an unstable linear-quadratic-Gaussian (LQG) plant when the communication link between the observer and the controller of the plant is noisy. Protecting the data transmitted between these components against transmission errors by using error control schemes is essential in this context and the main subject to this thesis. We propose novel error-correcting codes, so-called anytime codes, for this purpose and show that they asymptotically fulfill the reliability requirements known from theory when used for transmission over the binary erasure channel (BEC). We identify fundamental problems when the messages to be transmitted are very short and/or the communication channel quality is very low. We propose a combinatorial finite-length analysis which allows us to identify important parameters for a proper design of anytime codes. Various modifications of the basic code structure are explored, demonstrating the flexibility of the codes and the capability of the codes to be adapted to different practical constraints. To cope with communication channels of low quality, different feedback protocols are proposed for the BEC and the AWGN channel that together with the error-correcting codes ensure the reliability constraints at short delays even for very short message lengths. In the last part of this thesis, we integrate the proposed anytime codes in an automatic control setup. We specify the different components necessary for this and determine the control cost when controlling an unstable LQG plant over a BEC using either the anytime codes proposed in this thesis or block codes. We detail the relation between parameters such as channel quality, code rate, plant instability and resources available and highlight the advantage of using anytime codes in this context. Throughout the thesis, the performance of the anytime codes is evaluated using asymptotic analysis, finite-length analysis and/or simulation results.

Sammanfattning

Efterfrågan av en alltmer ihopkopplad värld återspeglas bland annat i moderna industriella styrsystem där kommunikationen mellan de olika komponenterna allt oftare sker över ett nätverk. Eftersom antalet komponenter som kommunicerar med varandra ökar, medans hårdvaruenheternas komplexitet är fortsatt låg, blir kommunikationsresurserna per kommunikationslänk alltmer begränsade. Trots detta måste styrsignalerna som skickas över länken uppfylla strikta krav på tillåtna tidsfördröjningar och nödvändig tillförlitlighet. Detta kräver helt nya metoder i gränssnittet mellan kommunikationsteknik och reglerteknik. I denna avhandlingen undersöker vi problemet med att stabilisera ett instabilt linjärt kvadratiskt Gaussiskt (LQG) system när kommunikationslänken mellan observatören och regulatorn är störd av brus. Att skydda styrsignalerna mot störningar i kommunikationskanalen med hjälp av felkorrigerande koder är viktigt i detta sammanhang och är huvudtemat för denna avhandlingen. Vi föreslår nya felkorrigerande koder, så kallade anytime-koder, och visar att de asymptotiskt uppnår kraven på tillförlitlighet från teorin för dataöverförning över en binär raderingskanal (BEC). Vi identifierar grundläggande problem när meddelandena som ska skickas är väldigt korta eller om kommunikationskanalen är av dålig kvalitet. Vi föreslår en kombinatorisk analys för meddelanden med ändlig blocklängd som tillåter oss att identifiera viktiga konstruktionsparametrar. Olika modifieringar av den grundläggande kodstrukturen undersöks som påvisar flexibiliteten hos koderna och möjligheten att anpassa koderna till diverse praktiska begränsningar. För att även använda koderna för kommunikationskanaler med mycket brus föreslår vi olika återkopplingsprotokoll som tillämpas vid överförning över BEC eller AWGN kanalen. I sista delen av avhandlingen integrerar vid de föreslagna koderna i ett reglertekniskt sammanhang. Vi specificerar de olika komponenterna och beräknar kontrollkostnaden för ett scenario där vi ska stabilisera ett instabilt LQG system och kommunikationslänken mellan observatören och regulatorn modelleras som en BEC kanal. Det föregående görs för både de föreslagna koderna och för blockkoder. Vi specificerar sambandet mellan parametrarna såsom kanalkvalitet, kodhastighet, instabilitet, och tillgängliga resurser. Dessutom lyfter vi fram fördelarna med att använda anytime-koder för detta sammanhang. Genom hela avhandlingen utvärderas kodernas prestanda med hjälp av asymptotisk analys, ändlig-blocklängdsanalys och/eller simuleringar.

Acknowledgements

Now that this five-year journey of my Ph.D. studies comes to an end, I would like to thank all the people that have supported me during this time. I would like to express my sincere gratitude to Prof. Lars Kildehøj Rasmussen, Associate Prof. Ragnar Thobaben, and Prof. Mikael Skoglund. Mikael gave me the opportunity to join the Communication Theory department and under the supervision of Lars and Ragnar I started to explore the academic world. I am deeply grateful to Lars and Ragnar for their guidance, for their research insights, their enthusiastic encouragement, and the independence they gave me when conducting my Ph.D. studies. Discussions with them have always been interesting, challenging, stimulating and as well lots of fun. Moreover I am truly thankful to them for their contribution in making this an open-minded, flexible and solution-driven workplace. Having had two children during my time at the department, I highly appreciate how smooth it went to combine this with my Ph.D. studies.

I am honored to have collaborated with Associate Prof. Joakim Jaldén and Associate Prof. Mats Bengtsson in teaching. I thank all my current and former colleagues from Floor 3 and 4 for creating such a nice working atmosphere. In particular, I would like to mention Dr. Kittipong Kittichokechai, Dr. Nicolas Schrammar, Dr. Isaac Skog, Dr. Dave Zachariah, Dr. Mattias Andersson, Dr. Ricardo Blasco Serrano, and Dr. Dennis Sundman. Special thanks to Raine Tiivel for her diligence in taking care of the administrative issues and Joakim Jaldén for doing the quality review of this thesis.

I would like to thank Associate Prof. Michael Lentmaier for taking the time to act as faculty opponent and Prof. Catherine Douillard, Dr. Ingmar Land and Associate Prof. Carlo Fischione for acting as grading committee.

I want to thank my parents Meike and Olaf and my sisters Kerrin and Jomtje for their great support from far away. Last but not least, I want to thank my husband Julien and my children for filling every day with love and happiness.

Leefke Grosjean
Stockholm, April 2016

Contents

Chapter 1

Introduction

In the past twenty years, we have witnessed an unprecedented increase of devices connected to the Internet. From an estimated 6.4 billion connected devices in 2016 this number is expected to increase to 20.7 billion devices connected to the Internet by 2020 [Gar15]. The progress so far has been mainly due to the improved communication infrastructure, and the rapid advances in embedded systems technology making low-cost sensors and small smart devices available. Connecting components via a shared network is however only the first step. The next logical step in this evolution is to make use of the monitoring and data collecting capabilities and to take actions in an automated manner. While social systems have not come so far a major trend in modern industrial systems is already to have control loops being closed over a common communication network (e.g. smart cities, smart transportation). This is where the new paradigm of Networked Control Systems (NCS) comes into play. A networked control system is a spatially distributed system in which the communication between the different components occurs through a shared network, as shown in Fig. 1.1. The use of wireless networks as a medium to interconnect the different components in an industrial control system has many advantages, such as enhanced resource utilization, reduced wiring, easier diagnosis and maintenance and reconfigurability. Not surprisingly NCSs have therefore already found application in a broad range of areas. These include among others remote surgery [MWC+04], automated highway systems and unmanned aerial vehicles [SS01, SS05]. Consider for instance automobiles. Cars are already nowadays equipped with a wide range of electronic services ranging from Global Position System (GPS), in-vehicle safety and security systems, to entertainment systems and drive-by-wire systems. But the number of services is expected to grow. All applications typically consist of sensors, actuators and controllers that are spatially distributed over the entire vehicle, with over 50 embedded computers running various applications. Having communication between these components taking place only over wired connections is sooner or later going to be unsustainable and in-vehicle networking will make use of many ideas developed in the field of networked control [SV03, LH02].

Using a shared network instead of direct connections in control loops introduces

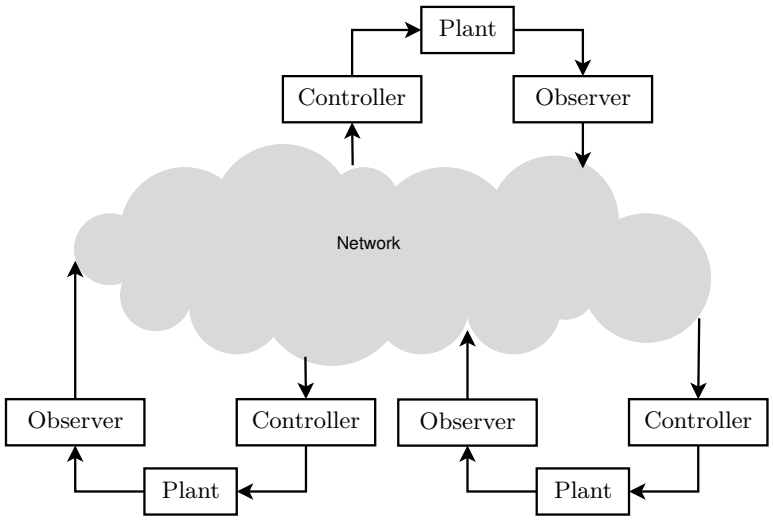


Figure 1.1: Networked control system.

new challenges however. With an increased number of components that communicate with each other the resources available per connection become increasingly limited and neither the underlying control problem nor the communication problem can be solved independently from each other.

The analysis of NCSs requires therefore a combined approach using both tools from control theory and communication theory. Traditionally these have been areas with little common ground. A standard assumption in classical control theory is that communication is instantaneous, reliable and with infinite precision. In classical information theory, in contrast, the focus is on transmitting information reliably over imperfect channels, largely one-way communication, where the specific purpose of the transmitted information is relatively unimportant. Whereas control theory deals with real-time constraints, many results in information theory are valid only in the asymptotic limit of large delay.

The combined approach necessary for analyzing NCSs opened up an entirely new field of research, which from the control theoretic perspective is explored by taking into account communication constraints in the control setup, whereas from the communication theoretic perspective the focus is on formulating the constraints on the communication channel and encoder/decoder protocol when the purpose is to control a single plant. The NCS pictured in Fig. 1.1 is for this purpose decomposed and only a single control loop is considered. The presence of the other control loops using the same shared networked is reflected in the severe restrictions on the communication resources available for the considered control loop. This is the approach that we take in this thesis. In the following we give a brief overview of the system and describe the problem among the many challenges within the field

of NCSs that we want to address.

1.1 System Overview

Fig. 1.2 shows the system overview of the setup that we consider in this thesis. It consists of a plant that is observed by an observer in noise. The observed signal

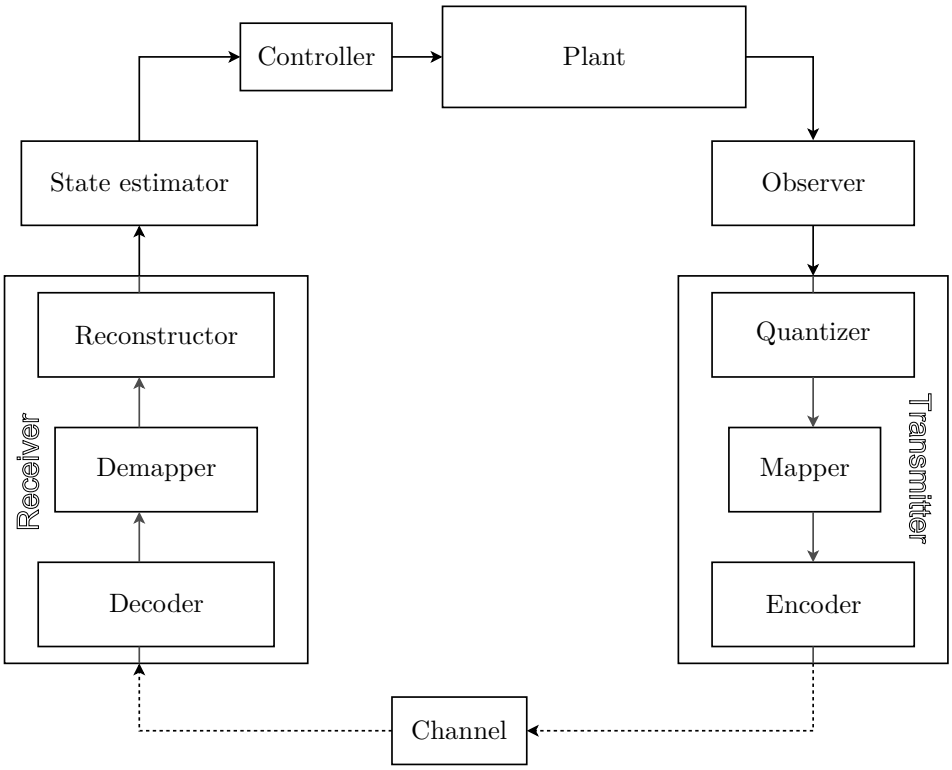


Figure 1.2: System overview.

is converted into a binary sequence by a quantizer followed by a mapper. The resulting message is encoded by the encoder, modulated and transmitted over the noisy channel. At the receiver side the corrupted signal is demodulated, decoded, demapped and reconstructed. Based on the information available the state estimator produces an estimate of the current state of the system and the controller applies a suitable control signal to the plant. The system evolves over time and the entire process is repeated for every time step. The plant is assumed to be linear time-invariant (LTI) and prone to process noise. Process noise and measurement noise are such that the resulting control problem is Linear-Quadratic-Gaussian (LQG). The details of the setup become clear throughout the thesis.

1.2 Problem Formulation

Error-correcting codes are an essential part of almost all digital communication systems nowadays. Without an encoder adding redundancy in a structured manner to the message to be transmitted over a noisy channel the receiver is often not capable of retrieving the content of the message properly. The challenge of developing error-correcting codes for application in a control loop is that these codes have to meet real-time constraints and strict reliability requirements simultaneously. In such a case classical concepts are often not applicable: It turns out that developing capacity-achieving codes is not sufficient as Shannon capacity is not the right figure of merit. The classical Shannon capacity was introduced and developed in Shannon's landmark paper [Sha48] in 1948. It is the supremum of the rate at which blocks of data can be communicated over a channel with arbitrarily small probability of error. In order to transmit reliably we therefore need to break up a message into blocks of k bits each and add redundancy by encoding each block independently into a larger block of n bits. Shannon showed that as long as the rate defined as $R = k/n$ is smaller than the channel capacity there exist block codes with which we can transmit the data over the channel with a probability of decoding error that vanishes to zero. This existence proof triggered the search for practical codes that reach the Shannon limit but at the same time can be decoded with affordable complexity. Several practical codes of that kind have been found over the last years including for example low density parity check (LDPC) codes when decoded with message-passing decoding, algebraic geometric codes when decoded with Berlekamp-Massey or list decoding. A common assumption for all these codes is that an encoding and decoding delay is not a problem when reliability is achieved. It is however precisely this assumption which is typically no longer valid when developing channel codes for NCSs. The notion of channel capacity is therefore not the right figure of merit in the context of NCSs and instead it is the so called *anytime capacity*. Anytime capacity is a new information theoretic notion that together with *anytime reliability* was introduced by Sahai [Sah01] and Mitter [SM06] together with the new framework of *anytime information theory*. Anytime capacity is the rate at which data can be transmitted over a channel such that the probability of decoding error is arbitrarily small and moreover decays exponentially fast with the decoding delay. Hence, the behavior over delay is explicitly taken into account. The related concept of anytime reliability characterizes the decay of the probability of error in terms of the anytime exponent. An error correcting code is called (R, α)-anytime reliable if the probability of decoding a bit sent d time steps in the past incorrectly decays exponentially in d with error exponent α.

The communication problem embedded in Fig. 1.2 emerging from the previous discussion is therefore to develop error-correcting codes that can be shown to have anytime characteristics.

1.3 Literature Review

As already indicated previously, the problem of stabilizing an unstable plant over a noisy communication link is addressed from the communication theoretical angle and the control theoretical angle.

Approaching the Problem with Communication Theory

Coming from an information/communication theoretical perspective, a new framework called *anytime information theory* was created [Sah01]. Within this framework results involve sufficient conditions on the rate R and anytime exponent α of error-correcting codes that ensure stability [Sah01, SM06]. Sufficient conditions for noiseless but rate limited communication channels were presented in [MDE06, MA07, MFDN09, NE02, TSM04]. The results are often asymptotic in nature, meaning that the second moment of the state is guaranteed to be finite for all times but it can be arbitrarily large. Moreover the messages transmitted over the channel are not necessarily in form of packets. However, messages are almost always considered to be within a finite field. Building on these results the problem of finding practical error correcting codes can then be restricted to the communication problem embedded in Fig. 1.2, consisting of the encoder, channel and the decoder when taking into account special requirements. The key properties of error control codes intended for controlling open-loop unstable systems are *causality* and *anytime reliability*, as described in the framework of anytime theory. We refer to a channel code as having anytime-reliable behavior if the probability of incorrect decoding of a block of the code decays exponentially with the decoding delay. The decay is characterized by the so-called anytime exponent. Due to the nature of systems control, information is only available causally and thus anytime codes are necessarily tree codes. Schulman showed that a class of nonlinear tree codes exists for interactive communication [Sch96]. The *trajectory codes* developed in [ORS05] as well as the *potent tree codes* developed in [GMS11] build on the work by Schulman; however, no deterministic constructions are known for these codes in general. For linear unstable systems with an exponential growth rate, encoding and decoding schemes have been developed for stabilizing such processes over the binary erasure channel (BEC) [Yuk09] and the binary symmetric channel (BSC) [SJV04]. However the state of the unstable process must be known at the encoder. In [SH11b] linear encoding schemes are explored for the general case where the state of the unstable linear process is only available through noisy measurements. It is shown in [SH11b, SH11a] that linear anytime-reliable codes exist with high probability, and furthermore they propose a simple constructive maximum-likelihood decoding algorithm for transmission over the BEC. In [CFZ10] it is shown that in general random linear tree codes are anytime reliable. A first result on practical anytime codes is presented in our work in [DRTS12], where protograph-based LDPC-Convolutional Codes (LDPC-CCs) are shown to have anytime properties asymptotically under message-passing decoding. Tarable et al. [TNDT13a] generalized our code struc-

ture and showed that the proposed codes achieve anytime reliability even on the additive white Gaussian noise (AWGN) channel. The LDPC-CCs were further analyzed in our work in [GRTS14] with a focus on the finite-length behavior. In [NARNL13] a class of spatially coupled codes was investigated in the context of anytime transmission and also shown to have anytime properties asymptotically. The code construction is based on a modified version of the randomized ensemble from [KRU11]. While in [KRU11] a uniform distribution is used to determine the connections from variable nodes at one position to check nodes at neighboring positions, the authors in [NARNL13] employ an exponential distribution. Furthermore an analysis approach is proposed in [NARNL15] to approximate the behavior for finite block lengths.

Approaching the Problem with Control Theory

Coming from a control theoretical perspective, the problem of stabilizing an unstable plant over a noisy communication link is addressed from a different angle. The control problem over a noise free channel is assumed to be largely solved. So research focuses on integrating channel constraints and/or quantization noise into the problem. There are many different approaches in literature on how the channel between observer and controller is modeled. For a signal-to-noise ratio (SNR)-constrained AWGN channel [SGQ10] analyzed the conditions on the minimum channel quality to achieve stability. Here the common approach is taken that assumes that the channel over which the system is closed is analog, thus allowing the transmission of real values. Quantization noise is then modeled as additive white Gaussian noise. The problem was studied for channels with random packet losses in e.g. [SSF$^+$03, GSHM05, IYB06]. More advanced channel models take into account correlations [MGS13]. There is an extensive literature on the minimum bit-rate that is needed to stabilize a system through feedback, e.g. [WB99, EM01, YB04, NE04, TSM04]. Many contributions have been devoted to the design of networked control systems considering the effects of delays introduced by the network (see [Nil98] and [Zam08] for a survey). There is however no research involving the actual delays introduced by employing different coding schemes to secure the data transmission over the channel, e.g. the length of the codewords transmitted over the channel.

Combining the Approaches

In Chapter 6 of this thesis we build on work where data losses are modeled with a Bernoulli process where packets of data are lost independently with a certain probability. It is a model chosen mainly for mathematical tractability. The channel considered is however better captured by the approach originally proposed in [BM97] where noise and bandwidth limitations in the communication channels are captured by modeling the channels as *bit pipes*, meaning that each channel is only capable of transmitting a fixed number n_{max} of bits in each time slot of the sys-

tem's evolution. While the authors of [BM97] considered the LQG control of *stable* dynamic systems, the extension to LQG control of unstable systems and to other kinds of channel models were made in [GDH+09, GSHM05, Sch96, TSM04]. In the literature two different protocols are typically considered: A protocol where packets are acknowledged at the receiver (e.g. Transmission Control Protocol - TCP) or a protocol where no such information is available (e.g. User Datagram Protocol - UDP). In Chapter 6 we consider only UDP-like protocols. For such a channel model it was shown in [SSF+03] that an unstable plant cannot be stabilized if the packet erasure rate exceeds a threshold that is determined by the plant dynamics. It was shown in [SSF+05a] that for TCP-like protocols the separation principle holds, meaning that the optimal control signal can be found by treating the estimation and control problem independently. Moreover it was shown that the optimal control is a linear function of the state. The controller is then formed by a time-varying Kalman filter and a linear-quadratic regulator. For UDP-like protocols the separation principle does not hold in general and the optimal LQG controller is in general nonlinear [SSF+05a]. By cascading state estimator and controller and using a *time-invariant* Kalman-like filter it was however shown in [CLSZ13] that a computationally simple system can be obtained. And even though the solution does not yield the optimal time-varying Kalman filter it allows for the explicit computation of the performance. The results presented in [CLSZ13] take into account delay and SNR-limitations. Other contributions taking into account multiple channel limitations, such as, packet loss, bit erasures, quantization errors at the same time are for instance presented in [ITQ11, TIH09].

1.4 Outline and Contributions of the Thesis

In the following, we briefly review the contents of each chapter and informally state the main contributions.

Chapter 2

As an introduction to the topics studied in this thesis, we provide an overview of the basic concepts behind anytime information theory, error-correcting codes and some aspects of control theory. First we introduce the communication system including the communication channels relevant to this thesis. We explain channel capacity as introduced by Shannon followed by the concept of anytime capacity and anytime reliability as introduced by Sahai and Mitter. Sufficient conditions for stabilizing an unstable plant over a noisy communication link are presented and the LQG control setup considered in this thesis is defined. Thereafter we give a short overview about the basic ideas and analysis tools of LDPC codes and particularly LDPC convolutional codes.

Chapter 3

In this chapter we focus entirely on the communication problem embedded in Fig. 1.2, consisting of the encoder, channel and the decoder. Our contributions in this chapter are the development of error-correcting codes that achieve anytime reliability asymptotically when decoded with an expanding-window message-passing decoder over the BEC. The corresponding anytime exponents are determined through protograph-based extrinsic information transfer charts. Fundamental complications arising when transmitting with finite block lengths are identified and a combinatorial performance analysis, when transmitting over a *static* BEC with a fixed number of erasures per codeword block, is developed. Although the analysis is developed for a static BEC we show numerically that we can design efficient low-complexity finite-length codes with anytime properties even for the conventional BEC. We extend the investigations of our codes to their application when transmitting over the AWGN channel. We show that the codes are suitable in this case and that the finite-length behavior is very similar to the behavior of the codes when transmitting over the BEC. In the final part of this chapter we compare the proposed anytime codes with other anytime codes described in literature and highlight their advantages. The results have been published or are submitted for publication in:

- L. Dössel, L.K. Rasmussen, R. Thobaben, and M. Skoglund. Anytime reliability of systematic LDPC convolutional codes. In *IEEE Int. Conf. Commun.,* pages 2171-2175, Ottawa, ON, June 2012. [10 citations]

- L. Grosjean, L.K. Rasmussen, R. Thobaben, and M. Skoglund. Systematic LDPC convolutional codes: Asymptotic and finite-length anytime properties. In *IEEE Trans. Commun.*, 62(12), pages 4165-4183, December 2014. [5 citations]

- L. Grosjean, L.K. Rasmussen, R. Thobaben, and M. Skoglund. On the performance evaluation of anytime codes for control applications. In *IEEE Commun. Letters*, (Submitted)

Chapter 4

In this chapter we further develop the codes presented in Chapter 3. Our contributions involve the demonstration of the flexibility of the code structure. Through asymptotic and finite-length analysis similar to the one in Chapter 3 we determine the anytime exponent and the finite-length behavior for a variety of modifications of the base code structure. These include an increase or decrease of the density of the codes, a change of the code rate and the limitation of the code memory. We investigate the complexity of the different schemes and show how the anytime codes proposed in this thesis can be adopted to practical constraints. Parts of the results have been published in:

- L. Dössel, L.K. Rasmussen, R. Thobaben, and M. Skoglund. Anytime reliability of systematic LDPC convolutional codes. In *IEEE Int. Conf. Commun.,* pages 2171-2175, Ottawa, ON, June 2012.
- L. Grosjean, L.K. Rasmussen, R. Thobaben, and M. Skoglund. Systematic LDPC convolutional codes: Asymptotic and finite-length anytime properties. In *IEEE Trans. Comm.*, 62(12), pages 4165-4183, December 2014.
- L. Grosjean, R. Thobaben, L.K. Rasmussen, and M. Skoglund. Variable-rate anytime transmission with feedback. In *IEEE Veh. Techn. Conf.*, Montreal, Canada, September 2016, Invited paper (Submitted).

Other parts are in preparation for publication in:

- L. Grosjean, L.K. Rasmussen, R. Thobaben, and M. Skoglund. Variable rate anytime codes. In *IEEE Trans. Commun.*, (In preparation).

Chapter 5

In this chapter, as opposed to the previous chapter, we are less concerned with improving the finite-length behavior of our codes through changing the code structure but we rather devise protocols that detect and resolve the unfavourable events occurring due to the limited codeword length. As a first step we design detectors for growing error patterns both for the BEC and the AWGN channel and analyze their performance. Then we investigate the possibility of terminating the codes in regular time intervals. Based on the performance analysis developed in Chapter 3 we explore the use of feedback for achieving anytime behavior with constraints on block length. The last contribution of this chapter are two variable rate schemes that significantly improve the performance of the proposed codes over channels with poor quality. Parts of the results have been published in:

- L. Dössel, L.K. Rasmussen, R. Thobaben, and M. Skoglund. Anytime reliability of systematic LDPC convolutional codes. In *IEEE Int. Conf. Commun.,* pages 2171-2175, Ottawa, ON, June 2012.
- L. Grosjean, L.K. Rasmussen, R. Thobaben, and M. Skoglund. Systematic LDPC convolutional codes: Asymptotic and finite-length anytime properties. In *IEEE Trans. Comm.*, 62(12), pages 4165-4183, December 2014.
- L. Grosjean, R. Thobaben, L.K. Rasmussen, and M. Skoglund. Variable-rate anytime transmission with feedback. In *IEEE Veh. Techn. Conf.*, Montreal, Canada, September 2016, Invited paper (Submitted).

The major part of this chapter is in preparation for publication in:

- L. Grosjean, L.K. Rasmussen, R. Thobaben, and M. Skoglund. Variable rate anytime codes. In *IEEE Trans. Commun.*, (In preparation).

Chapter 6

While in Chapters 3-5 we focus primarily on the development of anytime codes given the constraints that we have to fulfill known from anytime theory, in the final chapter we shift focus to the actual application of the codes in an automatic control setup. That is, we turn our attention to the control aspects of stabilizing an unstable plant over a noisy communication. We restrict ourselves to the problem of optimal Linear-Quadratic-Gaussian (LQG) control when plant measurements are transmitted over an erasure channel. We focus on mean-squared stability in closed-loop, meaning that the second moment of the state has to be asymptotically finite. Our contributions include the analysis of the control performance of the entire system when using either anytime codes or block codes for error-protection. By comparing the system using block codes with the system using anytime codes we can show the advantage that anytime codes have in terms of the control cost. The contributions in this chapter are submitted or in preparation for publication in:

- L. Grosjean, L.K. Rasmussen, R. Thobaben, and M. Skoglund. On the performance evaluation of anytime codes for control applications. In *IEEE Commun. Letters*, (Submitted).
- L. Grosjean, R. Thobaben, L.K. Rasmussen, and M. Skoglund. Application of anytime codes in automatic control. In *IEEE Trans. Commun.*, (In preparation).

1.5 Notation and Acronyms

The notation and acronyms used in this thesis are summarized in the following table:

NCS	Networked control system
LQG	Linear quadratic Gaussian
LTI	Linear time invariant
BEC	Binary erasure channel
AWGN	Additive white Gaussian noise
BI-AWGN	Binary-input additive white Gaussian noise
SNR	Signal-to-noise ratio
dB	Decibel
BPSK	Binary phase-shift-keying
LDPC	Low-density parity check
LDPC-CC	Low-density parity check convolutional code
SCLDPC	Spatially coupled low-density parity check code
P-EXIT	Protograph extrinsic information transfer
LLR	Log-likelihood ratio
MAP	Maximum-a-posteriori
ML	Maximum-likelihood
UDP	User datagram protocol
TCP	Transmission control protocol
$\mathcal{X}, \mathcal{Y}$	Alphabet
$x \in \mathcal{X}$	x belongs to $\mathcal{X}$
$\mathbb{R}$	The set of real numbers
$\mathbb{N}$	The set of natural numbers, $\{1, 2, 3, \ldots\}$
$\boldsymbol{x}$	Vector
$\boldsymbol{x}^T$	Transpose of $\boldsymbol{x}$
$\boldsymbol{X}$	Matrix
$\lvert x \rvert$	Absolute value of x
$\max(x, y)$	Maximum of x and y
$\min(x, y)$	Minimum of x and y
sup	Supremum
$\oplus$	Modulo-2 addition
log	Logarithm
$\log_2$	Logarithm base 2
$N!$	Factorial of N
tanh	Hyperbolic tangent
$\text{sign}(x)$	Sign of x
$p(\cdot)$	Probability density function
$p(\cdot\lvert\cdot)$	Conditional probability density function
$\Pr\{\cdot\}$	Probability

$\mathcal{N}(m, \sigma)$	Gaussian distribution function with mean m and variance σ^2
$\mathbb{E}(X)$	Expected value of X
$H(X)$	Entropy of X
$I(X;Y)$	Mutual information between X and Y
$L(x)$	Log-likelihood ratio associated with x
ϵ,ϵ_r	Erasure probability of the BEC
ϵ_s	Erasure probability of the static BEC
ϵ_{packet}	Packet erasure probability
ρ	SNR
α	Anytime exponent
α_o	Operational anytime exponent
τ	Termination tail length
ω	Decoding window size
t	Time index
k	Message length
n	Codeword length
R	Code rate
R_t	Code rate at time t
$\boldsymbol{x}_t$	Message at time t
$\boldsymbol{x}_{[1,t]}$	Message sequence at time t
$\hat{\boldsymbol{x}}_{[1,t]}$	Estimated message sequence at time t
$\boldsymbol{y}_t$	Codeword at time t
$\tilde{\boldsymbol{y}}_{[1,t]}$	Received codeword sequence at time t
E	Number of erasures in one message (static BEC)
λ	State coefficient
ν	Input coefficient
μ	Output coefficient
w_t	Process noise at time t
v_t	Measurement noise at time t
s_t	System state at time t
r_t	Observed state at time t
u_t	Control signal at time t
$\hat{s}_t$	Estimated state at time t
$\hat{s}_{[1,t]}$	Estimated state sequence at time t
r_t^q	Quantized observed signal at time t
$\hat{r}_{[1,t]}$	Estimated observed sequence at time t
δ	Quantization bin width
$\mathcal{E}^c(\cdot)$	Encoding operation
$\mathcal{D}^c(\cdot)$	Decoding operation
$\mathcal{E}^m(\cdot)$	Bit mapping operation
$\mathcal{D}^m(\cdot)$	Bit demapping operation
$\mathcal{E}^q(\cdot)$	Quantization operation
$\mathcal{D}^q(\cdot)$	Reconstruction operation

Chapter 2

Preliminaries

In this chapter we review preliminary definitions and results required for the forthcoming chapters. After a short introduction into the basic ideas of communication theory, we focus on anytime information theory. Thereafter we summarize ideas and results from control theory and end this chapter with an introduction to LDPC codes and LDPC-Convolutional Codes.

2.1 Communication Theory

In order to follow the contributions of error-correction coding and to understand its limitations some awareness of communication and information theory is required. We therefore introduce in this section the basic concepts behind communication theory. To this end we first introduce the classical setup of a communication system, then we focus on the particular type of channels that we want to consider in this thesis. After that we introduce a useful representation of the probabilities at the decoder called log-likelihood ratios (LLRs). Finally, we introduce the concept of channel capacity and the related measures coming from information theory.

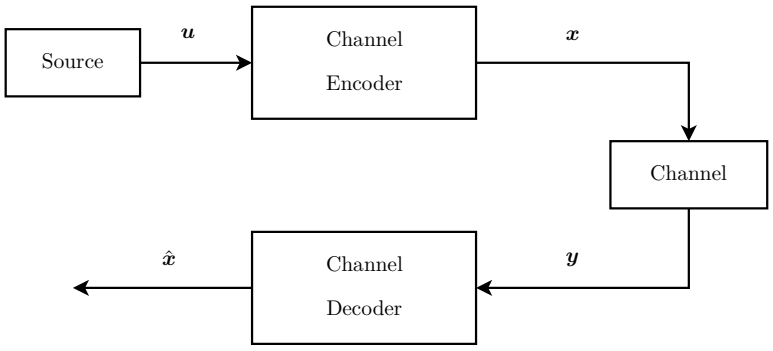


Figure 2.1: A typical communication system.

2.1.1 The Communication System

A typical communication system is depicted in Fig. 2.1. The communication system we consider in this thesis is slightly different; however, since this section deals with basic concepts of communication systems, we stick to the classical communication system and its notation. The differences will become clear in Section 2.2 when we introduce the communication system considered in anytime theory. Fig. 2.1 illustrates the four system components considered: the source, the encoder, the channel and the decoder and their relation to one another. The source emits binary data represented by a vector $\boldsymbol{u} = [u_1, \ldots, u_k]$ where $u_i \in \{0, 1\}, 1 \leq i \leq k$ and k is the length of the vector. The channel encoder adds redundancy to the information bit vector $\boldsymbol{u}$ and forms a binary codeword $\boldsymbol{v} = [v_1, \ldots, v_n]$ where $v_i \in \{0, 1\}, 1 \leq i \leq n$ and n is the length of the codeword. The ratio $R = k/n$ denotes the rate of the code. The codeword $\boldsymbol{v}$ is modulated, where in this thesis we only employ binary phase shift keying (BPSK). The modulated signal is given as $\boldsymbol{x} = 1-2\boldsymbol{v}$, where $x_i \in \{-1, +1\}$. The corresponding inverse operation (demodulation) is $\boldsymbol{v} = (1 - \boldsymbol{x})/2$. The coded and modulated signal $\boldsymbol{x}$ is transmitted over the channel. The output of the channel denoted as $\boldsymbol{y}$ takes on values in the alphabet $\mathcal{Y}$, which depends on the type of channel used. At the receiver the noisy observation $\boldsymbol{y}$ is demodulated and decoded. We now describe the type of channels that we deal with in this thesis.

2.1.2 Binary Memoryless Symmetric Channels

In this thesis we only consider transmission over binary memoryless symmetric channels. The concept of a memoryless channel is explained as follows: The channel can be described by the transition distribution $p(y_i|x_i)$ where x_i and y_i are the transmitted symbol and received symbol at time i. $p(\cdot|\cdot)$ is a probability mass function if y_i is discrete and a probability density function if y_i is continuous. The channel is memoryless if the output at any time instance only depends on the input at that time instance. It follows that for a transmitted sequence $\boldsymbol{x} = [x_1, \ldots, x_n]$ of length n and the output $\boldsymbol{y} = [y_1, \ldots, y_n]$ the following holds

$$p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{i=1}^{n} p(y_i|x_i). \tag{2.1}$$

The concept of a symmetric channel is explained as follows: Denote by $\boldsymbol{P}$ the probability transition matrix for a discrete channel, that expresses the probability of observing the output symbol y given that we send the symbol x for all possible combinations of x and y. A channel is said to be symmetric if the rows of the channel transition matrix $\boldsymbol{P}$ are permutations of each other and the columns are permutations of each other. In this thesis we consider transmission taking place over either the binary erasure channel or the binary-input additive Gaussian noise channel. Both channels are memoryless symmetric channels. Their characteristics are detailed in the following.

Binary Erasure Channel

The binary erasure channel (BEC) is the simplest non-trivial channel model for communications. Its simplicity allows us to develop analytical techniques and intuition. Although it cannot represent the characteristics of many real-world practical channels, many properties and statements encountered when studying the BEC hold in much greater generality [UR08]. Fig. 2.2 depicts the BEC. The channel input alphabet is binary $\{+1, -1\}$ and the channel output alphabet is ternary $\{+1, -1, e\}$ where e indicates an erasure. Fig. 2.2 shows the transition probabilities: a channel input is received correctly with probability $1 - \epsilon$, or it is erased with probability ϵ. The variable ϵ is therefore called the erasure probability.

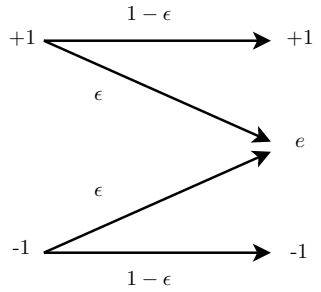


Figure 2.2: Binary erasure channel (BEC) with erasure probability ϵ.

BI-AWGN Channel

The Binary-Input Additive White Gaussian Noise channel (BI-AWGN) is at any time instance i described as

$$y_i = x_i + z_i, \tag{2.2}$$

where $x_i \in \{-1, +1\}$ is the binary input to the channel and z_i is a Gaussian random variable with zero mean and variance σ^2, having a probability density function given as

$$p(z_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-z^2/2\sigma^2}. \tag{2.3}$$

For the BI-AWGN channel the noise level is often expressed as the signal-to-noise ratio (SNR) E_b/N_0, which relates the energy E_b expended per information bit to the noise power spectral density $N_0 = 2\sigma^2$:

$$\frac{E_b}{N_0} = \frac{1}{R}\frac{1}{2\sigma^2}, \tag{2.4}$$

where R is the rate of the code.

2.1.3 Log-Likelihood Ratios

Given the received noisy observation of a bit y_i, when decoding, we are interested in the probability that x_i was sent given that y_i was received. We assume that each bit is equally likely to be transmitted. A convenient way to represent the metrics for a binary variable by a single value are log-likelihood ratios (LLRs). A log-likelihood ratio $L(x_i)$ for a bit $x_i \in \{+1, -1\}$ with probability $p(x_i)$ is defined as

$$L(x_i) = \log \frac{p(x_i = +1)}{p(x_i = -1)}. \tag{2.5}$$

In this thesis we only employ the natural logarithm to calculate LLRs. The sign of $L(x_i)$ provides a hard decision on the bit x, since

$$L(x_i) > 0 \qquad \text{if} \quad p(x_i = +1) > p(x_i = -1) \tag{2.6}$$
$$L(x_i) < 0 \qquad \text{if} \quad p(x_i = +1) < p(x_i = -1) \tag{2.7}$$
$$L(x_i) = 0 \qquad \text{if} \quad p(x_i = +1) = p(x_i = -1). \tag{2.8}$$

The magnitude $|L(x_i)|$ indicates the reliability of the hard decision. The larger the value of $|L(x_i)|$ the more reliable is the hard decision. The LLR itself is often called a soft decision for the variable x_i. The log-likelihood ratios involving the channel transition probability $p(y_i|x_i)$ and the a posteriori probability $p(x_i|y_i)$ are given as follows

$$L(y_i|x_i) = \log \frac{p(y_i|x_i = +1)}{p(y_i|x_i = -1)} \tag{2.9}$$
$$L(x_i|y_i) = \log \frac{p(x_i = +1|y_i)}{p(x_i = -1|y_i)}. \tag{2.10}$$

Using Baye's rule we can write

$$L(x_i|y_i) = L(y_i|x_i) + L(x_i). \tag{2.11}$$

To translate from LLRs back to probabilities we use

$$p(x_i = -1) = \frac{e^{-L(x_i)}}{1 + e^{-L(x_i)}} \tag{2.12}$$

and

$$p(x_i = +1) = \frac{e^{L(x_i)}}{1 + e^{L(x_i)}}. \tag{2.13}$$

Since the LLRs are a logarithmic representation of the probabilities, the multiplication of two probabilities translates into the addition of the LLRs. The decoders of the codes proposed in this thesis use LLRs in their implementation due to the lower implementation complexity. The decoding algorithms are explained in Section 2.4.3.

We now show how LLRs can be used to represent the probabilities at the decoder when using the two channel models introduced earlier.

BEC

Using LLRs to represent the metrics, for the BEC we get

$$L(x_i|y_i) = \log \frac{p(x_i = +1|y_i)}{p(x_i = -1|y_i)} = \begin{cases} -\infty & \text{if } y_i = -1 \\ +\infty & \text{if } y_i = +1 \\ 0 & \text{if } y_i = e. \end{cases} \tag{2.14}$$

BI-AWGN

Using LLRs to represent the metrics, the received LLRs for the BI-AWGN channel are

$$L(x_i|y_i) = \log \frac{p(x_i = +1|y_i)}{p(x_i = -1|y_i)} = \log \frac{p(y_i|x_i = +1)p(x_i = +1)}{p(y_i|x_i = -1)p(x_i = -1)}. \tag{2.15}$$

For an equiprobable source (e.g. $p(x_i = +1) = p(x_i = -1)$) we have

$$L(x_i|y_i) = \log \frac{p(y_i|x_i = +1)}{p(y_i|x_i = -1)} \tag{2.16}$$

and with

$$p(y_i|x_i = +1) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - 1)^2}{2\sigma^2}\right) \tag{2.17}$$

$$p(y_i|x_i = -1) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i + 1)^2}{2\sigma^2}\right) \tag{2.18}$$

we can determine the received LLRs of the BI-AWGN channel after some calculation to be

$$L(x_i|y_i) = \frac{2}{\sigma^2} y_i. \tag{2.19}$$

2.1.4 Channel Capacity

The capacity of a channel is the tight upper bound on the rate at which information can be reliably transmitted over the channel. The notion of capacity was introduced by Shannon [Sha48]. To understand the definition of channel capacity we need to first introduce two commonly used information measures, namely entropy and mutual information.

Entropy

The entropy measures the average uncertainty in a random variable and gives the number of bits on average required to describe the random variable. The entropy of

a discrete random variable $X \in \mathcal{X}$ with probability mass function $p(x) = \Pr(X = x)$ is defined as

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x). \tag{2.20}$$

The joint entropy $H(X, Y)$ of a pair of discrete random variables (X, Y) with $X \in \mathcal{X}$, $Y \in \mathcal{Y}$ and a joint distribution $p(x, y)$ is defined as

$$H(X, Y) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(x, y). \tag{2.21}$$

The conditional entropy of Y given the knowledge of X is defined as

$$H(Y|X) = -\sum_{x \in \mathcal{X}} p(x) \sum_{y \in \mathcal{Y}} p(y|x) \log p(y|x). \tag{2.22}$$

The chain rule describes the relation between the entropies defined above

$$H(X, Y) = H(X) + H(Y|X) \tag{2.23}$$
$$= H(Y) + H(X, Y). \tag{2.24}$$

Mutual Information

The mutual information describes the reduction in uncertainty of a variable due to the knowledge of another variable. The definition is as follows: For two discrete random variables X, Y with a joint probability mass function $p(x, y)$ and marginal probability mass functions $p(x)$ and $p(y)$, the mutual information is defined as

$$I(X; Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}. \tag{2.25}$$

It can be shown that

$$I(X; Y) = H(X) - H(X|Y) \tag{2.26}$$
$$= H(Y) - H(Y|X). \tag{2.27}$$

Channel Capacity

Given a discrete memoryless channel with input alphabet $\mathcal{X}$, output alphabet $\mathcal{Y}$, and transition probability $p(y|x)$, the channel capacity is defined as [Sha48]

$$C = \max_{p(x)} I(X; Y), \tag{2.28}$$

where the maximum is taken over all possible input distributions $p(x)$. The capacity of the BEC with erasure rate ϵ described above can be calculated as

$$C_{\mathrm{BEC}} = 1 - \epsilon \text{ [bits per channel use]}. \tag{2.29}$$

The capacity of the BI-AWGN channel described before can be calculated to

$$C_{\text{BI-AWGN}} = 1 - \int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y+1)^2}{2\sigma^2}} \log_2\left(1 + e^{\frac{2y}{\sigma^2}}\right) dy \text{ [bits per channel use].} \tag{2.30}$$

The maximizing input distribution $p(x)$ is the uniform distribution.

The Noisy-Channel Coding Theorem

Here we state the noisy-channel coding theorem proved in [Sha48]:

Theorem 2.1. *Associated with each discrete memoryless channel is the channel capacity C, with the following property: For any $\epsilon > 0$ and rate $R < C$, there is an error correction code of length n_0 such that there exist codes of length $n > n_0$ for which the decoded probability of error is less than ϵ.*

That means arbitrarily low probabilities of error can be obtained if a sufficiently long error correction code is used. A converse to the channel coding theorem states that if we try to transmit at a rate $R > C$ larger than capacity, the probability of error is bounded away from zero and therefore reliable transmission is not possible. The channel coding theorem tells us that codes *exist* that can be used for reliable communication, but it does not provide us with a method for constructing these codes.

2.2 Anytime Information Theory

In this section we introduce the basic concepts of anytime theory relevant to this thesis. We explain the anytime communication system model that better describes the system setup considered in this thesis than the classical communication system depicted in the previous section. Thereafter we introduce the notions of anytime capacity and anytime reliability. Anytime reliability plays a central role in the design of error-correcting codes in NCSs.

2.2.1 The Anytime Communication System

Fig. 2.3 shows the basic anytime system model typically considered in anytime information theory. A source produces an information sequence of increasing size $\boldsymbol{x}_{[1,t]} = [\boldsymbol{x}_1, \boldsymbol{x}_2 ..., \boldsymbol{x}_t]$, where $\boldsymbol{x}_i$ is a binary vector of size k and t is a time index. For every new information block $\boldsymbol{x}_t$ the encoder emits a code block $\boldsymbol{y}_t$ of size n which is a function of *all* information blocks seen so far: $\boldsymbol{y}_t = \mathcal{E}(\boldsymbol{x}_1, ..., \boldsymbol{x}_t)$. The code block $\boldsymbol{y}_t$ is transmitted over the channel. Note that all $\boldsymbol{y}_t$ are of the same size no matter how long the information sequence $\boldsymbol{x}_{[1,t]}$ is. At the receiver side a corrupted code block $\tilde{\boldsymbol{y}}_t$ is observed at each time step t and fed into the decoder. Based on *all* received code blocks $\tilde{\boldsymbol{y}}_1, ..., \tilde{\boldsymbol{y}}_t$ the decoder produces an estimate $\hat{\boldsymbol{x}}_{[1,t]} = [\hat{\boldsymbol{x}}_1(t), ..., \hat{\boldsymbol{x}}_t(t)] = \mathcal{D}(\tilde{\boldsymbol{y}}_1, ..., \tilde{\boldsymbol{y}}_t)$ where $\hat{\boldsymbol{x}}_i(t)$ denotes the estimate

of block $\boldsymbol{x}_i$ at time t. The receiver can decide to start decoding at *any time* and can deliver an estimate of *any* information block $\boldsymbol{x}_i$ transmitted so far at *any time* (therefore the terminology *anytime*). The main differences as compared to the classical communication system given in Fig. 2.1 are that the source is streaming data, and the way encoder and decoder are designed.

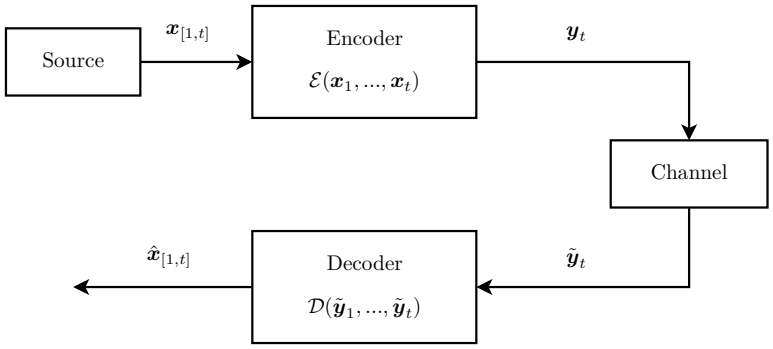


Figure 2.3: Anytime system model.

2.2.2 Anytime Capacity

Anytime capacity as opposed to Shannon capacity as introduced in Section 2.1.4 takes into account the delay d in its definition [Sah04]:

Definition 2.1. *The α-anytime capacity $C_{anytime}(\alpha)$ of a channel is given as the least upper bound of the rates at which the channel can be used to transmit data so that there exists a uniform constant β such that for all d and all times t we have*

$$P(\hat{\boldsymbol{x}}_{t-d}(t) \neq \boldsymbol{x}_{t-d}(t)) \leq \beta 2^{-\alpha d}. \tag{2.31}$$

Since the probability of error on every single bit goes to zero with increasing delay, it is clear that the anytime capacity is always less than or equal to the classical Shannon capacity. On the other hand anytime capacity is always greater than or equal to the zero-error capacity, which requires the probability of error to go to zero after a *fixed* delay.

The anytime capacity of the BEC and the input power constrained AWGN channel can be determined but for general cases the anytime capacity is still unknown.

Anytime Capacity of the BEC

For the BEC with erasure rate ϵ a parametric closed-form expression was found in [Sah01]:

$$C_{\text{anytime}}(\alpha) \geq 1 - \log_2(1+\epsilon) - \alpha. \tag{2.32}$$

Anytime Capacity of the AWGN channel

For the AWGN channel with input power constraint P the anytime capacity is determined in [Sah01] for the case when the encoder has access to noiseless feedback of the received signal delayed by one time unit. The anytime capacity is then equal to the Shannon capacity for the AWGN channel. For the case without feedback the anytime capacity is not known.

2.2.3 Anytime Reliability

The notion of anytime capacity inherently defines the concept of anytime reliability of the system in Fig. 2.3, given maximum-likelihood decoding, as

$$P(\hat{\boldsymbol{x}}_i(t) \neq \boldsymbol{x}_i | \boldsymbol{x}_{[1,t]} \text{ was transmitted}) \leq \beta 2^{-\alpha d(t,i)}, \tag{2.33}$$

where $d(t,i) = t-i$ denotes the decoding delay between information block i and the most recent block t, β is a positive constant, and $\alpha > 0$. Thus, the probability of incorrect decoding of block i decays exponentially with the decoding delay $d(t,i)$, which holds for *any* block $\boldsymbol{x}_i$ in the stream. For a channel and encoder-decoder pair of rate R the largest α such that (2.33) is fulfilled is referred to as the *anytime exponent*. In this thesis, we are interested in the *operational anytime exponent* α_o, which is the achievable exponent of a particular code ensemble.

Anytime reliability plays a vital role in the design of error-correcting codes meant to be employed in NCSs: The task of error-correcting codes in the context of NCSs is to assure that anytime reliability is guaranteed. This is the subject of Chapter 3.

2.3 Control Theory

Since we ultimately want to apply the error-correcting codes developed in this thesis in a control context, in this section we review the main ideas from control theory relevant to this thesis. First we introduce the basic system setup. Then we specify the Linear-Quadratic-Gaussian (LQG) control problem which is the particular problem we want to consider. Furthermore we summarize some of the results known in control theory in the context of NCSs.

2.3.1 Control System Setup

Fig. 2.4 depicts the basic control setup that we want to consider. It consists of a plant that is evolving over time. In each time step the observer observes the state of the plant. The observed signal is transmitted over the channel to the state estimator. Based on the output of the state estimator the controller finds a suitable control signal which is applied to the plant in the next time step.

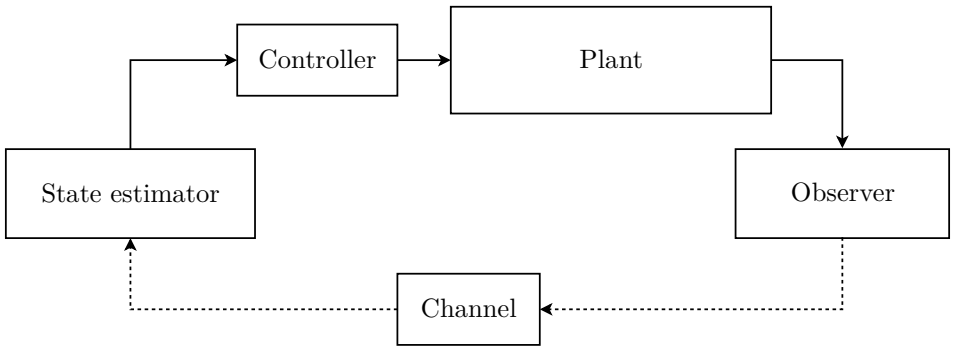


Figure 2.4: Basic control system.

2.3.2 LQG Control

The Linear-Quadratic-Gaussian (LQG) problem is concerned with controlling a plant, modeled as a discrete-time, scalar linear time invariant (LTI) system where both process and measurement noise are modelled as additive white Gaussian noise. The state update equation and observer state are given by:

$$s_{t+1} = \lambda s_t + \nu u_t + w_t \tag{2.34}$$

$$r_t = \mu s_t + v_t, \tag{2.35}$$

where s_t is the state of the system, r_t is the observed signal and u_t is the control signal. The additive terms w_t and v_t are independent discrete-time white Gaussian noise processes representing process and measurement noise. Both are zero-mean with variance σ_w^2 and σ_v^2, respectively. The variables $\lambda, \nu \in \mathbb{R}$ and $\mu \in \mathbb{R}, \mu > 0$ are known nonzero values. The initial condition, s_0, is zero mean, and is uncorrelated with the processes w_t and v_t. In this thesis s_0 is assumed to be known. The plant is unstable if for the variable λ we have $|\lambda| > 1$. Without loss of generality we let $\mu = \nu = 1$ for the sake of a more compact notation. In this thesis we only consider scalar variables but the problem can be extended to state vectors and observation vectors as well. The objective of the LQG problem is to minimize the output variance $\mathbb{E}[r_t^2]$ by means of a suitable control signal.

Performance Index

The performance index is given as

$$J = \lim_{t\to\infty} \sup \mathbb{E}[r_t^2]. \tag{2.36}$$

The problem of finding the control signal for a plant given in (2.35) that minimizes the quadratic cost function J subject to the constraint that u_t only depends

(strictly) causally on the output signal $\boldsymbol{r}_{[1,t-1]}$ and possibly on its previous values $\boldsymbol{u}_{[1,t-1]}$ is called the LQG control problem. We make use of the performance index given in (2.36) when evaluating the control performance using different error-correcting codes in Chapter 6.

LQG Control Over Noiseless Channels

In the classical LQG problem the channel between the observer and the controller is noise free. Solutions to this problem can be found in standard textbooks on optimal control theory; for example, see [Ste94]. Some key results are summarized as follows:

- The optimal controller is linear in the current state estimate; that is $u_t = L\hat{s}_t$, where $\hat{s}_t$ is the optimal estimate of the current state, and L can be determined based on the coefficients and the covariance matrices.
- The control gain L is independent of the statistics of the problem.
- The separation principle between estimation and control holds, meaning that in order to obtain the optimal solution we can independently solve the estimation problem (assuming no control) and the control problem (assuming perfect information).
- The time-varying Kalman filter [Kal60] can be used to estimate the current state.

For noiseless channels, the smallest rate necessary for communication over the communication link between the observer and controller above, which an unstable discrete linear system can be stabilized with, is specified by the data rate theorem: In the scalar case the rate R has to satisfy

$$R > \log_2 |\lambda| \text{ [bits per sample]}. \tag{2.37}$$

This implies that for an unstable plant with large λ a larger rate is required for stabilization than for a plant with smaller λ. The data rate theorem was proved for bounded disturbances and bounded initial support in [TM04]. The same theorem holds for unbounded disturbances and unbounded initial support but bounded higher moments if second moment stability is required [NE04].

2.3.3 LQG Control Over Noisy Channels

Having introduced the main results for LQG control over noiseless channels, we now turn our attention to the case relevant in this thesis, where the channel between the observer and the state estimator is not error-free. The general solution to the LQG problem when transmission takes place over a noisy channel is unknown. Solutions have however been derived for specific channel models. In this thesis we relate to

results where packets of data are transmitted over the channel and the channel is modeled by a Bernoulli process. Packets of data are then lost independently with a certain probability. For TCP-like transmission protocols, where data packets are acknowledged by the receiver it was shown in [SSF+05c, SSF+05a, SSF+05b] that the separation principle holds and that the optimal controller is a linear function of the state. For the case considered in this thesis where no packet acknowledgements are available it is shown in [SSF+06] that:

- The separation principle does not hold in general. That is, the controller and the estimator cannot be designed independently.
- The optimal controller is in general non-linear.

A sub-optimal solution leading to useful results is however derived in [SSF+06] by assuming that the separation principle does hold and by designing a constant gain Kalman filter. In Chapter 6 we make use of this approach.

2.4 LDPC Codes

The error-correcting codes proposed in this thesis are a type of low-density parity-check convolutional codes (LDPC-CCs). In order to understand their structure and properties we first step through the main ideas related to low-density parity-check codes (LDPC) and then deepen the description of LDPC-CCs. We now review the structure of LDPC codes, the construction mechanism using protographs and their decoding. After that we outline protograph-extrinsic information transfer (P-EXIT) analysis which is a useful tool to analyze the asymptotic performance of protograph based LDPC codes.

2.4.1 Definition

LDPC codes were invented by Gallager [Gal63] in the 1960's. They are a class of binary linear block codes that are capable of performing extremely close to the Shannon capacity. Their potential remained undiscovered until 35 years later, when McKay and Neal introduced a "new" class of block codes in the 1990s that were soon recognized to be a rediscovery of the LDPC codes developed by Gallager. An LDPC code is defined by an $m \times n$ binary parity-check matrix $\boldsymbol{H}$. The *code* is the set of all binary vectors $\boldsymbol{y}$, called codewords, satisfying

$$\boldsymbol{H}\boldsymbol{y}^T = \boldsymbol{0}. \tag{2.38}$$

As the name suggests, the parity-check matrix of an LDPC code only contains a small portion of non-zero entries. As shown above the parity-check constraints on the codewords are defined in a matrix form: That means, that each row of $\boldsymbol{H}$ corresponds to a parity-check equation and each column of $\boldsymbol{H}$ corresponds to a bit in the codeword. The entry (i, j) in $\boldsymbol{H}$ is equal to 1 if the j-th codeword bit is

included in the i-th parity check equation; otherwise, the entry is equal to 0. The *Tanner graph* [TSS+04] is a graphical representation for a parity-check matrix $\boldsymbol{H}$. It is a bipartite graph consisting of a set of n variable nodes denoted as $V_1, \ldots, V_n$ and a set of m check nodes denoted as $C_1 \ldots C_m$. An entry (i, j) in $\boldsymbol{H}$ translates into an edge between check node C_i and variable node V_j in the Tanner graph. Fig. 2.5 shows the Tanner graph of the example of a parity-check matrix given below.

$$\boldsymbol{H}\boldsymbol{y}^T = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad \begin{array}{c} y_2 \oplus y_4 = 0 \\ y_1 \oplus y_3 = 0 \\ y_1 \oplus y_4 \oplus y_5 = 0 \end{array} \tag{2.39}$$

A LDPC code is said to be (d_v, d_c)-*regular* if each row in $\boldsymbol{H}$ contains the same

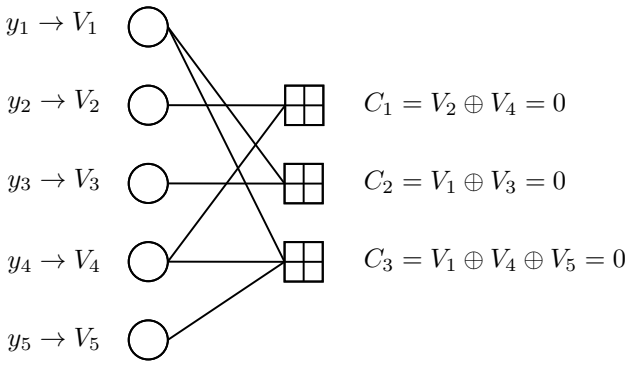


Figure 2.5: Tanner graph of the parity-check matrix given in (2.39). Circles represent variable nodes, and squares indicate check nodes.

number d_c of 1s and each column contains the same number d_v of 1s. Equivalently, in the Tanner graph each variable node has degree d_v and each check node has degree d_c. For *irregular* LDPC codes, the variable and check node degrees are allowed to vary. Irregular LDPC codes can approach the channel capacity if designed properly [RU01, RSU01, CFRU01]. However, since irregular LDPC codes are not used in this thesis, details are omitted.

Cycles

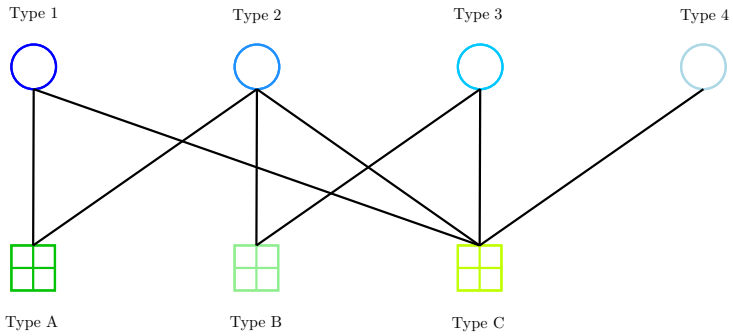
A cycle in a Tanner graph is a sequence of connected vertices which starts and ends at the same vertex in the graph but which contains other vertices no more than once. The length of the cycles is the number of edges it contains. The existence of cycles in the Tanner graph of a code reduces the effectiveness of the iterative decoding process. This will become more clear in Section 2.4.3.

2.4.2 Protograph Ensembles

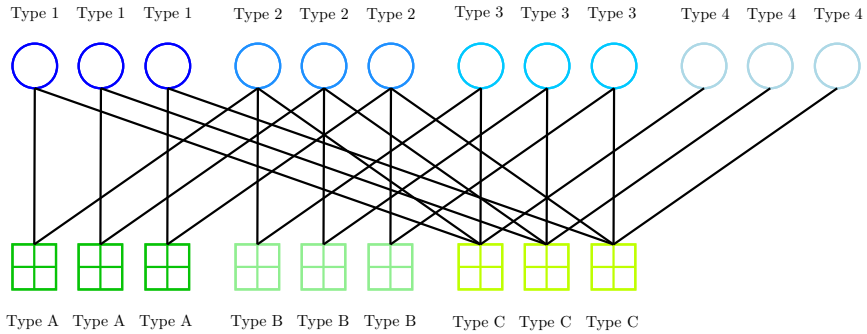
When designing an LDPC code instead of analyzing a particular LDPC code, we often think in terms of ensembles of codes. An ensemble of (d_v, d_c)-codes for instance is the family of codes with a parity-check matrix having a check node degree equal to d_c and a variable node degree equal to d_v. Concentration results [RU01] tell us then that a code randomly chosen from the ensemble has properties that match the properties of the ensemble average with high probability if the codeword length is sufficiently large. A protograph-based LDPC code ensemble is a class of LDPC codes constructed from a template called a protograph [Tho03]. Based on the protograph, LDPC codes of arbitrary size can be constructed and their performance can be predicted by analyzing the protograph only. A protograph $G = (V, C, E)$ is a Tanner graph typically of small size. It consists of a set of N_B variable nodes V, a set of M_B check nodes C and a set of edges E. Fig. 2.6a shows a protograph with 4 variable nodes and 3 check nodes. A larger graph $G' = (V', C', E')$, the so-called *derived* graph, can be constructed from the protograph by a copy-and-permute operation, also called *lifting*. The operation is illustrated in Fig. 2.6. The protograph of Fig. 2.6a has been copied three times to obtain the three disconnected subgraphs shown in Fig. 2.6b. After the copy-operation the permute-operation consists of permuting the endpoints of the three copies of each edge in the protograph among the three copies of the corresponding variable and check nodes as shown in Fig. 2.6c. The corresponding matrix of size $M_B \times N_B$ describing the protograph is called the *base matrix* $\boldsymbol{B}$. The base matrix entry $B_{m,n}$ indicates the number of edges between check node C_m and variable node V_n in the protograph. The derived matrix is obtained by replacing each entry $B_{m,n}$ in the base matrix with the sum of $B_{m,n}$ distinct (non-overlapping) randomly selected permutation matrices of size M (the lifting factor). The matrix corresponding to the derived graph is the parity-check matrix of the protograph code. The protograph base matrix, intermediate matrix and lifted matrix of the graphs shown in Figs. 2.6a, 2.6b, 2.6c are given as follows:

$$\boldsymbol{B} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{2.40}$$

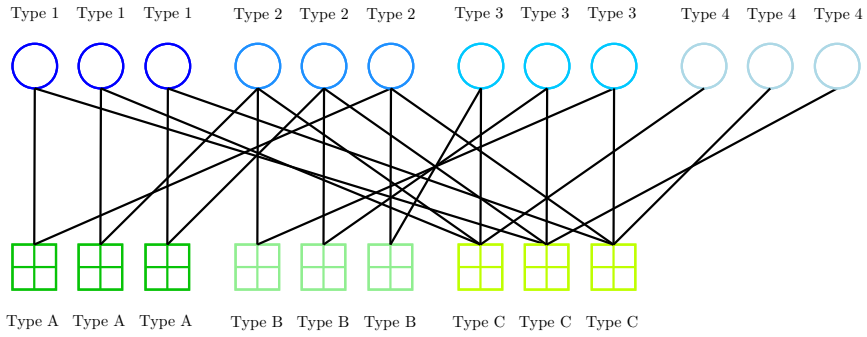
$$\boldsymbol{H}_{copy} = \begin{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} & \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} \\ \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{bmatrix} \tag{2.41}$$



(a) A simple protograph corresponding to (2.40).



(b) A protograph copied three times resulting in the Tanner graph of $\boldsymbol{H}_{copy}$ given in (2.41).



(c) A derived graph corresponding to $\boldsymbol{H}_{copy\&permute}$ given in (2.42).

Figure 2.6: Lifting process of obtaining the derived graph from the protograph.

$$\boldsymbol{H}_{copy\&permute} = \begin{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} & \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} & \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} \\ \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} & \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \end{bmatrix} \tag{2.42}$$

Multiple parallel edges are allowed in the protograph however not in the derived graph. Note that during the copy-and-permute operation the degrees of neither check nor variable nodes change. Note as well that the type of the variable node that a check node of certain type is connected to is not changed by the copy-and-permute operation either.

The local neighborhood to depth d of node (i, j) consists of all check and variable nodes connected to node (i, j) by a path of length d or less. The advantage of designing LDPC codes with the help of protographs lies in the fact that a protograph code unlike any other irregular code has a local neighborhood of a node (i, j) in the derived graph G' that is deterministic. Moreover it is completely determined by the protograph G. As an example the neighborhood of a variable node of type 1 in the derived graph given in Fig. 2.6c to depth $d = 3$ is shown in Fig. 2.7, which is completely determined by the neighborhood of a variable node of type 1 in

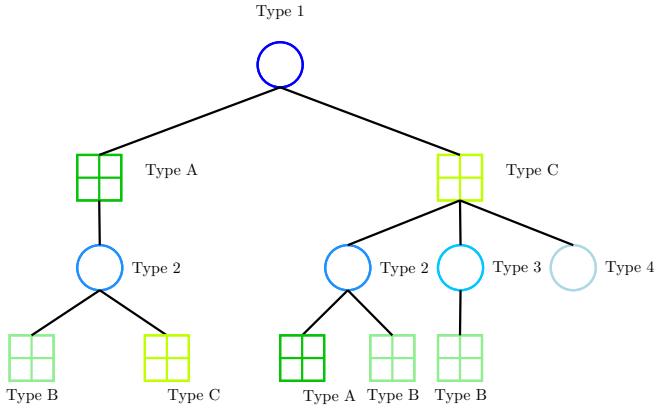


Figure 2.7: The neighborhood of a variable node of type 1 for the derived graph given in Fig. 2.6c.

the protograph given in Fig. 2.6a. Due to the deterministic neighborhood we can restrict ourselves to analyzing the protograph instead of the derived graph. This approach is taken when performing P-EXIT analysis as described in Section 2.4.5.

2.4.3 Decoding

There exist different types of decoders. The maximum-likelihood (ML) decoder returns the decoded codeword $\hat{\boldsymbol{y}}$ according to the rule

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y} \in Y}{\operatorname{argmax}}\, p(\tilde{\boldsymbol{y}}|\boldsymbol{y}). \tag{2.43}$$

The maximum-a-posteriori (MAP) decoder on the other hand chooses the codeword $\hat{\boldsymbol{y}}$ that maximizes $p(\boldsymbol{y}|\tilde{\boldsymbol{y}})$ (the a posteriori probability), that is $\hat{\boldsymbol{y}}$ is chosen according to the rule

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y} \in Y}{\operatorname{argmax}}\, p(\boldsymbol{y}|\tilde{\boldsymbol{y}}). \tag{2.44}$$

If each codeword is equally likely, then MAP and ML decoding return the same result. If the decoder is provided with a priori information about $\boldsymbol{y}$ then only the MAP decoder takes into account this extra information. MAP decoding can as well be performed on a symbol by symbol basis. The symbol $\hat{y}_i$ is then chosen according to the rule

$$\hat{y}_i = \underset{y_i \in \{+1,-1\}}{\operatorname{argmax}}\, p(y_i, \tilde{\boldsymbol{y}}) \tag{2.45}$$

The disadvantage of both the ML and MAP decoder is that for a code with message length k, they need to calculate the probabilities mentioned above for all the 2^k codewords in the code. This can quickly become very complex. LDPC codes are therefore decoded in a different way. Instead of calculating the probabilities above their decoding algorithms work on *estimates* of the probabilities $p(y_i|\tilde{y}_i)$ that are obtained in an iterative manner through low-complexity processes. This kind of decoder is called an iterative message-passing decoder. While a message-passing decoder does not deliver the optimal ML or MAP estimate, it can come very close to it. The algorithm is explained in the following. We will first outline the basic idea of message-passing decoding, and then describe the algorithm in more detail for decoding over the BEC and the AWGN channel.

A message-passing algorithm consists of an exchange of messages along the edges of a Tanner graph. Each message contains either an estimate of the sender or the recipient coded bit. The messages are passed back and forward between bit and check nodes in an iterative way. Every node has only access to the messages on the edges connected to it. The message-passing process continues until a result is achieved or the maximum number of iterations is attained. Message-passing decoding on the BEC is often referred to as iterative erasure decoding or peeling decoding [LMSS01]. For other channels than the BEC the decoding is called belief propagation decoding.

Peeling Decoding

The peeling decoder [Gal63] is a special case of message-passing algorithm when transmission takes place over the BEC. The messages passed along the Tanner graph

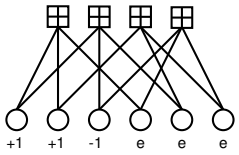
edges are binary messages and contain hard-decisions, meaning that a message is either the erasure symbol or the exact bit (+1 or -1). The algorithm is described below. In each iteration round the check to variable and variable to check messages are updated.

- **Initialization:** Each variable node contains the value of the corresponding received bit.
- **Check to variable node update**: For each connected variable node separately, the check node determines the value of the variable node such that the parity-check equation is satisfied and sends a message to the variable node containing this value. If more than one of the connected variable nodes is erased, the check node sends an erasure.
- **Variable to check node update**: If an unknown variable node receives an incoming message which is not an erasure, it takes the value. The message from a variable node to a check node contains the value that the variable node holds.
- **Stopping Criterion**: If at some point all check equations are satisfied, (i.e. $\hat{\boldsymbol{y}}\boldsymbol{H}^T = \mathbf{0}$), a valid codeword is found and the decoder terminates. Otherwise the decoder continues iterating until a valid codeword is found or a maximum number of iterations is reached.

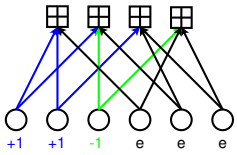
Fig. 2.8 illustrates the progress of the peeling decoder. Given the Tanner graph in Fig. 2.8a and the received codeword $\tilde{\boldsymbol{y}} = [+1, +1, -1, e, e, e]$ the algorithm first initializes each variable node with the corresponding received bit and then updates check and variable nodes according to the previously described procedure until a valid codeword is found.

Belief Propagation

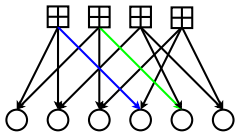
The belief propagation algorithm is a soft decision message-passing algorithm. Instead of messages containing hard decisions, the messages now involve probabilities. When using probabilities represented with the help of log-likelihood ratios, the algorithm is called the sum-product algorithm. The sum-product algorithm aims at computing the MAP probability P_i for each codeword bit y_i. We now describe the decoding algorithm for the BI-AWGN channel. There are three different types of probabilities involved in the algorithm: The *a priori* probabilities, which are the received bit probabilities known before the algorithm embarks. The *extrinsic* information, and the *a posteriori* probability for each bit i, are both explained in detail now. The extrinsic information passed from check node j to variable node i, denoted as $P^{ext}(j, i)$ is the probability that $y_i = -1$ based on the information available from the other variable nodes connected to check node j. In other words, it is the probability that the parity-check equation corresponding to check node j



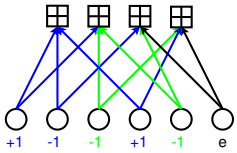
(a) Initialization



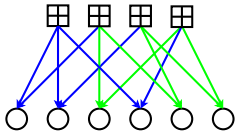
(b) Variable to check node update.



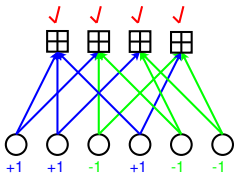
(c) Check to variable node update.



(d) Variable to check node update.



(e) Check to variable node update.



(f) All check equations fulfilled. Algorithm stopped.

Figure 2.8: Peeling decoder message-passing for the received codeword $\tilde{\boldsymbol{y}} = [1, 1, -1, e, e, e]$. Messages either contain a '-1' (green), a '+1' (blue), or an 'e' (black).

is satisfied if $y_i = -1$. This is the probability that an odd number of the bits in that parity-check equation are -1s:

$$P^{ext}(j,i) = \frac{1}{2} - \frac{1}{2} \prod_{i' \in B_j, i' \neq i} (1 - 2P(j,i')) \tag{2.46}$$

where $P(j,i')$ is the current estimate of check node j of the probability that $y_{i'} = -1$ and B_j is the set of variable nodes connected to check node j. The extrinsic probability that $y_i = +1$ is then equal to $1 - P^{ext}(j,i)$. We omit the iteration index in the derivations. In general the messages determined at one node are based on the information received from the connected nodes in the previous iteration step. The extrinsic probability can be expressed using LLRs as introduced in Section 2.1.3:

$$L(P^{ext}(j,i)) = \log \frac{1 - P^{ext}(j,i)}{P^{ext}(j,i)}. \tag{2.47}$$

For the BI-AWGN channel the received LLRs where derived in Section 2.1.3 as $L_{ch}(i) = 2/\sigma^2 \tilde{y}_i$. With the above we can now determine the LLR of the i-th bit as

the sum of the extrinsic LLR and the received LLR:

$$L(P_i) = L_{ch}(i) + \sum_{j \in A_i} L(P^{ext}(j,i)), \tag{2.48}$$

here A_i denotes the set of check nodes connected to variable node i. The LLRs $L(P^{ext}(j,i))$ are determined by substituting (2.46) into (2.47). After some calculations we arrive at

$$L(P^{ext}(j,i)) = \log \frac{1 + \prod_{i' \in B_j, i' \neq i} \frac{1-e^{M(j,i')}}{1+e^{M(j,i')}}}{1 - \prod_{i' \in B_j, i' \neq i} \frac{1-e^{M(j,i')}}{1+e^{M(j,i')}}}, \tag{2.49}$$

where

$$M(j,i') = L(P(j,i')) = \log \frac{1 - P(j,i')}{P(j,i')}. \tag{2.50}$$

$M(j,i)$ is the message sent from variable node i to check node j. It is given as

$$M(j,i) = L(P(j,i))) = L_{ch}(i) + \sum_{j' \in A_i, j' \neq j} L(P^{ext}(j',i)), \tag{2.51}$$

which is the sum in (2.48) without the component $L(P(j,i))$ just received from the j-th check node. The algorithm is now given as follows:

- **Initialization:** Each variable node contains the value L_{ch} of the corresponding received bit. The message sent from variable node i to a connected check node j is therefore $L(P(i)) = L_{ch}(i)$.
- **Check to variable node update**: Determine for all check nodes j all $L(P^{ext}(j,i))$ as in (2.49) and send the messages to the corresponding variable nodes.
- **Variable to check node update**: Determine for all variable nodes i all $L(P(j,i))$ as in (2.51) and send the message to the corresponding check node. Determine all $L(P_i)$ as in (2.48) and do a hard decision on the LLRs $\hat{y}_i = \text{sign}(L(P_i))$.
- **Stopping Criterion**: If at some point, all check equations are satisfied, i.e. $\hat{\boldsymbol{y}}\boldsymbol{H}^T = \boldsymbol{0}$, a valid codeword is found and the decoder terminates. Otherwise the decoder continues iterating until a valid codeword is found or a maximum number of iterations is reached.

The sum-product algorithm computes an approximation of the MAP value for each variable node. Only when the Tanner graph is cycle free the resulting MAP value is exact. The approximation is however very good even when used on a Tanner graph containing cycles. The effect of cycles on the decoding process is explained in the next paragraph.

2.4.4 Stopping Sets and Trapping Sets

Small cycles have a detrimental effect on the performance of the message-passing decoder. When transmitting over the BEC, cycles in the Tanner graph provoke the existence of so called stopping sets. A stopping set $\mathcal{S}$ in the codes Tanner graph is a subset of the variable node set such that every check node connected to $\mathcal{S}$ connects into $\mathcal{S}$ at least twice [DPT+02]. When decoding over the BEC, given that all bits in $\mathcal{S}$ are erased, the peeling decoder cannot recover any of these bits. The set of stopping sets therefore determines the erasure patterns that cause the decoder to fail.

For sum-product decoding, cycles lead to correlations in the marginal probabilities computed in the process of the algorithm. The smaller the cycles are, the fewer are the number of iterations during which the marginal probabilities are correlation free. The equivalent to stopping sets, when transmitting over the BI-AWGN channel are trapping sets. Trapping sets [Ric03] are an approximate characterization of decoding failures on the BI-AWGN channel. There has been done substantial research on stopping/trapping sets, for instance on how to determine their size, how to remove them, etc. We will see however later in Section 3.2.2 that in this work we are interested in a slightly different type of error/erasure patterns that cause the decoder to fail.

2.4.5 P-EXIT Analysis

Protograph-based extrinsic information transfer analysis (P-EXIT) is an analysis that tracks the mutual information between the messages on the edges of the protograph and the underlying bits [LC07]. It is a useful tool for designing protograph based LDPC codes since it allows us to determine the performance of the codes in the asymptotic limit of an infinite lifting factor and we will make frequent use of it throughout the thesis. Denote by $I_{Av}^{\ell}(i,j)$ the mutual information sent between check node i and variable node j at iteration ℓ and similarly denote by $I_{Ev}^{\ell}(i,j)$ the mutual information sent between variable node j and check node i at iteration ℓ. The a posteriori mutual information of variable node j is denoted as $I_{APP}(j)$. The P-EXIT update equations as proposed in [LC07] for the BEC are then given as

- **Initialization:** $I_{Ev}^{0}(i,j) = 1 - (1 - I_{ch}) = 1 - \epsilon$ (2.52)
- **Check node i to variable node j update**:

$$I_{Av}^{\ell+1}(i,j) = \prod_{s=1, s\neq j}^{d_c(j)} I_{Ev}^{\ell}(i,s) \tag{2.53}$$

- **Variable node j to check node i update**:

$$I_{Ev}^{\ell+1}(i,j) = 1 - \epsilon \prod_{s=1, s\neq i}^{d_v(i)} (1 - I_{Av}^{\ell+1}(s,j)) \tag{2.54}$$

• **APP mutual information evaluation**:

$$I_{\mathrm{APP}}(j) = 1 - \epsilon \prod_{s=1}^{d_v(i)} (1 - I_{Av}^{\infty}(s, j)). \tag{2.55}$$

For the BEC with erasure rate ϵ, $I_{ch} = 1 - \epsilon$ and the decoding erasure probability can then easily be determined as $P_{APP}(j) = 1 - I_{APP}(j)$. Please refer to [LC07] for the implementation of the algorithm when transmitting over the AWGN channel.

The P-EXIT equations are closely related to the density evolution equations. As opposed to P-EXIT analysis, density evolution for irregular LDPC codes tracks the average message distributions over all codes in an ensemble, which are equal along *all* edges in the graph. Since a protograph however imposes a deterministic structure on the neighborhood of each node the message distributions are different between the edges in the protograph. If the density evolution equations are tracked separately for each edge of the protograph, then a mulit-edge density evolution approach results in the same equations as the ones given in P-EXIT analysis. This is done in [LTF09].

2.5 LDPC-Convolutional Codes

Here we briefly define LDPC convolutional codes (LDPC-CCs). A more detailed description can be found in [Pus08, JFZ99].

2.5.1 Structure and Basic Definitions

A rate $R = k/n$ binary LDPC-CC code with so-called syndrome former memory m_s is defined by an infinite-sized parity-check matrix

$$\boldsymbol{H}_{[\infty]} = \begin{bmatrix} \boldsymbol{H}_0(1) & & & & & \\ \boldsymbol{H}_1(1) & \boldsymbol{H}_0(2) & & & & \\ \vdots & \vdots & \ddots & & & \\ \boldsymbol{H}_{m_s}(1) & \vdots & \ddots & \boldsymbol{H}_0(t) & & \\ & \boldsymbol{H}_{m_s}(2) & \ddots & \boldsymbol{H}_1(t) & \ddots & \\ & & \ddots & \vdots & \ddots & \\ & & & \boldsymbol{H}_{m_s}(t) & \ddots & \\ & & & & \ddots & \end{bmatrix}, \tag{2.56}$$

where $t, m_s \in \mathbb{N}$ and the elements $\boldsymbol{H}_i(t), i = 0, 1, ...m_s$ are binary matrices of size $(n-k) \times n$ that satisfy the properties: $\boldsymbol{H}_i(t) = 0$, $i < 0$ and $i > m_s, \forall t.\ \exists t > 0$

such that $\boldsymbol{H}_{m_s}(t) \neq 0$. $\boldsymbol{H}_0(t) \neq 0$, $\forall t$, has full rank. The submatrices are given by

$$\boldsymbol{H}_i(t) = \begin{bmatrix} h_i^{(1,1)}(t) & \dots & h_i^{(1,n)}(t) \\ \vdots & & \vdots \\ h_i^{(n-k,1)}(t) & \dots & h_i^{(n-k,n)}(t) \end{bmatrix}. \tag{2.57}$$

Given an information sequence at time t

$$\boldsymbol{x}_{[0,t-1]} = [\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_{t-1}], \tag{2.58}$$

where $\boldsymbol{x}_i = (x_i^1, x_i^2, \dots, x_i^k)$ is a binary vector of length k, and $0 \leq i < t$, the encoder maps the information sequence into the code sequence

$$\boldsymbol{y}_{[0,t-1]} = [\boldsymbol{y}_0, \boldsymbol{y}_1, \dots, \boldsymbol{y}_{t-1}]. \tag{2.59}$$

Here $\boldsymbol{y}_i = (y_i^1, y_i^2, \dots, y_i^n)$ is a binary vector of length n. In this thesis we only consider systematic encoders and we therefore write

$$\boldsymbol{y}_i = [\boldsymbol{y}_i^s, \boldsymbol{y}_i^c], \tag{2.60}$$

where $\boldsymbol{y}_i^s = \boldsymbol{x}_i$ is the systematic part and $\boldsymbol{y}_i^c$ is the parity-check vector of length $n-k$. The set of semi-infinite binary row vectors $\boldsymbol{y}_{[\infty]}$ that satisfy

$$\boldsymbol{H}_{[\infty]} \boldsymbol{y}_{[\infty]}^T = \boldsymbol{0}_{[\infty]}^T \tag{2.61}$$

are the codewords of the LDPC-CC. A LDPC-CC is (J, K)-*regular* if the number of ones in every column of $\boldsymbol{H}_{[0,\infty]}$ is equal to J and the number of ones in every row is equal to K. As their block-code counterparts, LDPC-CCs can be constructed based on a protograph. The protograph of an LDPC-CC with memory m_s has then the following structure

$$\boldsymbol{B}_{[\infty]} = \begin{bmatrix} \boldsymbol{B}_0 & & \\ \boldsymbol{B}_1 & \boldsymbol{B}_0 & \\ \vdots & \boldsymbol{B}_1 & \ddots \\ \boldsymbol{B}_{m_s} & \vdots & \ddots \\ & \boldsymbol{B}_{m_s} & \ddots \\ & & \ddots \end{bmatrix}. \tag{2.62}$$

The size of the matrices $\boldsymbol{B}_i$ depends on the code we want to construct.

2.5.2 Encoding

Due to the lower-left band diagonal structure it is possible to causally encode the information sequence in real-time, making LDPC-CCs suitable for streaming applications [JFZ99]. There are two different realizations presented in [Pus08] for

encoding LDPC-CCs of rate $R = k/n$ that are used in this thesis. These are a syndrome former and a partial syndrome former realization. We only consider systematic codes therefore we choose the last $n-k$ columns of $\boldsymbol{H}_0(t)$ to be the identity matrix:

$$\boldsymbol{H}_0(t) = [\boldsymbol{H}_0^0(t)\boldsymbol{I}_{(n-k)}]. \tag{2.63}$$

Here $\boldsymbol{I}_{(n-k)}$ denotes the identity matrix of size $(n-k) \times (n-k)$.

Syndrome Former Realization

The syndrome former realization is found by rewriting equation (2.61):

$$\boldsymbol{y}_t \boldsymbol{H}_0^T(t) + \boldsymbol{y}_{t-1}\boldsymbol{H}_1^T(t) + \ldots + \boldsymbol{y}_{t-m_s}\boldsymbol{H}_{m_s}^T(t) = \boldsymbol{0}. \tag{2.64}$$

From the definition of LDPC-CCs we know that all submatrices $\boldsymbol{H}_0(t)$ have full rank. Since furthermore the last $n-k$ columns of $\boldsymbol{H}_0(t)$ are equal to the identity matrix we can calculate the code block $\boldsymbol{y}_t$ based on the previous code blocks $\boldsymbol{y}_{t-1}, \boldsymbol{y}_{t-2}, \ldots, \boldsymbol{y}_{t-m_s}$ and the current information block $\boldsymbol{x}_t = \boldsymbol{y}_t^s$ as follows

$$y_t^j \begin{cases} = x_t^j, & j = 1, \ldots, k \\ = \sum_\ell^k y_t^\ell h_0^{j-k,\ell}(t) + \sum_{i=1}^{m_s}\sum_{\ell=1}^{n} y_{t-i}^\ell h_i^{j-k,\ell}, & j = k+1, \ldots, n \end{cases}. \tag{2.65}$$

The syndrome former realization can be implemented using shift-registers [JFZ99]. It requires $m_s n + k$ memory units. The encoding complexity per bit is proportional to the number K of ones per row in $\boldsymbol{H}_{[0,\infty]}$ but it is independent of the codeword length and the syndrome former memory m_s.

The syndrome former realization is used to encode the unterminated anytime codes proposed in this thesis.

Partial Syndrome Former Realization

The partial syndrome former realization is an alternative encoding realization that is useful when encoding terminated LDPC-CCs. It is found by rewriting equation (2.61):

$$\boldsymbol{y}_{[0,t-1]}\boldsymbol{H}_{[0,t-1]}^T = [\boldsymbol{0}_{[0,t-1]}|\boldsymbol{p}_t], \tag{2.66}$$

where $\boldsymbol{0}_{[0,t-1]}$ is a zero vector of length $(n-k)t$ and

$$\boldsymbol{p}_t = [\boldsymbol{p}_{t,1}, \boldsymbol{p}_{t,2}, \ldots, \boldsymbol{p}_{t,m_s}]. \tag{2.67}$$

The vector $\boldsymbol{p}_{t,i} = [p_{t,i}^1, p_{t,i}^2, \ldots, p_{t,i}^{(n-k)}], i = 1, 2, \ldots, m_s$ is called the *partial syndrome* and describes the state of the partial syndrome former encoder at time t. The values of $\boldsymbol{p}_t$ can be calculated recursively as follows

$$\begin{cases} \boldsymbol{p}_{t,i} = \boldsymbol{p}_{t-1,i+1} + \boldsymbol{y}_{t-1}\boldsymbol{H}_i^T(t+i-1) & \text{for } i = 1, 2, \ldots, m_s - 1 \\ \boldsymbol{p}_{t,i} = \boldsymbol{y}_{t-1}\boldsymbol{H}_{m_s}^T(t+m_s-1) & \text{for } i = m_s. \end{cases} \tag{2.68}$$

From (2.66) we can see that

$$[\boldsymbol{y}_t^s, \boldsymbol{y}_t^c]\boldsymbol{H}_0^T(t) = \boldsymbol{p}_{t,1}. \tag{2.69}$$

Since the last $n-k$ columns of $\boldsymbol{H}_0(t)$ are equal to the identity matrix we can solve the above for $\boldsymbol{y}_t^c$ using

$$\boldsymbol{y}_t^c = \boldsymbol{y}_t^s[\boldsymbol{H}_0^0(t)]^T + \boldsymbol{p}_{t,1}. \tag{2.70}$$

The partial syndrome former realization can be implemented using a shift-register [JFZ99]. It requires $(n-k)m_s$ memory units. The encoding complexity per bit is proportional to the number K of ones per row in $\boldsymbol{H}_{[0,\infty]}$ but it is independent of the codeword length and the syndrome former memory m_s.

2.5.3 Decoding

As for their block-code counterparts, LDPC-CCs can be decoded with iterative message-passing algorithms. Moreover sliding-window decoding is possible for LDPC-CCs in their non-terminated form [PIS+10].

Sliding Window Decoding

The sliding window decoder exploits the characteristic that variable nodes that are more than a fixed number of columns apart cannot be involved in the same check equation. Therefore it is possible to decode continuously by *sliding* a window along the received bit sequence instead of waiting until the entire bit sequence has arrived at the receiver. This is particularly important for unterminated LDPC-CCs.

The windowed decoder is explained in [PIS+10] for a (J, K)-regular protograph-based LDPC-CC constructed by a protograph as given in (2.62), where the matrices $\boldsymbol{B}_i$ are of size $J' \times K'$. K' and J' are positive integers such that $J = aJ', K = aK'$ and a is the greatest common divisor of J and K. Fig. 2.9 illustrates the procedure. Here the parity-check matrix is depicted together with the received bitstream. The entries in the parity-check matrix which are different from 0 are shown as non-white. The different colors are explained in the following. The windowed decoder works on sub-matrices of the parity-check matrix. W denotes the size of the decoding window: The submatrix in the parity-check matrix for a decoding window of size W has size $WJ'M \times K'M(m_s + W)$. Since variable nodes that are more than $(m_s + 1)K'M$ columns apart cannot be involved in the same check equation, the decoder can aim at decoding the windows separately and sequentially by sliding down J' rows and K' columns right in the graph. Another parameter of the decoder is the target erasure probability $\delta \geq 0$. The decoder aims at reducing the erasure probability of every bit in the codeword to a value no larger than δ. For the windowed decoder, only the so called *targeted bits* need to be recovered with this reliability when decoding the corresponding window. The targeted bits are the first $K'M$ variable nodes within the window. When the window slides to its next position, the targeted bits are another group of symbols. In Fig. 2.9 the targeted

bits are shown in red. The decoded bits colored in green are no longer part of the decoding process.

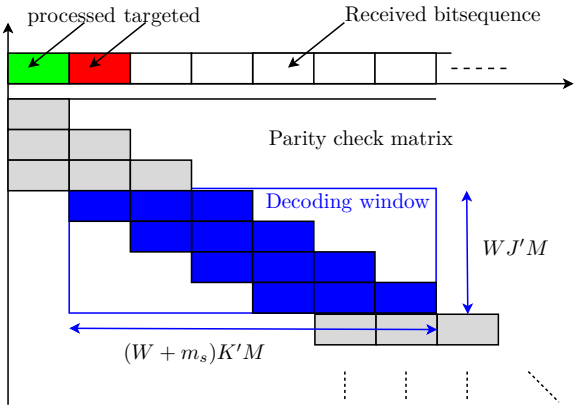


Figure 2.9: Sliding window decoding with $W = 4$, for a $(J, 2J)$-regular LDPC-CC with $m_s = 2$. The decoding window consists of $WJ'M$ rows and $(W + m_s)K'M$ columns. The first K' symbols in the window are the *targeted symbols*.

2.5.4 Termination of LDPC-CCs

A LDPC-CC is terminated if we append to the information sequence a tail of symbols that force the encoder to the zero state at the end of the encoding process. The tail depends on the encoded information and can be calculated easily when using the partial syndrome former realization for encoding as described in the previous section. Assume that an information sequence $\boldsymbol{x}_{[0,L-1]}$ has been encoded, then we have to find the tail $\boldsymbol{y}_{[L,L+\tau-1]}$ of length τn that forces the state $\boldsymbol{p}_{L+\tau-1}$ to the zero state. Inserting $\boldsymbol{p}_{L+\tau-1} = \mathbf{0}$ into (2.66) we get

$$\boldsymbol{y}_{[0,L+\tau-1]}\boldsymbol{H}^T_{[0,L+\tau-1]} = \mathbf{0}_{[0,L+\tau+m_s-1]}. \tag{2.71}$$

From this we get

$$\boldsymbol{y}_{[0,L-1]}\boldsymbol{H}^T_{[0,L-1]} = [\mathbf{0}_{[0,L-1]}|\boldsymbol{p}_L] \tag{2.72}$$

$$\boldsymbol{y}_{[L,L+\tau-1]}\boldsymbol{H}^T_{[L,L+\tau-1]} = [\boldsymbol{p}_L|\mathbf{0}_{[0,\tau-m_s-1]}], \tag{2.73}$$

where the second condition defines a linear system of $(n-k)\tau$ equations to find the tail $\boldsymbol{y}_{[L,L+\tau-1]}$. Note however that not all of the linear equations need to be linearly independent.

It was shown in [Sri05] that if $m_s \to \infty$ almost all $(m_s, J, 2J)$-codes can be terminated with a tail of length $\tau = 2(m_s + 1)$.

Chapter 3

LDPC Convolutional Anytime Codes

In this chapter we focus entirely on the communication problem that is part of the overall problem of stabilizing an unstable plant over a noisy channel. Fig. 3.1 highlights the part of the system that we examine and design here. We develop error-correcting channel codes and the corresponding encoders and decoders that fulfill the anytime reliability requirements formulated in Chapter 2. We propose an ensemble of non-terminated systematic LDPC convolutional codes with increasing memory, and show that over the BEC these codes achieve anytime reliability asymptotically when decoded with an expanding-window message-passing decoder. The corresponding anytime exponents are determined through protograph-based extrinsic information transfer charts. Fundamental complications arising when transmitting with finite block lengths are identified and a combinatorial performance analysis, when transmitting over a *static* BEC with a fixed number of erasures per codeword block, is developed. Although the analysis is developed for a static BEC we show numerically that we can design efficient low-complexity finite-length codes with anytime properties even for the conventional BEC. Furthermore we investigate the performance of the codes when transmitting over the AWGN channel. In the last part of this chapter we compare the proposed codes to two other approaches formulated in literature. Large parts of this chapter are based on [GRTS14] and therefore contain partly verbatim extracts of the paper.

3.1 Code Development

We now first revisit the anytime system model introduced in Chapter 2.2 and then explain the code structure that we have developed in this thesis.

3.1.1 Anytime System Model

The communication problem in the context of anytime theory is often pictured using the anytime system model as shown in Fig. 3.2 already described in Chapter 2.2. A source produces an information sequence of increasing size $\boldsymbol{x}_{[1,t]} = [\boldsymbol{x}_1, \boldsymbol{x}_2 ..., \boldsymbol{x}_t]$,

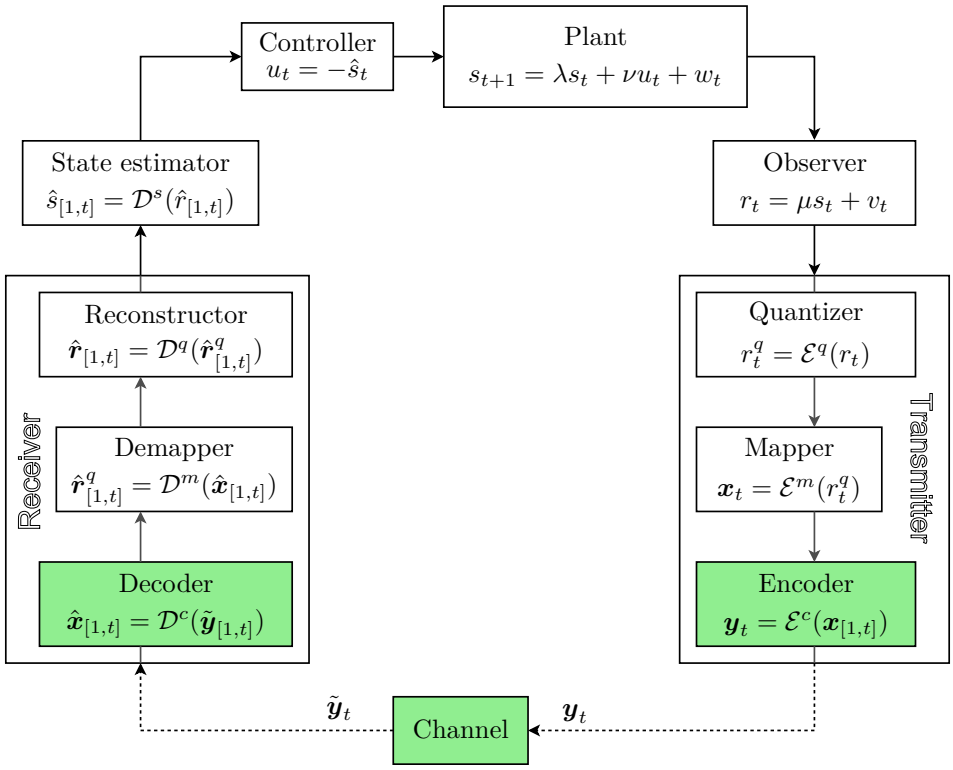


Figure 3.1: System overview with focus on the communication problem.

where $\boldsymbol{x}_i$ is a binary vector of size k and t is a time index. For every new information block $\boldsymbol{x}_t$ the encoder emits a code block $\boldsymbol{y}_t$ of size n which is a function of *all* information blocks seen so far: $\boldsymbol{y}_t = \mathcal{E}^c(\boldsymbol{x}_1, ..., \boldsymbol{x}_t)$. The code block $\boldsymbol{y}_t$ is transmitted over the channel. Note that all $\boldsymbol{y}_t$ are of the same size no matter how long the information sequence $\boldsymbol{x}_{[1,t]}$ is. At the receiver side a corrupted code block $\tilde{\boldsymbol{y}}_t$ is observed at each time step t and fed into the decoder. Based on *all* received code blocks $\tilde{\boldsymbol{y}}_1, ..., \tilde{\boldsymbol{y}}_t$ the decoder produces an estimate $\hat{\boldsymbol{x}}_{[1,t]} = [\hat{\boldsymbol{x}}_1(t), ..., \hat{\boldsymbol{x}}_t(t)] = \mathcal{D}^c(\tilde{\boldsymbol{y}}_1, ..., \tilde{\boldsymbol{y}}_t)$ where $\hat{\boldsymbol{x}}_i(t)$ denotes the estimate of block $\boldsymbol{x}_i$ at time t. The receiver can decide to start decoding at *any time* and can deliver an estimate of *any* information block $\boldsymbol{x}_i$ transmitted so far at *any time*. Note that there is no direct feedback link from the receiver to the transmitter. The transmitter has therefore no knowledge other than the one obtained through the plant on whether a transmission was successful or not. In contrast to conventional LDPC block and convolutional codes, the relevant performance measure for the proposed anytime LDPC-CC ensemble is the anytime capacity, rather than the Shannon capacity. See Section 2.2 for the definition of anytime capacity. The notion of anytime capacity inherently defines the concept of

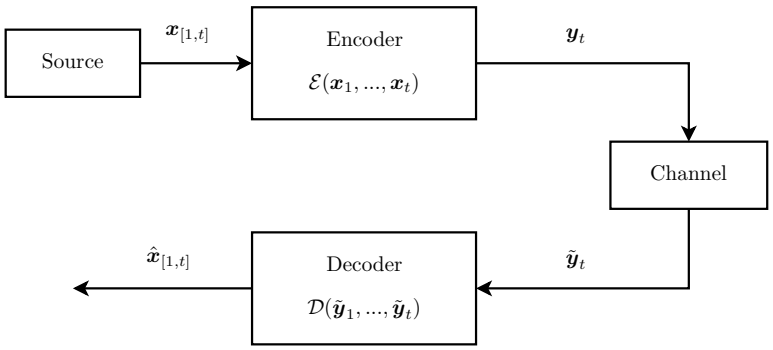


Figure 3.2: Anytime system model.

anytime reliability of the system in Fig. 3.2, given maximum-likelihood decoding, as

$$P(\hat{\boldsymbol{x}}_i(t) \neq \boldsymbol{x}_i | \boldsymbol{x}_{[1,t]} \text{ was transmitted}) \leq \beta 2^{-\alpha d(t,i)}, \tag{3.1}$$

where $d(t,i) = t - i$ denotes the decoding delay between information block i and the most recent block t, β is a constant, and $\alpha > 0$. Thus, the probability of incorrect decoding of block i decays exponentially with the decoding delay $d(t,i)$, which holds for *any* block $\boldsymbol{x}_i$ in the stream. In this work, we are interested in the *operational anytime exponent* α_o, which is the achievable exponent of a particular protograph-based code ensemble, as a first step towards a design framework for anytime LDPC-CC ensembles. It follows that the message-passing decoder erasure probability of the ensemble, as a function of the decoding delay $d(t,i)$ is the measure of interest. A constant decoding delay $\Delta(\epsilon)$, depending on the channel erasure probability, is required until the exponential decay kicks in. The effect of this constant delay is absorbed in the constant β in (3.1).

3.1.2 Code Structure

Here we describe the code structure that is the basis of all anytime codes developed in this thesis. The code structure is similar to the structure of LDPC-Convolutional Codes (LDPC-CCs) introduced in [JFZ99]. Please refer to Chapter 2 for a more detailed introduction to protograph based LDPC-CCs. A number of reasons make LDPC-CCs a suitable starting point for the development of our anytime codes: It has been shown that the parity-check matrix of a linear anytime code must have a lower-left block triangular structure to enable causal operation and anytime reliability [CFZ10], [SH11b]. LDPC-CCs have a lower-left band diagonal structure. Anytime codes are intended to work in a streaming context. LDPC-CCs can be causally encoded in real-time and are therefore suitable for streaming applications [JFZ99]. Due to the delay-sensitivity of the real-time control setup, an anytime decoder needs to be efficient. As for their blockcode counterparts, LDPC-CCs can

be decoded with iterative message-passing algorithms. Moreover sliding-window decoding is possible [PIS+10, CIP+10] enabling efficient trade-offs between decoding latency and reliability. Apart from the desired encoding and decoding possibilities and the excellent decoding thresholds, another reason for why protograph based LDPC-CCs are suitable as a basis for our code design is that they can be analyzed in the asymptotic limit using Protograph-EXIT (P-EXIT) analysis [LC07].

These arguments led us to consider the following general base matrix at time instance t to describe an ensemble of semi-infinite protograph-based LDPC-CCs:

$$\boldsymbol{B}_{[1,t]} = \begin{bmatrix} \boldsymbol{B}_0 & & & & \\ \boldsymbol{B}_1 & \boldsymbol{B}_0 & & & \\ \boldsymbol{B}_2 & \boldsymbol{B}_1 & \boldsymbol{B}_0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ \boldsymbol{B}_{t-1} & \boldsymbol{B}_{t-2} & \dots & \boldsymbol{B}_1 & \boldsymbol{B}_0 \end{bmatrix}. \tag{3.2}$$

Compared to the structure of the base matrix of a protograph based LDPC-CC with memory m_s as given in (2.62), we can see that as the number of time instances grows, the memory m_s also grows. The following notation is used: Each submatrix $\boldsymbol{B}_\ell$ is of size $J_B \times K_B$. Indicated by the index t there are t submatrices $\boldsymbol{B}_\ell$ per row and column, so that the overall size of the protograph at time t is equal to $J_B t \times K_B t$. Given the general protograph structure in (3.2), our first goal is to design a particularly tractable protograph ensemble, and through asymptotic analysis show that the ensemble exhibits the desired anytime properties. We consider an ensemble of regular protographs of rate $R = 1/2$ given by $\boldsymbol{B}_{[1,t]}$ in (3.3), where $t \to \infty$.

$$\boldsymbol{B}_{[1,t]} = \begin{bmatrix} 1 & 1 & & & & & & & \\ 1 & 0 & 1 & 1 & & & & & \\ 1 & 0 & 1 & 0 & 1 & 1 & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & & \\ 1 & 0 & 1 & 0 & 1 & 0 & \dots & 1 & 1 \end{bmatrix}, \tag{3.3}$$

The proposed protograph can, following the structure of (3.2), be compactly written as: $\boldsymbol{B}_{[1,t]}$ with $\boldsymbol{B}_0 = [1\ 1]$ and $\boldsymbol{B}_i = [1\ 0]$ for all $i \geq 1$. In order to obtain the parity-check matrix from the protograph we lift the protograph with the lifting factor M. The details on the lifting are given in Section 2.4.2. For the codes considered here the lifting factor M is equal to the block length of the message $\boldsymbol{x}_t$, that is $M = k$. The basic structure of the parity-check matrix is then given as

$$\boldsymbol{H}_{[1,t]} = \begin{bmatrix} \boldsymbol{I}_k^{rand} & \boldsymbol{I}_k & & & & & \\ \boldsymbol{I}_k^{rand} & \boldsymbol{0}_k & \boldsymbol{I}_k^{rand} & \boldsymbol{I}_k & & & \\ \boldsymbol{I}_k^{rand} & \boldsymbol{0}_k & \boldsymbol{I}_k^{rand} & \boldsymbol{0}_k & \boldsymbol{I}_k^{rand} & \boldsymbol{I}_k & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \tag{3.4}$$

where $\boldsymbol{I}_k^{rand}$ denotes a random permutation matrix of size $k \times k$, $\boldsymbol{I}_k$ denotes the identity matrix of size $k \times k$ and $\boldsymbol{0}_k$ denotes the all zero matrix of size $k \times k$. Although

written above with the same name, all matrices $\boldsymbol{I}_k^{rand}$ are different realizations of a random permutation matrix. If we write $\boldsymbol{H}_{[0,t]}$ similar to (3.2) as

$$\boldsymbol{H}_{[1,t]} = \begin{bmatrix} \boldsymbol{H}_0 & & & & \\ \boldsymbol{H}_1 & \boldsymbol{H}_0 & & & \\ \boldsymbol{H}_2 & \boldsymbol{H}_1 & \boldsymbol{H}_0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ \boldsymbol{H}_{t-1} & \boldsymbol{H}_{t-2} & \dots & \boldsymbol{H}_1 & \boldsymbol{H}_0 \end{bmatrix}, \tag{3.5}$$

the structure can be compactly written as $\boldsymbol{H}_0 = \begin{bmatrix} \boldsymbol{I}_k^{rand} & \boldsymbol{I}_k \end{bmatrix}$ and $\boldsymbol{H}_i = \begin{bmatrix} \boldsymbol{I}_k^{rand} & \boldsymbol{0}_k \end{bmatrix}$ for $i \geq 1$. By choosing the last k columns in $\boldsymbol{H}_0$ to be the identity matrix we design systematic codes. This is desired because we are interested in on-the-fly encoding which can be realized with the encoders described in Section 2.5.2.

3.1.3 Encoding and Decoding

The codes proposed in this chapter can be encoded using the syndrome former encoding realization presented in [PFS+08]. Please refer to Section 2.5.2 for the details of the encoding process. Note that the encoder produces a codeword taking into account the information from the *entire* sequence of messages $\boldsymbol{x}_{[1,t]}$. In order to use the encoding technique from [PFS+08] the code needs to be systematic. We therefore restrict ourselves to the development of systematic codes. The code blocks have then the form $\boldsymbol{y}_t = [\boldsymbol{y}_t^s, \boldsymbol{y}_t^c]$ where $\boldsymbol{y}_t^s = \boldsymbol{x}_t$ denotes the systematic block and $\boldsymbol{y}_t^c$ the parity block. Similarly $\tilde{\boldsymbol{y}}_t = [\tilde{\boldsymbol{y}}_t^s, \tilde{\boldsymbol{y}}_t^c]$, where $\tilde{\boldsymbol{y}}_t^s = \tilde{\boldsymbol{x}}_t$ and $\hat{\boldsymbol{y}}_t = [\hat{\boldsymbol{y}}_t^s, \hat{\boldsymbol{y}}_t^c]$ where $\hat{\boldsymbol{y}}_t^s = \hat{\boldsymbol{x}}_t$. In order to decode the codes proposed in this chapter we modify the sliding-window decoding scheme in [PIS+10] into a corresponding *expanding-window* decoding strategy. The operation of the new decoder structure is shown schematically in Fig. 3.3. When a new block of the code sequence arrives at time instance t the decoder performs message-passing on the *entire* sequence of received blocks from time instance 1 to t. In the next time step the decoding window is expanded to include the next code block. The size of the initial window is $kJ_B \times kK_B$. The lifting factor k of the underlying LDPC-CC determines the size $n = k/R$ of the code blocks transmitted over the channel. For the asymptotic analysis we consider the performance in the limit of infinite block lengths ($k \to \infty$), whereas for the finite-length behavior we are interested in keeping k small to provide timely interaction.

Notation of Variable and Check Nodes

The following notation and terms are used: Consider the protograph $\boldsymbol{B}_{[1,t]}$ given in (3.3) and a code block $\boldsymbol{y}_t = [\boldsymbol{y}_t^s, \boldsymbol{y}_t^c]$. A variable node of the protograph, or the corresponding k variable nodes of the derived graph, are termed *information variable nodes* if they correspond to the systematic block $\boldsymbol{y}_t^s$ within a code block and *parity variable nodes* if they correspond to the parity block $\boldsymbol{y}_t^c$ within a code

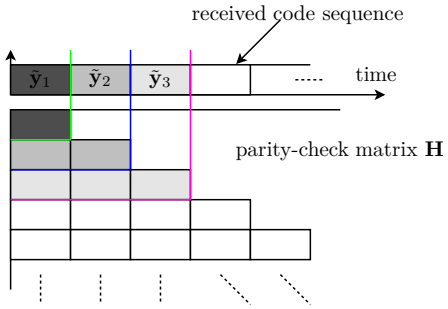


Figure 3.3: Expanding-window-decoding over three time instances.

block. Fig. 3.4 shows the protograph when going from $t = 3$ to $t = 4$. The check-node index i in the protograph corresponds to the block index i. Please refer to Section 2.4.2 for details on the protograph description. The variable nodes can also be tracked by the block index where information variable nodes have index $2i - 1$ and parity variable nodes have index $2i$, respectively for $i = 1, 2, \ldots, t$. However, for notational reasons, it can be convenient to use a common variable-node index, j, to easily distinguish between variable nodes and check nodes. In this case odd j's refer to information variable nodes and even j's refer to parity variable nodes, as shown in Fig. 3.4. The decoding delay is determined as $d(t, i) = d(t, \lceil j/2 \rceil)$ in terms of block index and variable-node index, respectively. It should be clear from

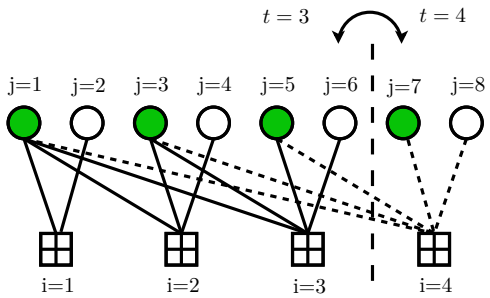


Figure 3.4: The protograph when going from time $t = 3$ to $t = 4$. Green nodes are information variable nodes and have increasing degrees and white nodes are parity variable nodes and have constant degrees equal to 1.

the context whether we are talking about variable nodes of the protograph or of the derived graph.

3.1.4 Characteristics

The proposed protograph design has a number of useful features for anytime behavior. The lower-left block diagonal structure is essential for block-streaming, which in turn enables anytime reliability. Together with the ease of systematic encoding it allows for information variable nodes to be appended on-the-fly, and to subsequently be part of the encoding process for every new code block. For each new information variable node, there is one new parity variable node which is only connected to the new check node in the corresponding block. It follows that all parity nodes are of degree 1. The importance of these degree-1 nodes becomes clear in the asymptotic analysis in Section 3.2.1. The protograph structure is also amenable to parallel edges, thus making it more dense, as well as to pruning edges either deterministically or randomly. A combination of these modifications provides the freedom to obtain a wide range of operational anytime exponents. The structure also allows for introducing irregularity as well as further randomness. In Chapter 4 some modifications of the code structure are presented that are particularly useful when introducing practical constraints on, e.g., the memory or the complexity. In contrast to conventional LDPC block codes and LDPC-CCs, the proposed anytime LDPC-CC ensemble has information variable node degrees and check node degrees that grow with t. At time t the protograph has t information variable nodes, t parity variable nodes, and t check nodes, respectively. The parity variable nodes are all of degree 1, while the degrees of the information variable nodes are decreasing by one with the time index, i.e., the information variable node at block τ is of degree $t-\tau+1$. In contrast the check node degrees are increasing by one with the time index τ, i.e., the check node at block τ is of degree $\tau+1$. It follows that the averaged variable node degree is $\frac{t+3}{2}$, where the averaged information variable node degree is $\frac{t+1}{2}$ and the averaged parity variable node degree is 1. Correspondingly the averaged check node degree is $\frac{t+3}{2}$. The property of growing averaged degrees is akin to rateless codes; however, a significant difference is that for rateless codes the number of information variable nodes stays fixed while for our proposed anytime codes the rate remains fixed. The resulting erasure probability behavior at the output of the decoder is closely related to this property.

3.2 Performance on the BEC

We now analyze the performance of the proposed codes when transmitting over the BEC. This is done first in the asymptotic limit of infinite block length and then for finite block lengths.

3.2.1 Asymptotic Analysis

In this section by using P-EXIT analysis we show that the proposed code structure has asymptotically anytime properties. To this end, we analyze the average decoder erasure probability performance of the protograph ensemble with expanding-

window message-passing as a function of the decoding delay, first in the limit of an infinite number of decoder iterations ($\ell \to \infty$), and subsequently, in the limit of an infinite block length ($t \to \infty$). For the proposed codes the information bits with the longest decoding delay are most reliable, while information bits decoded with the lowest decoding delay are least reliable. The objective of the analysis is to determine the exponential decay of the average decoder erasure probability performance of the protograph ensemble as a function of the decoding delay, as well as the corresponding operational anytime exponent. The decoder erasure performance over time is evaluated using the P-EXIT algorithm [LC07] working on the matrices $\boldsymbol{I_{Av,t}}$ and $\boldsymbol{I_{Ev,t}}$ containing the mutual information of the messages sent between variable nodes and check nodes. A brief introduction to P-EXIT analysis is given in Chapter 2. With reference to the variable-node and check-node indexing in Fig. 3.4, the entries of $\boldsymbol{I_{Ev,t}}$ and $\boldsymbol{I_{Av,t}}$ in position (i,j) at time t when ℓ iterations are performed are denoted as $I^{\ell}_{Ev,t}(i,j)$ and $I^{\ell}_{Av,t}(i,j)$. The vector of a-posteriori decoder erasure probabilities at time t for a code of rate $R = 1/2$ is given as $\boldsymbol{P}_{\mathrm{APP},t} = [P_{\mathrm{APP},t}(1), ..., P_{\mathrm{APP},t}(2t)]$ where $P_{\mathrm{APP},t}(j)$ denotes the decoder erasure probability of variable node j for the protograph at time t. For the BEC $\boldsymbol{P}_{\mathrm{APP},\boldsymbol{t}} = 1 - \boldsymbol{I}_{\mathrm{APP},\boldsymbol{t}}$. The behavior of the decoder erasure probability of one particular variable node (index j) over time is described with the vector $\boldsymbol{P}^{j}_{\mathrm{APP},[1,t]} = [P_{\mathrm{APP},1}(j), P_{\mathrm{APP},2}(j), ..., P_{\mathrm{APP},t}(j)]$. In every time step t the decoding window is described by its underlying protograph $\boldsymbol{B}_{[1,t]}$. Through P-EXIT analysis a vector $\boldsymbol{P}_{\mathrm{APP},t}$ of a-posteriori decoder erasure probabilities is obtained. From these vectors we determine the a-posteriori decoder erasure probability of one particular variable node over time as $\boldsymbol{P}^{j}_{\mathrm{APP},[1,t]}$. Here we assume a BEC with erasure probability ϵ such that $R < 1 - \epsilon$. We modify the P-EXIT algorithm to accommodate the protograph structure in (3.4) and the operation of the expanding-window decoder. The P-EXIT update equations at time t are compactly written as follows: For all (i,j) where $B_t(i,j) = 0$ the entries of $I_{Ev,t}(i,j)$ and $I_{Av,t}(i,j)$ are equal to zero and for all (i,j) if $B_t(i,j) \neq 0$,

- **Initialization:** $$I^{0}_{Ev,t}(i,j) = 1 - \epsilon \tag{3.6}$$
- **Check node i to variable node j update**:
$$I^{\ell+1}_{Av,t}(i,j) = \prod_{s=1, s\neq j}^{2i} I^{\ell}_{Ev,t}(i,s)^{B_t(i,s)} I^{\ell}_{Ev,t}(i,j)^{B_t(i,j)-1} \tag{3.7}$$
- **Variable node j to check node i update**:
$$I^{\ell+1}_{Ev,t}(i,j) = 1 - \epsilon \prod_{s=\lceil j/2 \rceil, s\neq i}^{t} (1 - I^{\ell+1}_{Av,t}(s,j))^{B_t(s,j)} (1 - I^{\ell+1}_{Av,t}(i,j))^{B_t(i,j)-1} \tag{3.8}$$
- **APP mutual information evaluation**:

$$I_{\mathrm{APP},t}(j) = 1 - \epsilon \prod_{s=\lceil j/2 \rceil}^{t} (1 - I_{Av,t}^{\infty}(s,j))^{B_t(s,j)} \tag{3.9}$$

The product indices are modified to take into account the block triangular structure and the code rate.

Observations and Conjectures Based on P-EXIT Numerical Evaluation

To gain some preliminary insight, we first present a series of numerical results based on iterating the P-EXIT equations for the protograph depicted in Fig. 3.4. Based on the numerical results, we highlight essential observations, and state a number of conjectures to support our further analysis. Unless stated otherwise, in the simulations we assume a channel erasure probability of $\epsilon = 0.3$ but the results have been verified for other erasure probabilities as well. Furthermore we assume a rate of $R = 1/2$. For this rate all stated observations are valid for channel erasure probabilities $\epsilon \leq 0.5$.

In Fig. 3.5 a plot of $D_{Av,t}^{\ell}(i, i{+}1, j) = I_{Av,t}^{\ell}(i,j) - I_{Av,t}^{\ell}(i{+}1,j)$ is presented for $t = 20$ and $t = 40$, respectively, as a function of the number of iterations ℓ. For both $t = 20$ and $t = 40$ we focus on check nodes $i = 10$, $i = 15$, while for $t = 40$ we further consider check nodes $i = 30$, $i = 35$. This check-node selection enables us to observe the behavior for different check nodes with the same decoding delay but different block length t, as well as the behavior of the same check nodes for different decoding delay at different block lengths. Although difficult to see, we show the behavior for all of the connected variable nodes going from $j = 1$ to $j = 2i$ for each considered check node. In the bundle of curves associated with a check node the lowest curve is for $j = 1$, with increasing index moving upwards through the curves concluding with $j = 2i$. Correspondingly, in Fig. 3.6 a similar plot of $D_{Ev,t}^{\ell}(i, j, j{+}2) = I_{Ev,t}^{\ell}(i,j) - I_{Ev,t}^{\ell}(i, j{+}2)$ is presented for $t = 20$ and $t = 40$, respectively, as a function of the number of iterations ℓ. For $t = 20$ and $t = 40$ ($2t$ variable nodes) we focus on information variable nodes $j = 5$, $j = 15$, while for $t = 40$ we further consider information variable nodes $j = 45$, $j = 55$. This variable-node selection enables us to observe the behavior for different variable nodes with the same decoding delay but different block length t, as well as the behavior of the same variable nodes for different decoding delay at different block lengths. As before, we include the behavior for all of the connected check nodes going from $i = \lceil j/2 \rceil$ to $i = t$ for each considered information variable node. In the bundle of curves associated with an information variable node the lowest curve is for $i = \lceil j/2 \rceil$, with increasing index moving upwards through the curves concluding with $i = t$. Based on the numerical results in Figs. 3.5 and 3.6, and related results when varying the erasure rate ϵ, we note the following two observations:

Obs. 1: For $\infty > \ell \geq \bar{\ell}_{D_{Av,t}}$, $D_{Av,t}^{\ell}(i, i+1, j)$ converges to some value $D_{Av,t}^{\bar{\ell}_{D_{Av,t}}}(i, i+1, j) \geq 0$; Likewise, for some $\infty > \ell \geq \bar{\ell}_{D_{Ev,t}}$, and odd j,

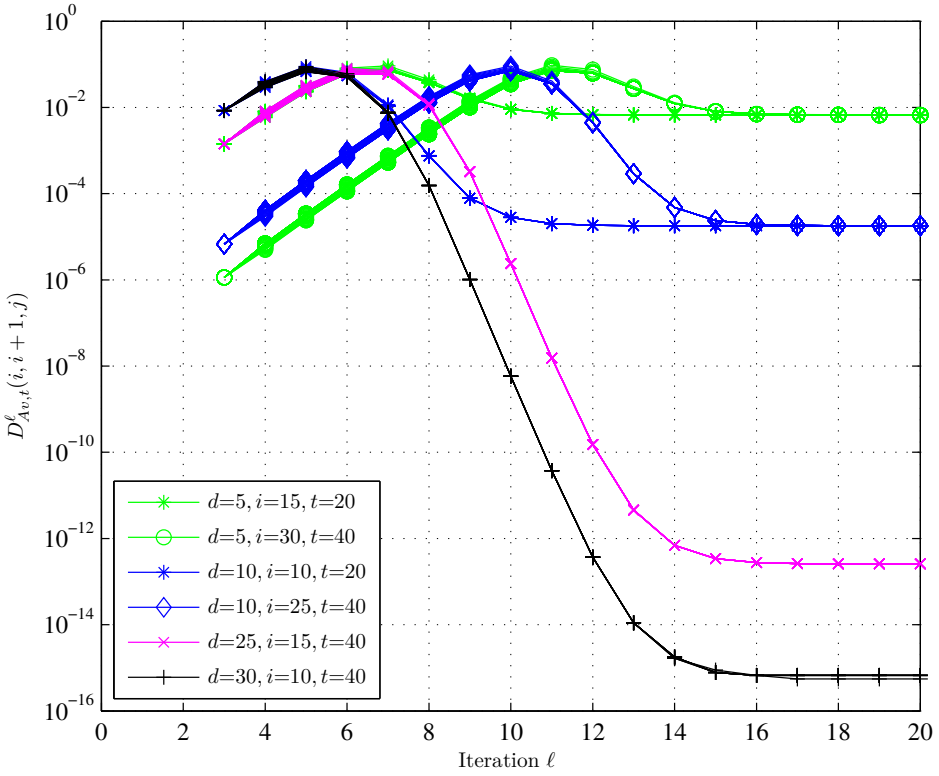


Figure 3.5: $D^{\ell}_{Av,t}(i,i+1,j) = I^{\ell}_{Av,t}(i,j) - I^{\ell}_{Av,t}(i+1,j)$ as a function of the number of iterations ℓ for $t = 20$ and $t = 40$ and $\epsilon = 0.3$.

$D^{\ell}_{Ev,t}(i,j,j+2)$ converges to some value $D^{\bar{\ell}_{D_{Ev,t}}}_{Ev,t}(i,j,j+2) \geq 0$;

Obs. 2: $D^{\bar{\ell}_{D_{Av,t}}}_{Av,t}(i,i+1,j)$ and $D^{\bar{\ell}_{D_{Ev,t}}}_{Ev,t}(i,j,j+2)$ decrease with increasing $d(t,i)$ and decreasing ϵ;

We further state the following Conjecture:

Conjecture 3.1. *Consider an $R = 1/2$-anytime LDPC-CC ensemble defined by (3.3), for transmission over a BEC. Then for the mutual information defined in (3.7) and (3.8), respectively, and for all $\epsilon \leq 0.5$, we conjecture that*

$$\begin{aligned} D^{k}_{Av,t}(i,i+1,j) &= I^{\ell}_{Av,t}(i,j) - I^{\ell}_{Av,t}(i+1,j) \geq 0 \\ &\Leftrightarrow I^{\ell}_{Av,t}(i,j) \geq I^{\ell}_{Av,t}(i+1,j) \\ D^{\ell}_{Ev,t}(i,j,j+2) &= I^{\ell}_{Ev,t}(i,j) - I^{\ell}_{Ev,t}(i,j+2) \geq 0 \end{aligned} \tag{3.10}$$

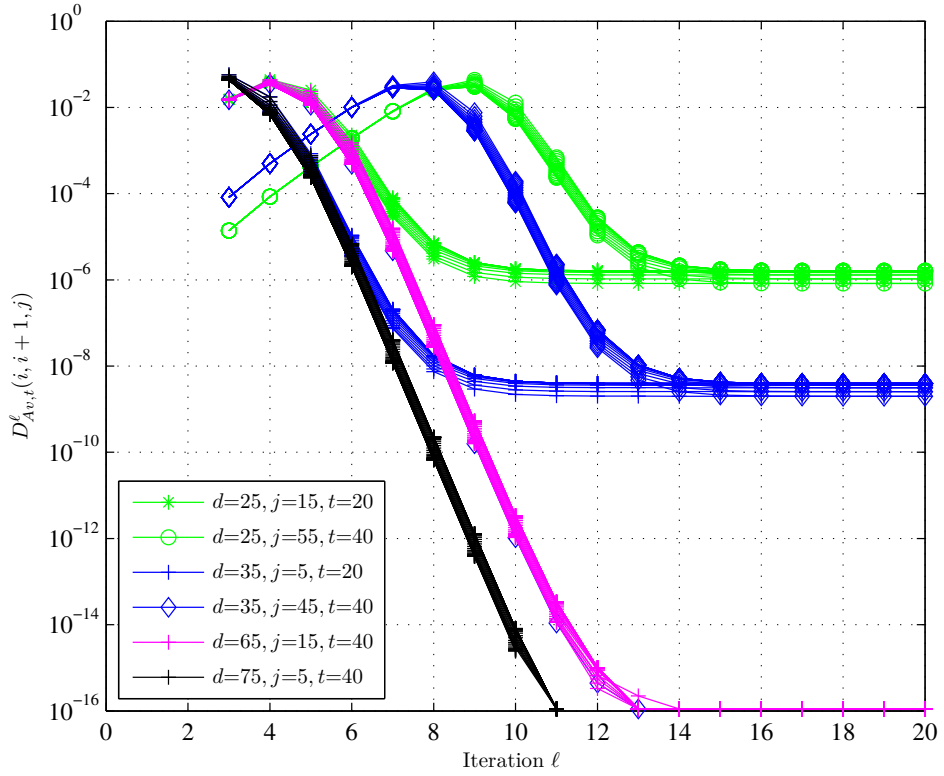


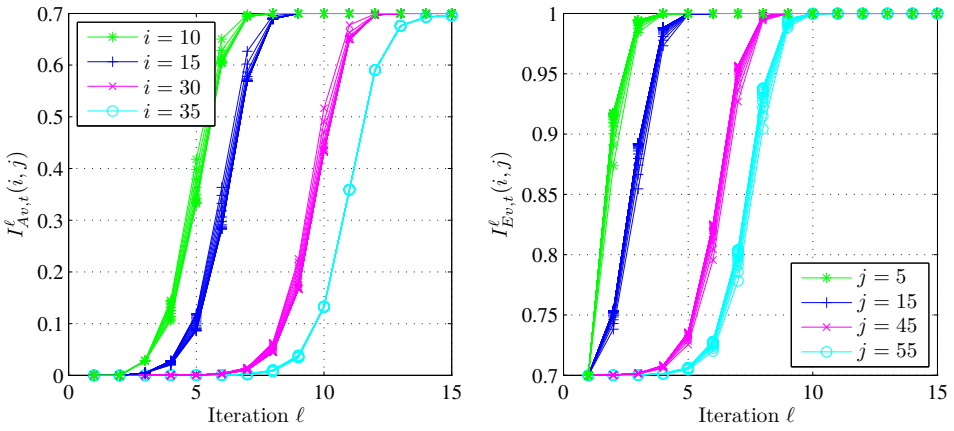
Figure 3.6: $D^{\ell}_{Ev,t}(i,j,j+2) = I^{\ell}_{Ev,t}(i,j) - I^{\ell}_{Ev,t}(i,j+2)$ as a function of the number of iterations ℓ for $t = 20$ and $t = 40$ and $\epsilon = 0.3$.

$$\Leftrightarrow I^{\ell}_{Ev,t}(i,j) \geq I^{\ell}_{Ev,t}(i,j+2), \ \textit{for } j \ \textit{odd}. \tag{3.11}$$

□

The intuitive argument for (3.10) is that due to the structure of the protograph $I^{\ell}_{Av,t}(i,j)$ decreases with increasing i since the number of connected information variable nodes increases. The intuitive argument for (3.11) is similarly that $I^{\ell}_{Ev,t}(i,j)$ decreases with increasing j since an information variable node with a higher index j is connected to fewer check nodes than an information variable node with a lower index j.

In Fig. 3.7a we have further investigated the evolution of the mutual information $I^{\ell}_{Av,t}(i,j)$ as a function of the number of iterations ℓ for $t = 40$. Again, we consider check nodes $i = 10$, $i = 15$, $i = 30$, $i = 35$. To obtain additional information, we plot $I^{1000}_{Av,t}(i,j)$ as a function of the decoding delay $d(t,i)$ in Fig. 3.8. The corresponding plots for $I^{\ell}_{Ev,t}(i,j)$ are presented in Figs. 3.7b and 3.9. As before, we consider information variable nodes $j = 5$, $j = 15$, $j = 45$, $j = 55$. Based on the



(a) $I^{\ell}_{Av,t}(i,j)$ (b) $I^{\ell}_{Ev,t}(i,j)$

Figure 3.7: $I^{\ell}_{Av,t}(i,j)$ and $I^{\ell}_{Ev,t}(i,j)$ as a function of the number of iterations ℓ for $t = 40$, $\epsilon = 0.3$.

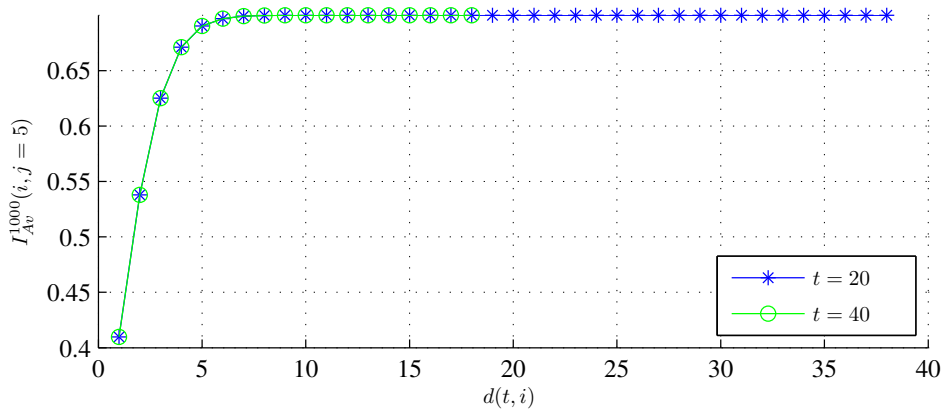


Figure 3.8: $I^{1000}_{Av,t}(i,j)$ as a function of $d(t,i)$ for variable node $j = 5$ for $t = 40$ and $\ell = 1000$, $\epsilon = 0.3$.

numerical results in Figs. 3.7a, 3.7b, 3.8, 3.9 and similar results when varying the erasure rate ϵ, we make the following observation:

Obs. 3: For a sufficient number of iterations $\infty > \ell \geq \bar{\ell}_{I_{Av,t}}$, $I^{\ell}_{Av,t}(i,j)$ converges to some value $I^{\bar{\ell}_{I_{Av,t}}}_{Av,t}(i,j)$ bounded by

$$1 - \epsilon \geq I^{\bar{\ell}_{I_{Av,t}}}_{Av,t}(i,j) \geq 0, \text{ for } j \text{ odd} \tag{3.12}$$

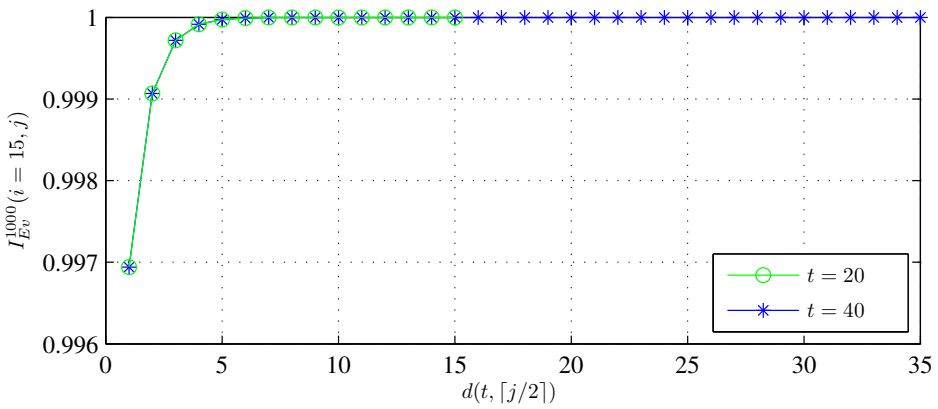


Figure 3.9: $I_{Ev,t}^{1000}(i,j)$ as a function of $d(t,i)$ for check node $i=15$ for $t=40$ and $\ell=1000$, $\epsilon=0.3$.

$$1 \geq I_{Av,t}^{\bar{\ell}_{I_{Av,t}}}(i,j) \geq 0, \text{ for } j \text{ even.} \tag{3.13}$$

Furthermore, for a sufficient number of iterations $\infty > \ell \geq \bar{\ell}_{I_{Ev,t}}$, $I_{Ev,t}^{\ell}(i,j)$ converges to some value $I_{Ev,t}^{\bar{\ell}_{I_{Ev,t}}}(i,j)$ bounded by

$$1 \geq I_{Ev,t}^{\bar{\ell}_{I_{Ev,t}}}(i,j) \geq 1-\epsilon, \text{ for } j \text{ odd} \tag{3.14}$$

We note that $I_{Ev,t}^{\ell}(i,j) = 1-\epsilon$ for j even, and all i, ℓ, and t.

We further state the following conjecture:

Conjecture 3.2. *For the same code ensemble and transmission channel considered in Conjecture 3.1, and all channel erasure probabilities $\epsilon \leq 0.5$, let $\Delta_{I_{Av,t}}$ and $\Delta_{I_{Ev,t}}$ be initial decoding delays. For $d(t,i) \geq \Delta_{I_{Av,t}}$ and $\infty > \ell \geq \bar{\ell}_{I_{Av,t}}$, $I_{Av,t}^{\ell}(i,j)$ converges to some value $I_{Av,t}^{\bar{\ell}_{I_{Av,t}}}(i,j)$ bounded by*

$$1-\epsilon \geq I_{Av,t}^{\bar{\ell}_{I_{Av,t}}}(i,j) \geq 1-\epsilon-\delta_{I_{Av,t}}(\Delta_{I_{Av,t}}), \text{ for } j \text{ odd} \tag{3.15}$$

$$1 \geq I_{Av,t}^{\bar{\ell}_{I_{Av,t}}}(i,j) \geq 1-\delta_{I_{Av,t}}(\Delta_{I_{Av,t}}), \text{ for } j \text{ even.} \tag{3.16}$$

Likewise, for $d(t,\lceil j/2\rceil) \geq \Delta_{I_{Ev,t}}$ and $\infty > \ell \geq \bar{\ell}_{I_{Ev,t}}$, $I_{Ev,t}^{\ell}(i,j)$ converges to some value $I_{Ev,t}^{\bar{\ell}_{I_{Ev,t}}}(i,j)$ bounded by

$$1 \geq I_{Ev,t}^{\bar{\ell}_{I_{Ev,t}}}(i,j) \geq 1-\delta_{I_{Ev,t}}(\Delta_{I_{Ev,t}}), \text{ for } j \text{ odd.} \tag{3.17}$$

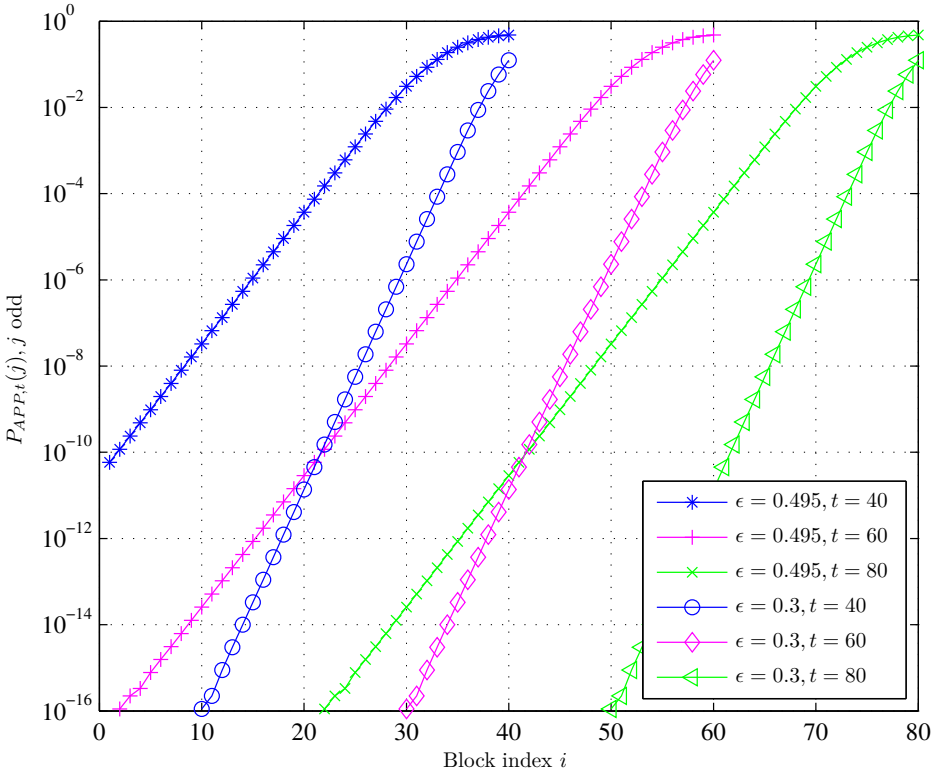


Figure 3.10: $P_{\mathrm{APP},t}(j)$ (j odd) as a function of the information block index $i = \lceil j/2 \rceil$, for $\epsilon = 0.3, 0.495$ and $t = 40, 60, 80$.

As t increases $\delta_{I_{Av,t}}(\Delta_{I_{Av,t}})$ and $\delta_{I_{Ev,t}}(\Delta_{I_{Ev,t}})$ decrease towards zero, and thus, as $t \to \infty$, we obtain

$$I_{Av,t}^{\bar{\ell}_{I_{Av,t}}}(i,j) \to 1 - \epsilon, \text{ for } j \text{ odd} \tag{3.18}$$

$$I_{Av,t}^{\bar{\ell}_{I_{Av,t}}}(i,j) \to 1, \text{ for } j \text{ even} \tag{3.19}$$

$$I_{Ev,t}^{\bar{\ell}_{I_{Ev,t}}}(i,j) \to 1, \text{ for } j \text{ odd}, \tag{3.20}$$

providing three useful limits for asymptotic analysis. □

In Fig. 3.10 a plot of $P_{APP,t}(j)$ for j odd (the t information variable nodes) is shown as a function of the block index $i = \lceil j/2 \rceil$, for $t = 40$, $t = 60$ and $t = 80$, as well as channel erasure probabilities of $\epsilon = 0.3$ and $\epsilon = 0.495$. In Fig. 3.11 a similar plot is presented for $t = 40$ and channel erasure probabilities

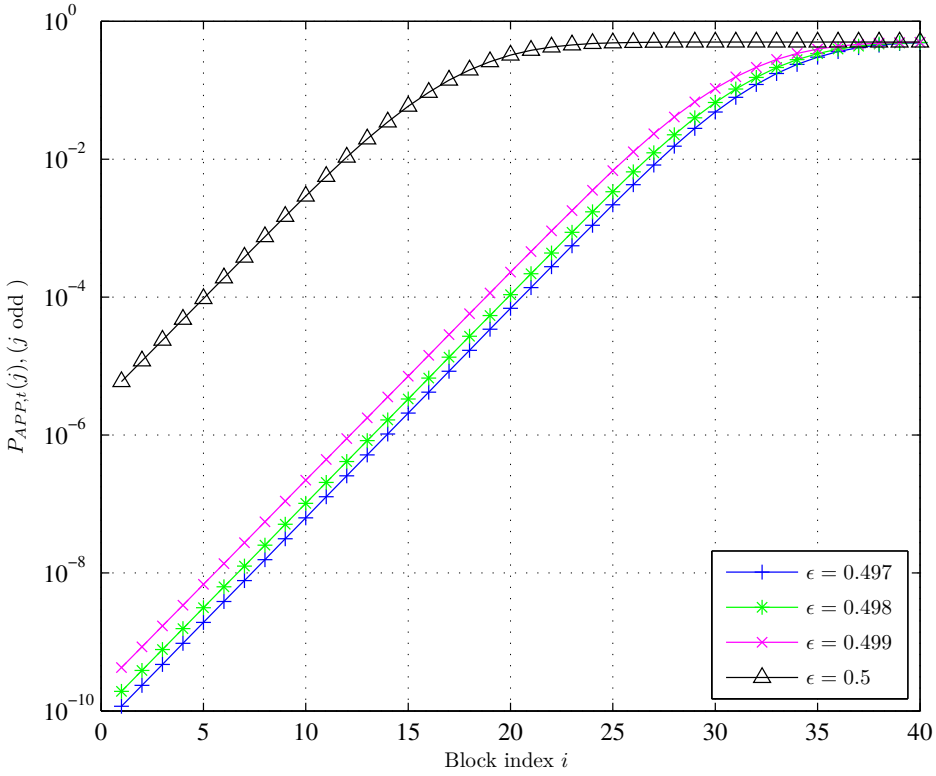


Figure 3.11: $P_{\mathrm{APP},t}(j)$ (j odd) as a function of the information block index $i = \lceil j/2 \rceil$,for $\epsilon = 0.497, 0.498, 0.499, 0.5$, and $t = 40$.

$\epsilon = 0.497, 0.498, 0.499, 0.5$. Based on the numerical results in Figs. 3.10 and 3.11, and related results, we now make three observations:

Obs. 4: For a particular information bit position j odd, $P_{APP,t}(j)$ decreases and $d(t, \lceil j/2 \rceil) = d(t, i)$ increases as t increases;

Obs. 5: For an information variable node the behavior of $P_{APP,t}(j)$ (j odd) is characterized by two phases for all $\epsilon \leq 0.5$, both as a function of the information of the block index and as a function of the decoding delay $d(t, i)$; namely the initial phase and the steady-state phase. The initial phase extends $\Delta(\epsilon)$ information block indices, corresponding to an initial decoding delay of $\Delta(\epsilon)$, before the expected exponential decay kicks in;

Obs. 6: The initial decoding delay, $\Delta(\epsilon)$, is a function of ϵ. It is not possible to determine the value of $\Delta(\epsilon)$ exactly based on numerical results.

We also state a final conjecture as follows:

Conjecture 3.3. *For the same code ensemble and transmission channel considered in Conjecture 3.1, consider the behavior of $P_{APP,t}(j)$ for information variables only (odd j's). As t increases, $d(t,i)$ increases and $P_{APP,t}(j)$ decreases. As $t \to \infty$,*

$$\begin{aligned} P_{APP,t}(j) \to 0 \text{ for all } \epsilon \leq 0.5 \\ \text{and all } j = 2i-1 \text{ for } i = 1,2,3,\ldots, \end{aligned} \tag{3.21}$$

where, after an initial decoding delay of $\Delta(\epsilon)$, the decay is exponential according to the channel erasure probability and the protograph structure. □

Fixed Point Discussion

Based on the two inequalities (3.10), (3.11) from Conjecture 3.1, and the two P-EXIT update equations (3.7) and (3.8), we immediately obtain the following two inequalities

$$I^{\ell}_{Av,t}(i,j+j') \geq I^{\ell}_{Av,t}(i,j) \text{ and } I^{\ell}_{Ev,t}(i+1,j) \geq I^{\ell}_{Ev,t}(i,j). \tag{3.22}$$

Furthermore, based on inequalities (3.10), (3.11), and (3.22) we can show by induction that the mutual information $I^{\ell}_{Ev,t}(i,j)$ of an information variable node is monotonically increasing over the number of iterations and over time; that is

$$I^{\ell}_{Ev,t+1}(i,j) \geq I^{\ell}_{Ev,t}(i,j) \text{ and } I^{\ell+1}_{Ev,t}(i,j) \geq I^{\ell}_{Ev,t}(i,j). \tag{3.23}$$

The corresponding relations for $I^{\ell}_{Av,t}(i,j)$ follow immediately through the update equations as

$$I^{\ell}_{Av,t+1}(i,j) \geq I^{\ell}_{Av,t}(i,j) \text{ and } I^{\ell+1}_{Av,t}(i,j) \geq I^{\ell}_{Av,t}(i,j). \tag{3.24}$$

For all parity variable nodes the update equation of $I^{\ell+1}_{Ev,t}(i,j)$ reduces to $I^{\ell+1}_{Ev,t}(i,j) = 1-\epsilon$ for all i, ℓ and t. Thus the mutual information I_{Ev} of a parity variable node is constant. For all information variable nodes (odd j's) we can derive the following recursive equation for $I^{\ell+1}_{Ev,t}(i,j)$ based on $I^{\ell}_{Ev,t}(i,j)$:

$$I^{\ell+1}_{Ev,t}(i,j) = 1-\epsilon \prod_{\substack{s=\lceil j/2 \rceil \\ s \neq i}}^{t} \left(1-(1-\epsilon) \prod_{\substack{u=1 \\ u \neq j}}^{2s-1} I^{\ell}_{Ev,t}(s,u)^{B_t(s,u)} \right), \tag{3.25}$$

where the variable u runs from 1 to $2s-1$ only since $I^{\ell}_{Ev}(s,2s)$ is always equal to $1-\epsilon$ as argued above. For an infinite number of iterations we write: $I^{\infty}_{Ev,t}(i,j) := \lim_{\ell \to \infty} I^{\ell+1}_{Ev,t}(i,j)$, for odd j's. Due to the degree-1 parity variable nodes in the

protograph we can upper bound $I_{Av,t}(i,j)$ for odd j's, and for an infinite number of iterations, by the constant term $1-\epsilon$,

$$\begin{aligned} I^{\infty}_{Av,t}(i,j) &= \prod_{s=1,s\neq j}^{2i-1} I^{\infty}_{Ev,t}(i,s)^{B(i,s)} I^{\infty}_{Ev,t}(i,2i)^{B(i,2i)} \\ &= (1-\epsilon) \prod_{s=1,s\neq j}^{2i-1} I^{\infty}_{Ev,t}(i,s)^{B(i,s)} \\ &\leq (1-\epsilon). \end{aligned} \tag{3.26}$$

Properties of $P_{\mathbf{APP},t}(j)$ for odd j's

In this subsection we will only consider information variable nodes (odd j's). Since $P_{\mathrm{APP},t}(j)$ is equal to $1 - I_{\mathrm{APP},t}(j)$ we can write $P_{\mathrm{APP},t}(j)$ as

$$\begin{aligned} P_{\mathrm{APP},t}(j) &= \epsilon \prod_{s=\lceil j/2\rceil}^{t} (1-I^{\infty}_{Av,t}(s,j))^{B_t(s,j)} \\ &= \epsilon \prod_{s=\lceil j/2\rceil+1}^{t} (1-I^{\infty}_{Av,t}(s,j))^{B_t(s,j)} \\ &\qquad (1-I^{\infty}_{Av,t}(\lceil j/2\rceil,j))^{B_t(\lceil j/2\rceil,j)}. \end{aligned} \tag{3.27}$$

Now we first invoke inequality (3.22) and then (3.26) from the previous section to obtain

$$\begin{aligned} P_{\mathrm{APP},t}(j) &\geq \epsilon \prod_{s=\lceil j/2\rceil+1}^{t} \big(1 - I^{\infty}_{Av,t}(s,j+2)\big)^{B_t(s,j+2)} \\ &\qquad (1 - I^{\infty}_{Av,t}(\lceil j/2\rceil,j))^{B_t(\lceil j/2\rceil,j)} \qquad (3.28) \\ &\geq \epsilon \prod_{s=\lceil j/2\rceil+1}^{t} \big(1 - I^{\infty}_{Av,t}(s,j+2)\big)^{B_t(s,j+2)} \epsilon^{B_t(\lceil j/2\rceil,j)} \qquad (3.29) \end{aligned}$$

for odd j's. Replacing j by $j+2$ in (3.27) we observe that the first term above is equal to $P_{\mathrm{APP},t}(j+2)$ and thus if $B_t(\lceil j/2\rceil,j)=1$,

$$P_{\mathrm{APP},t}(j) \geq P_{\mathrm{APP},t}(j+2)\epsilon. \tag{3.30}$$

This is the first important property of $P_{\mathrm{APP},t}(j)$ describing the behavior over information blocks j: The decoder erasure probability of two consecutive received information blocks only differs by a factor of ϵ. The second important property describes the behavior over time. Combining inequality (3.24) and (3.10) we see

that $I_{Av,t+1}(i,j) \geq I_{Av,t}(i+1,j)$. We can then upper bound $P_{\text{APP},t+1}(j)$ as follows

$$\begin{aligned} P_{\text{APP},t+1}(j) &= \epsilon \prod_{s=\lceil j/2 \rceil}^{t+1} \left(1-I_{Av,t+1}^{\infty}(s,j)\right)^{B_{t+1}(s,j)} \\ &\leq \epsilon \prod_{s=\lceil j/2 \rceil}^{t} \left(1-I_{Av,t}^{\infty}(s,j)\right)^{B_{t}(s,j)} \epsilon^{B_t(\lceil j/2 \rceil,j)}. \end{aligned} \tag{3.31}$$

We notice above that the first term equals (3.27) and thus if $B_t(\lceil j/2 \rceil, j) = 1$ we obtain the second important property of $P_{\text{APP},t}(j)$, for odd j's, as:

$$P_{\text{APP},t+1}(j) \leq P_{\text{APP},t}(j)\epsilon. \tag{3.32}$$

Steady-State Behavior

As discussed earlier, the behavior of $P_{\text{APP},t}(j)$ for information variable nodes can be divided into two phases. The initial phase starts directly after the first decoding attempt of information block $\tilde{\boldsymbol{x}}_j$. Steady-state behavior is obtained with some delay $\Delta(\epsilon)$. Assuming we are in steady-state, we have

$$\begin{aligned} P_{\text{APP},t}(j) &= \epsilon \prod_{s=\lceil j/2 \rceil}^{t} \left(1 - I_{Av,t}^{\infty}(s,j)\right)^{B_t(s,j)} \\ &= \epsilon \prod_{s=\lceil j/2 \rceil}^{t} \left(1 - (1-\epsilon)\right)^{B_t(s,j)}, \end{aligned} \tag{3.33}$$

where we have applied the limits conjectured in Conjecture 3.2. If $B_t(s,j) = 1$ for $s \in \{\lceil j/2 \rceil, \lceil j/2 \rceil + 1, ..., t\}$ then

$$P_{\text{APP},t}(j) = \epsilon^{t-\lceil j/2 \rceil + 1} = \epsilon^{t-\lceil (2i-1)/2 \rceil + 1} = \epsilon^{t-i+1}. \tag{3.34}$$

It follows directly that

$$P_{\text{APP},t}(j+2) = \epsilon^{t-i}, \text{ and thus } P_{\text{APP},t}(j) = P_{\text{APP},t}(j+2)\epsilon \tag{3.35}$$

$$P_{\text{APP},t+1}(j) = \epsilon^{t-i+2}, \text{ and thus } P_{\text{APP},t+1}(j) = P_{\text{APP},t}(j)\epsilon, \tag{3.36}$$

which have the following meaning: The decoder erasure performances have the same shape across different information variable nodes and are only shifted versions of each other. Also the decoder erasure probability decays exponentially over time. It follows that the code has anytime reliability, and thus, we can substitute the bound from (2.33) into the expression above to get the corresponding anytime exponent. For the considered protograph the operational anytime exponent is equal to

$$\alpha_o = -\log_2(\epsilon). \tag{3.37}$$

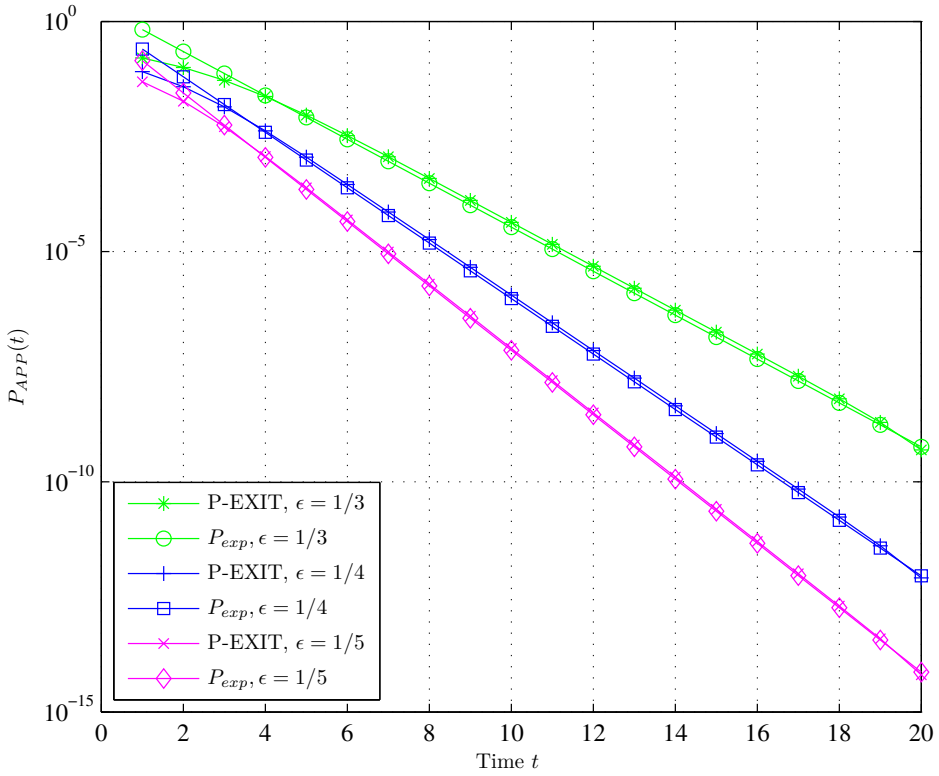


Figure 3.12: Asymptotic decoder erasure probability of the first block $P^1_{APP,[1,t]}$ together with curves decaying with the exponents $\alpha_o = -\log_2(\epsilon)$.

Numerical Examples

To support the derivation of the anytime exponent, we compare the slopes of the asymptotic performance of the previously proposed protograph obtained by P-EXIT analysis with the slope determined by the corresponding determined anytime exponent for different erasure rates ϵ. Fig. 3.12 shows the asymptotic performance as a function of the delay for the first information block. It is clear from Fig. 3.12 that the slopes of the decoding erasure probability obtained with P-EXIT analysis match perfectly with the theoretically determined anytime exponents $\alpha_o = -\log_2(\epsilon)$. The decoder erasure probabilities of any other block are identical to the performance of the first block, confirming (3.35) that the decoder erasure performances for different information variable nodes are just shifted versions of each other.

3.2.2 Finite-Length Behavior

We now proceed to analyze the finite-length behavior of codes derived from the proposed protograph structure. All the numerical simulation results presented in this section are for the finite-length ensemble of codes derived from the protograph base matrix detailed in (3.3). The lifting of the protograph is done using different random permutation matrices for each simulation run, and for each relevant block in the base matrix. A new collection of random permutation matrices are generated for each subsequent simulation run. Note that no attempt has been made to remove small cycles. We can expect the performance to improve by removing small cycles but cycles are inevitable in the structure presented in (3.4). Their negative effects are therefore always encountered even if we remove small cycles. To get a clear understanding of the finite-length behavior independent of any cycle-removal procedures we therefore do not remove cycles at all. We propose the decoder detailed in Algorithm 3.1 for transmission over the BEC. This algorithm can be implemented

Algorithm 3.1 Decoder for the BEC

1: At time t append the newly received channel outputs $\tilde{\boldsymbol{y}}_t$ to the vector $\tilde{\boldsymbol{y}}_{[1,t-1]} = [\tilde{\boldsymbol{y}}_1 ... \tilde{\boldsymbol{y}}_{t-1}]$ of previously received channel outputs.
2: Replace received bits in $\tilde{\boldsymbol{y}}_{[1,t-1]}$ that are known from previous decoding attempts.
3: Determine the index i_{eue} of the code block containing the earliest undecodable erasure (eue).
4: Expand the parity-check matrix so that it contains parity-check equations covering all code blocks from the code block containing the earliest uncorrected error to the newly received block.
5: Run the message-passing algorithm on the vector $\tilde{\boldsymbol{y}}_{[i_{eue}-1,t]}$ and the expanded parity-check matrix $\boldsymbol{H}_{i_{eue}-1,t}$. This results in the estimate $\hat{\boldsymbol{y}}_{i_{eue}-1,t}$ from which we can obtain the systematic parts $\hat{\boldsymbol{y}}^s_{[i_{eue}-1,t]}$.
6: Update the overall received information and parity vector $\hat{\boldsymbol{x}}_{[1,t]} = \hat{\boldsymbol{y}}^s_{[1,t]}$ and $\hat{\boldsymbol{y}}^c_{[1,t]}$.

very efficiently using parallel and distributed techniques. The decoding algorithm can even be modified for other channels.

Cycles and their Impact on the Performance

For finite-length anytime codes derived from (3.3) cycles are a substantial problem when transmitting over the BEC. Fig. 3.13 illustrates the problem. Here, the ensemble average decoding performance $P^i_{e,[1,t]}$ is shown for blocks $i = 1 \dots 40$ when transmitting over a BEC with $\epsilon = 1/3$ using a code obtained by lifting the base matrix in (3.3) with $k = 3$. The different curves correspond from left to right to the average performance of information blocks $\boldsymbol{x}_1$ to $\boldsymbol{x}_{40}$ as a function of the decoding delay. The behavior is averaged over 10^6 simulation runs. We can see that as

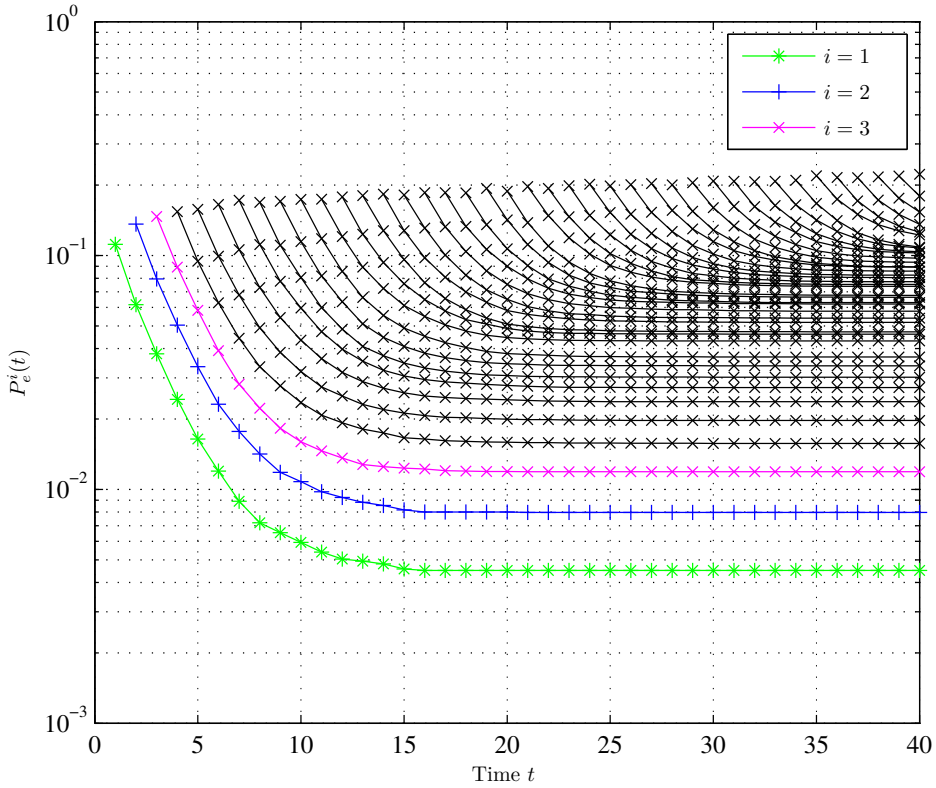


Figure 3.13: $P_e^i(t)$ of the 40 first blocks of the code obtained by lifting (3.3) with $k = 3$. The curves corresponds from left to right to the performance of information blocks $\boldsymbol{x}_1..\boldsymbol{x}_{40}$ from the time $t = i$ that they are decodable for the first time. Transmission over the BEC with $\epsilon = 1/3$.

opposed to the asymptotic ensemble behavior the ensemble average finite-length behavior of a particular block is not independent of its position. Moreover an erasure floor exists. We observe that the average decoding capability decreases over time and for even larger t than shown here it eventually converges to the performance obtained when transmitting uncoded bits. We deduce from Fig. 3.13 that there exist erasure patterns that cause the decoder to fail over all times. Fig. 3.14 shows the averaged finite-length performance for block $i = 20$ for codes derived from the proposed protograph when lifting with different lifting factors k together with the asymptotic performance obtained by P-EXIT analysis. We see that the error floor is decreasing with increasing k and the asymptotic performance is approached. We can say that for large k the code has anytime behavior with high probability. Note however that we cannot know from the simulations whether or

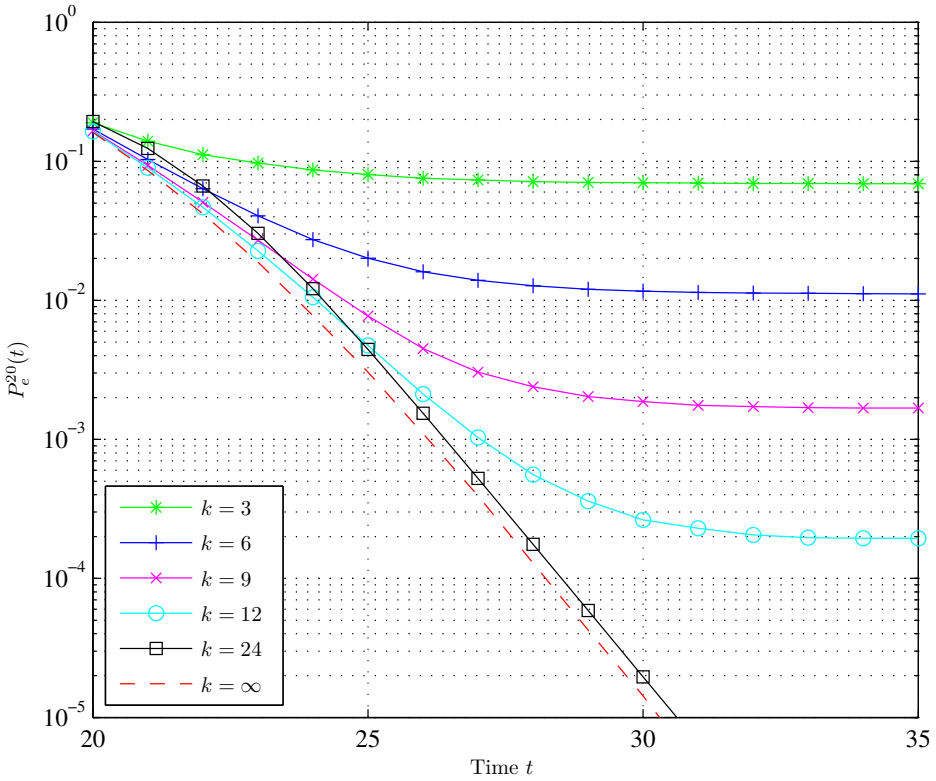


Figure 3.14: Performance $P_e^{20}(t)$ of the 20th block. The codes are obtained by lifting (3.3) with different lifting factors $k = 3, 6, 9, 12, 24$. Transmission over the BEC with $\epsilon = 1/3$.

not an error floor occurs since it might be too low to be apparent in the simulations. Since the experienced error floor is increasing with block index it is important in the context of streaming to have a clear understanding about if and why an erasure floor occurs. In the following we analyze erasure patterns that cause the decoder to fail over all times. We call these patterns *increasing erasure patterns* and *increasing error patterns*, the event is called *growing error event*. The protograph is designed to permit the decoding of a continuously streaming bit stream, where reliability increases with decoding delay. Therefore, for a specific realization of a parity-check matrix and a received stream of bits there is a high probability for undecodable erasures at any particular time instance. Consider the set of undecodable erasures at time t. With the arrival of the next code block, and another decoding attempt at time $t+1$, the set of undecodable erasures typically changes. For example, with the new parity-check equations some erasures may now be decodable and are therefore

no longer part of the set, or erasures in the newly received code block may enter the set. A problem emerges if erasures in the newly received code block enter the set of undecodable erasures but no earlier erasures get resolved, and thus no erasures leave this set. If this happens repeatedly over a number of time steps, it becomes less and less likely for the decoder to be able to decode any erasures due to the lower diagonal structure of the protograph. In the following we focus on erased information bits $\tilde{\boldsymbol{y}}^s_{[1,t]}$ within the set of undecodable erasures. Denote by s_t the size of the erasure pattern that the decoder fails to decode at time t; namely the number of erased information bits in the set. In order for an increasing erasure pattern to emerge the set of undecodable erasures has to grow on average in each time step, due to the way the protograph is expanded. The set of undecodable erasures is related to stopping sets. A stopping set $\mathcal{S}$ is a subset of variable nodes in a Tanner graph for which every check node connected a variable node in $\mathcal{S}$ is connected to at least one more variable node in $\mathcal{S}$. Therefore if all variable nodes in S are erased it is not possible for the message-passing decoder to recover the variable nodes since every check equation that operates on the variable nodes in $\mathcal{S}$ involves more than one erased bit. The set of stopping sets therefore determines the erasure patterns that cause the decoder to fail. In our context, however, we cannot use the results of stopping set analysis since the parity-check matrix is growing over time and it is the development of only certain erasure patterns over time that is of interest.

3.2.3 Finite-Length Analysis

The following model is used in order to analyze the finite-length behavior. In place of a general BEC we assume a modified BEC. The modified BEC affects the transmitted bit stream such that the fraction of erasures per block is *exactly* equal to the erasure rate ϵ. That is, the number E of erasures randomly introduced by the channel into an information or parity block of k bits is the same for all blocks and always equal to ϵk, where we choose k or ϵ such that E is an integer. We refer to the modified channel as the *static* BEC, and write the corresponding erasure rate with an index s: ϵ_s. Note that for the finite-length analysis we have to take into account the structure of the lifted graph as well as the protograph. We assume the protograph is lifted with random permutation matrices and perform no attempt to remove small cycles. We particularly focus on worst-case erasure patterns where the decoder fails in decoding *all* erasures within information blocks from t_0 in time onwards and we want to know how likely such erasure patterns are. If all bits are decodable up to time t_0 then for the analysis we can equivalently assume that $t_0 = 0$. For the considered protograph the worst-case decoding scenario when transmitting over the static BEC with erasure rate $\epsilon_s = E/k$ implies that the erasure pattern from t_0 onwards consists of *all* erased information bits from t_0 up to the current time and an additional E new erasures for every new received information block $\tilde{\boldsymbol{y}}^s_t$ per time step t. We now want to determine the probability $P(t)$ of the occurrence of an erasure event containing exactly $s_t = Et$ unknown information bits at time t given a randomly lifted protograph and a static BEC with erasure rate ϵ_s. Hence, $P(t)$ is

the probability of the worst-case erasure event to happen. This is done by finding erasure patterns that the decoder fails to decode through random permutations of E erasures per information block for a randomly chosen parity-check matrix from the ensemble. We make use of the following probabilities:

- $P(t) := \Pr\{s_t = Et\}$ is the probability of an erasure pattern containing exactly Et unknown information bits in the stream $\hat{\boldsymbol{x}}_{[1,t]}$ up to time t. Due to the nature of the static BEC this can only be the case if an erasure pattern was formed from $t = 0$ onwards, where there are exactly E erasures in each information block $\tilde{\boldsymbol{x}}_i$, which furthermore cannot be recovered at any decoding attempt up to time t.
- $P(t|t-1) := \Pr\{s_t = Et | s_{t-1} = E(t-1)\}$ is the probability that an erasure pattern containing $s_{t-1} = E(t-1)$ undecodable information bits at time $t-1$ is expanded to contain $s_t = Et$ undecodable information bits at time t. It follows that going from time $t-1$ to time t the worst-case erasure pattern propagates since it contains again all erasures within all information blocks transmitted so far.

The probability $P(t)$ can be written recursively: $P(t) = P(t|t-1)P(t-1)$. In every time step t a new check node and two new variable nodes are appended to the overall matrix $\boldsymbol{B}_{[1,t]}$, due to the protograph structure given in (3.3). The remaining protograph does not change. A new check node in the protograph translates to k new check nodes in the lifted graph. The probability $P(t|t-1)$ that a worst-case erasure pattern given at time $t-1$ is expanded by the E erased information bits within the $t-$th information block depends on these k new check nodes introduced at time t only. If none of these k check equations can be solved for any of the undecodable information bits then the size of the erasure event increases by E and we have found a worst-case erasure pattern for time t. If we only want to determine $P(t|t-1)$ we can therefore restrict our attention to these k new check nodes. Denote the k new check nodes by $C_t^1 \dots C_t^k$. Let n_t^l denote the number of erased information bits connected to check node C_t^l and let m_t^l be the value of the connected parity bit from $\tilde{\boldsymbol{y}}_t^c$. Then we know that check node C_t^l can be used to recover an erased information bit if $n_t^l = 1$ and $m_t^l = 0$. If at time t none of the tuples (n_t^l, m_t^l) with $l = 1, 2, \dots, k$ is equal to $(1, 0)$ then the k check nodes cannot be used to recover any erased information bit. The structure of the proposed protograph imposes that at time t a check node C_t^i is connected to t information bits where each of the t information bits is from a different information block $\tilde{\boldsymbol{x}}_i$ where $i = 1, 2, \dots, t$. The random lifting imposes that the connected information bit within each of the different information blocks $\tilde{\boldsymbol{x}}_i$ is chosen in a random way. For example, if we consider the structure given in (3.3) the connections of information bits within each of the blocks $\tilde{\boldsymbol{x}}_i$ with $i = 1, 2, \dots, t$ to check nodes $C_t^1 \dots C_t^k$ is obtained by a random permutation. Now consider in this framework an erasure pattern with $s_{t-1} = E(t-1)$; that is, the pattern contains all E erasures from each of the information blocks $\tilde{\boldsymbol{x}}_i$, with $i = 1, 2, \dots, t-1$. We now investigate how

such an erasure pattern propagates if we go from $t-1$ to t. The newly received information and parity blocks at time t contain exactly E erasures each due to the static BEC . Moreover if $s_{t-1} = E(t-1)$ then there are exactly E erasures in each of the information blocks $\tilde{\boldsymbol{x}}_i$ where $i = 1, 2, \ldots, t-1$. It follows that if at time t we consider the edges of a single check node C_t^l, then each of its t randomly chosen connections to information blocks has a probability $\epsilon_s = E/k$ of being a connection to an erasure. Therefore at time t the variable n_t^l is distributed according to the binomial distribution $\mathcal{B}(\epsilon_s, t)$:

$$\Pr\{n_t^l = 1 | s_{t-1} = E(t-1)\} = t\epsilon_s \left(1-\epsilon_s\right)^{t-1}. \tag{3.38}$$

Note however that if we consider more than a single check node C_t^l the variables n_t^l with $l = 1 \ldots k$ are not independent due to the connections being chosen by permutations. We can now see that the erasure pattern with $s_{t-1} = E(t-1)$ increases in size if none of the check nodes $C_t^1 \ldots C_t^k$ is connected to a single erased information bit only. The probability $P(t|t-1)$ can therefore be written as

$$P(t|t-1) = \Pr\{n_t^1, ..., n_t^{k-E} \neq 1 | s_{t-1} = E(t-1)\}, \tag{3.39}$$

where we reordered the check nodes $C_t^1, C_t^1, \ldots, C_t^k$ such that the check nodes $l = 1, 2, \ldots, (k-E)$ correspond to the check nodes where $m_t^l = 0$. Finding the exact probability $P(t)$ is difficult since it involves finding all erasure patterns and the corresponding probabilities that cause the decoder to fail decoding any of the erasures. However we can approximate the probability $P(t)$ by approximating $P(t|t-1)$, which is obtained by assuming that the variables n_t^l *are* independent. Under the assumption of independence we can write

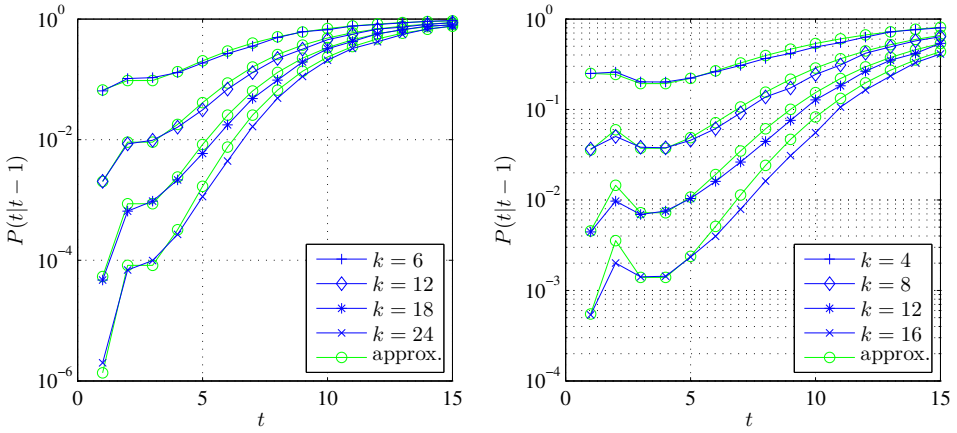
$$\begin{aligned} P(t|t-1) &= \Pr\{n_t^1, .., n_t^{k-E} \neq 1 | s_{t-1} = E(t-1)\} && (3.40)\\ &= \prod_{l=1}^{k-E} \Pr\{n_t^l \neq 1 | n_t^{l-1}, ..., n_t^1 \neq 1, s_{t-1} = E(t-1)\} \\ &\approx \prod_{l=1}^{k-E} \Pr\{n_t^l \neq 1 | s_{t-1} = E(t-1)\}, && (3.41) \end{aligned}$$

where $\Pr\{n_t^l \neq 1 | s_{t-1} = E(t-1)\}$ is given as

$$\begin{aligned} \Pr\{n_t^l \neq 1 | s_{t-1} = E(t-1)\} &= 1 - \Pr\{n_t^l = 1 | s_{t-1} = E(t-1)\} \\ &= 1 - t\epsilon_s \left(1-\epsilon_s\right)^{t-1}, \end{aligned} \tag{3.42}$$

where we used (3.38). For increasing t the probability $\Pr\{n_t^l = 1\}$ converges to 0. Assuming independence, $P(t|t-1)$ can then be approximated as

$$P(t|t-1) \approx \left(1 - t\epsilon_s \left(1-\epsilon_s\right)^{t-1}\right)^{k-E}. \tag{3.43}$$



(a) $P(t|t-1)$ for $\epsilon_s = 1/3$.

(b) $P(t|t-1)$ for $\epsilon_s = 1/4$.

Figure 3.15: Comparison between simulated and approximated $P(t|t-1)$ for different k and different ϵ_s.

Fig. 3.15 shows the simulated and approximated probability $P(t|t-1)$ for different k when transmitting over a static BEC with different erasure rates ϵ_s. We note that the predicted and simulated behavior is very close. The approximation is good in general if the assumption of independence is justified. This is approximatively the case if either $Et \ll k$ but $Et > k/2$ or if $Et \gg k$. From Fig. 3.15 we see as well that $P(t|t-1)$ converges rapidly to 1. However it is clear from the figures that the probability for an erasure event to start propagating is significantly lower for larger k. With the above approximation we can approximate the probability $P(t)$ of a worst-case erasure event containing exactly $s_t = Et$ erased information bits and stretching over t time instances:

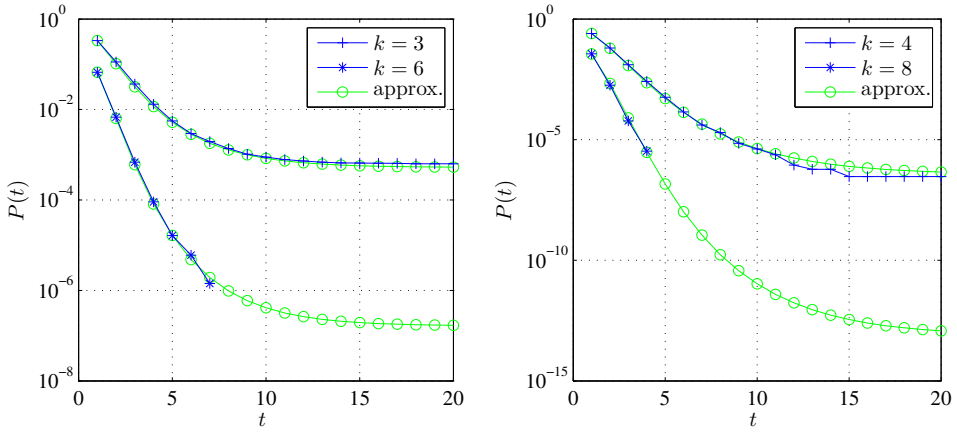
$$P(t) = P(t|t-1)P(t-1) = P(t=1)\prod_{i=2}^{t} P(i|i-1) \tag{3.44}$$

$$\approx P(t=1)\left[\prod_{i=2}^{t}\left(1 - i\epsilon_s\left(1-\epsilon_s\right)^{i-1}\right)\right]^{k(1-\epsilon_s)} \tag{3.45}$$

$$\text{where} \quad P(t=1) = \frac{(\epsilon_s k)!(k(1-\epsilon_s))!}{k!}$$

$$\approx \sqrt{2\pi\epsilon_s k(1-\epsilon_s)}\left[1-\epsilon_s\right]^{k(1-\epsilon_s)}\epsilon_s^{\epsilon_s k}, \tag{3.46}$$

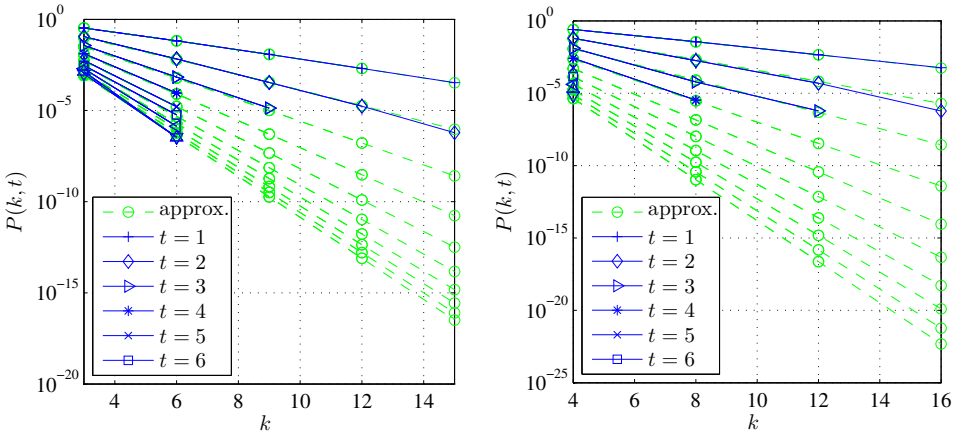
by using Stirling's approximation of the factorial. Here we calculated $P(t=1)$ exactly through combinatorics. From the above we see that the asymptote of the probability $P(t)$ decays exponentially with k. Since $P(t|t-1)$ converges over time t, so does the probability $P(t)$.



(a) $P(t)$ for $k = 3$ and $k = 6$ with $\epsilon_s = 1/3$. (b) $P(t)$ for $k = 4$ and $k = 8$ with $\epsilon_s = 1/4$.

Figure 3.16: Simulation and approximation of the probability $P(t)$ that an erasure pattern contains exactly $k\epsilon_s t$ unknowns for different k and ϵ_s.

Fig. 3.16 shows the probability $P(t)$ as a function of t for different block lengths k. We can verify that the approximation is very close to the simulation results. Moreover we observe the saturation of $P(t)$ with t.



(a) BEC with $\epsilon_s = 1/3$. (b) BEC with $\epsilon_s = 1/4$.

Figure 3.17: $P(t)$ for transmission over a BEC with different static erasure rates.

Figs. 3.17a and 3.17b show the behavior of the probability $P(t)$ with k for different $s_t = Et$. We observe the exponential decay with k. Since $P(t)$ is very

small for increasing k there are no simulation results over the entire range. However for the obtained results the approximation is very close. We conclude from the approximation that for all k and $\epsilon_s \geq 1/k$ there exist erasure patterns causing the decoder to fail over all times. However the probability can be made arbitrarily small by increasing k.

3.2.4 Turning Point

From the finite-length analysis described in the previous paragraph we can find an important parameter. We note in Fig. 3.15 that from some time t onwards the probability $P(t|t-1)$ is monotonically increasing to 1. Moreover we know that $P(t)$ flattens out. Since $P(t) = P(t-1)P(t|t-1)$, a turning point t^* of the probability $P(t)$ is, when the probability $P(t|t-1)$ that the erasure pattern increases is larger than the probability that it decreases or stays constant. From the analysis we cannot solve for t^* explicitly but we can derive the following inequality for t

$$P(t|t-1) \leq 0.5 \tag{3.47}$$

$$\left[1 - t\epsilon_s \left(1-\epsilon_s\right)^{t-1}\right]^{k(1-\epsilon_s)} \leq 0.5 \tag{3.48}$$

$$1 - t\epsilon_s \left(1-\epsilon_s\right)^{t-1} \leq 0.5^{\frac{1}{k(1-\epsilon_s)}}. \tag{3.49}$$

The turning point is the maximum value t^* of t where $P(t|t-1) \leq 0.5$. This is an important point that we make frequently use of in Chapter 4 when we investigate modified code structures taking practical constraints into account.

3.2.5 Transmission Over the Standard BEC

So far we have only considered finite block length transmission over the so-called static BEC which introduced exactly ϵk erasures in a block of k bits. The motivation for this modification of the standard BEC was analytical tractability. It is difficult to do the analysis for the general case of a standard BEC. However we can observe through simulations that the results obtained by the analysis of the static BEC can be used for the design of codes for the standard BEC as well. We use the probability $P(\omega)$ that the decoding window has a certain size ω to demonstrate the similarity. From the decoding algorithm described in Algorithm 3.1 we know that the decoding window contains all blocks from $t = i_{eue} - 1$ onwards. The decoding window size is therefore $\omega = t - (i_{eue} - 1)$. In Fig. 3.18 we compare the decoding window size obtained through simulations between transmission over the static and the standard BEC. Moreover we show the theoretically determined probability based on P_e^i obtained through the asymptotic analysis by noting that the decoding window is of size ω only if all k bits in $\hat{\boldsymbol{x}}_i$ are known for all messages with indices $i > \omega + 1$, that is,

$$P(\omega) = \prod_{i=\omega+1}^{\infty} (1 - P_e^i)^k. \tag{3.50}$$

We can see that the probability of a decoding window of size ω is very similar

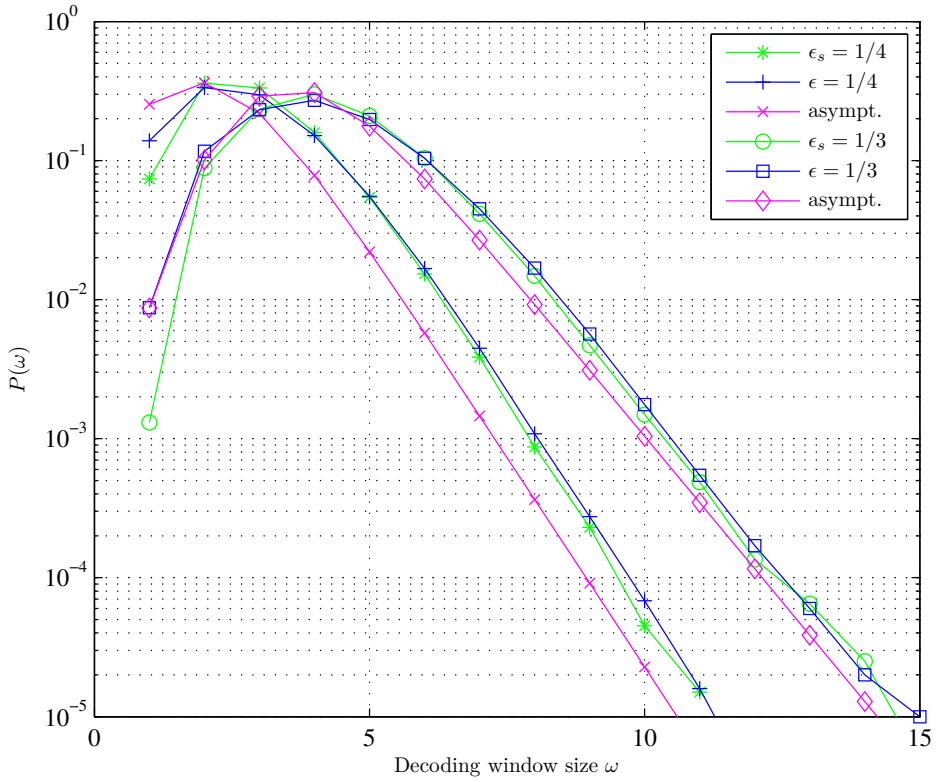


Figure 3.18: Comparison of the decoding window size probability $P(\omega)$ for the static BEC and the conventional BEC together with a calculated value based on the P-EXIT analysis. Here $\epsilon = \epsilon_s = 1/4$ and $k = 20$ and $\epsilon = \epsilon_s = 1/3$ and $k = 24$.

for both the standard and the static BEC. Moreover the theoretically determined probability based on the asymptotic performance is a good approximation already for moderate block lengths k. From (3.50) we can see that for $k \to \infty$ the probability $P(\omega)$ decays exponentially with the operational anytime exponent $\alpha_o = -\log_2(\epsilon)$.

Operating Range

From the finite-length analysis of the anytime codes, e.g. (3.46) and (3.43), we can see that the erasure probability ϵ_s has a strong influence on the performance of the codes. For high erasure rates the probability $P(t|t-1)$ that an erasure event increases in size is quickly converging to 1. Operating the anytime codes at high erasure rates is therefore not reasonable. In general without feedback as described

in a later chapter the probability that the decoder encounters an increasing error pattern is never equal to 0 (assuming that $\epsilon_s > 0$). But by increasing the block length k we can lower the probability of the occurrence of such a pattern. This is reasonable for erasure rates that are in a certain range. Fig. 3.19 shows the reason. Here we depict the calculated probability $P(t|t-1)$ for different erasure rates ϵ_s for $k = 12$. We can see that after $\epsilon_s = 1/4$ the curves do not increase immediately to 1 but first decrease for a short moment. This drop of the probability is necessary for a good performance since it indicates that whereas a small number of erasures is always tolerated, an increasing number of erasures should be prevented as much as possible by the code. For $k = 12$ a reasonable operation region is therefore $\epsilon_s < 1/4$.

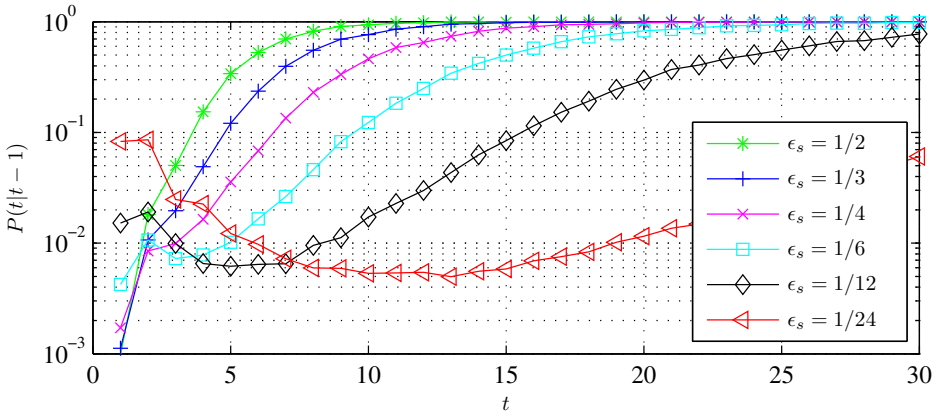


Figure 3.19: $P(t|t-1)$ for different erasure rates ϵ_s and $k = 12$.

Fig. 3.20 confirms this. Here we show the simulated decoding error probability for messages $\boldsymbol{x}_{40}, \boldsymbol{x}_{42}$ when transmitting over the (conventional) BEC with different erasure rates ϵ. For $\epsilon = 0.5$ the decoding error probability is the same for $\boldsymbol{x}_{40}$ and $\boldsymbol{x}_{42}$ (only $\boldsymbol{x}_{42}$ is shown in the figure) since the erasure floor for blocks with large indices is increasing to the level where the code has lost its error protection capabilities. For $\epsilon = 1/3$ the curves decay to a relative high error floor. And for $\epsilon = 1/4$ the erasure floor is lower than 10^{-6}. We conclude that an estimate of the proper operating region can be obtained from the calculations of $P(t|t-1)$. Although the estimate is obtained from an analysis for the static BEC, it is useful when transmitting over the conventional BEC.

3.3 Performance on the AWGN Channel

In this section we briefly summarize the results by Tarable et al. [TNDT13a, TNDT13b] on extending the results obtained for the codes developed in the previ-

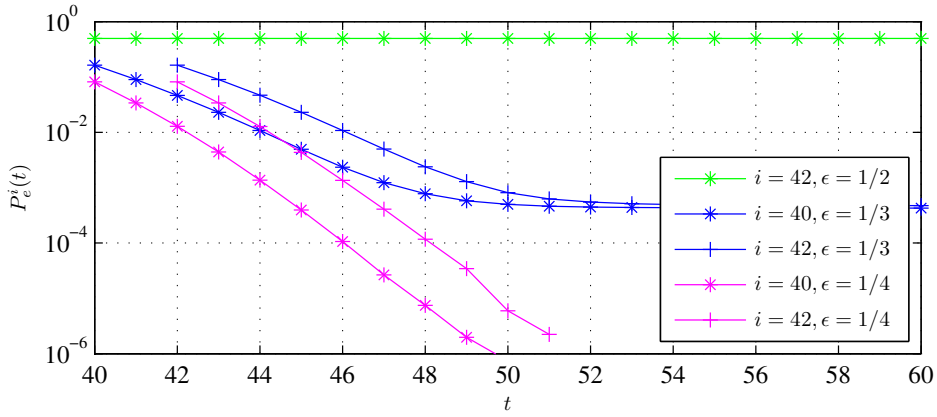


Figure 3.20: $P_e^i(t)$ for messages $\boldsymbol{x}_{40}, \boldsymbol{x}_{42}$ for different erasure rates ϵ and $k = 12$.

ous sections to the case where transmission takes place over the AWGN channel. The authors in [TNDT13a, TNDT13b] prove that the systematic LDPC-CCs are anytime reliable and give a lower bound on their anytime exponent. Due to their choice of simulation parameters the authors do however not see a finite-length behavior that differs significantly from the asymptotic behavior. We show through simulations that when transmitting over the AWGN channel a very similar problem as when transmitting over the BEC occurs: The decoder can end up in a state where it fails to decode an increasing number of bits. Note the following differences between transmitting over the BEC and the AWGN channel: The decoder when transmitting over the AWGN channel is very similar to the one described in Algorithm 3.1 however since the decoder has no knowledge about whether or not a bit is correct or not, it is not possible to determine an equivalent to the index of the earliest undecodable erasure i_{eue}. As a consequence there is no easy way to limit the decoding window but instead it is growing over all times. The decoder for the AWGN channel has therefore the same description as given in Algorithm 3.1 except that i_{eue} is always equal to 0. Note that when transmitting over the AWGN channel, it is possible that bits that have been decoded correctly once might be decoded incorrectly at a later point in time. This is not possible when transmitting over the BEC.

3.3.1 Asymptotic Performance Analysis

Tarable et al. extended in [TNDT13b] the P-EXIT analysis for transmission over the BEC derived in Section 2.4.5 to the case where transmission takes place over the AWGN channel. The extension is given below. Denote by $\rho_{CV,t}^{\ell}(i,j)$ the SNR at time t for a message travelling from check node i to variable node j at the ℓ-th iteration. Similarly, denote by $\rho_{VC,t}^{\ell}(i,j)$ the SNR at time t for a message travelling

from variable node i to check node j at the ℓ-th iteration. The physical-channel SNR for a variable node is denoted by ρ_{ch}. Furthermore let $\mathcal{N}_c(i)$ denote the set of variable nodes that are neighbors of the i-th check node and let $\mathcal{N}_v(j)$ denote the set of check nodes that are neighbors of the j-th variable node. The SNR evolution at time t at positions (i,j) where $B_t(i,j)=0$ is equal to 0, at all other positions (i,j) it can be approximately determined through the following set of update equations:

- **Initialization:** $\rho^0_{VC,t}(i,j) = \rho_{ch}$ (3.51)
- **Check node i to variable node j:**

$$\rho^{\ell+1}_{CV,t}(i,j) \approx M\left(\sum_{s\in\mathcal{N}_c(j),s\neq j} M\left(\rho^{\ell}_{VC,t}(i,s)\right)\right) \tag{3.52}$$

- **Variable node j to check node i:**

$$\rho^{\ell}_{VC,t}(i,j) = \sum_{s\in\mathcal{N}_v(j),s\neq i} \rho^{\ell+1}_{CV,t}(s,j) + \rho_{ch} \tag{3.53}$$

- **Output decision variable SNR**:

$$\rho^{\ell+1}_{t}(j) = \sum_{s\in\mathcal{N}_v(j)} I^{\ell+1}_{Av,t}(s,j) + \rho_{ch}. \tag{3.54}$$

The function $M(\rho)$ appearing above is defined as

$$M(\rho) = J^{-1}(1-J(\rho)), \tag{3.55}$$

where the relation between the mutual information $J(\rho)$ between the input of a binary-input AWGN channel with SNR ρ and the corresponding output is given as

$$J(\rho) = 1 - \int_{-\infty}^{+\infty} \frac{e^{-(y-2\rho)^2/(8\rho)}}{\sqrt{8\pi\rho}} \log_2(1+e^{-y})dy. \tag{3.56}$$

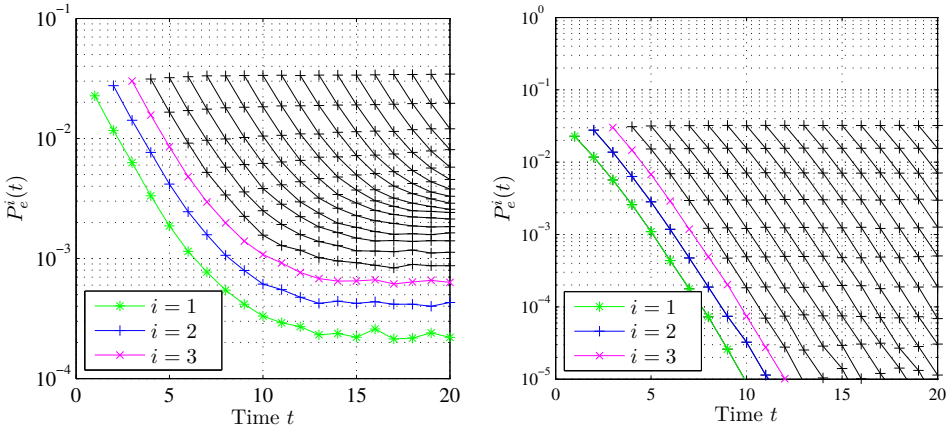
The approximation involved in determining $\rho^{\ell+1}_{CV,t}(i,j)$ above is according to [tBKA04] tight in most cases of interest. As for the BEC, the sequences $\rho^{\ell}_{VC,t}(i,j)$ and $\rho^{\ell}_{CV,t}(i,j)$ are monotonically increasing with iteration index ℓ [TNDT13b] and the output decision variable SNR converges to a limit $\rho^{\infty}_{t}(j)$. An upper bound on $\rho^{\infty}_{t}(j)$ is derived as

$$\rho^{\infty}_{t}(j) \leq (|N_v(j)+1|)\rho_{ch}. \tag{3.57}$$

The authors show that this limit is approximately achieved for variable nodes j with a sufficient delay if $\rho_{ch}\to\infty$. Since the number $|N_v(j)|$ of connected check nodes of variable node j increases linearly over t and decreases linearly over j in the protograph we can see that even the output SNR on average increases linearly with decreasing index and increasing time. This is a useful property that we make use of in Chapter 5 when we propose an algorithm to detect increasing error patterns.

3.3.2 Finite-Length Behavior

Due to the choice of block length and SNR ($k = 12$, SNR $= 3.5$, 4.5, 5 dB) the authors in [TNDT13a] observe a finite-length behavior similar to the asymptotic behavior. In particular no error floors are observed. When lowering the block length k this is different. In Fig. 3.21a we show the finite-length behavior over time when simulating the transmission with message lengths of $k = 6$ over an AWGN channel with SNR $= 3$ dB. Comparing with Fig. 3.13 we see a very similar behavior as for the BEC: The decoding error probability does not decay exponentially over time but reaches an error floor. And moreover the error floor increases for increasing block indices. From the previous discussion about the behavior of the codes when transmitting over the BEC channels we know that an erasure floor that is increasing over the block index indicates that there are erasure patterns that the decoder fails to decode correctly over all times. We therefore conclude that when transmitting over the AWGN channel the problem of increasing erasure patterns as observed for the BEC translates into a problem of increasing error patterns. Fig. 3.21b shows



(a) $P_e^i(t)$ for SNR $= 3$ dB and $k = 6$.

(b) $P_e^i(t)$ for SNR $= 3$ dB and $k = 24$.

Figure 3.21: Simulated decoding error probability $P_e^i(t)$ for indices $i = 1, \ldots, 20$ over time for different message lengths k.

the simulated finite-length behavior when transmitting over an AWGN channel with SNR $= 3$ dB using the proposed anytime code with message length $k = 24$. We can see that the performance over time is improved by increasing the block length. This is analogue to the observations made for the BEC. Similarly the performance can be improved by increasing the SNR.

Finally, we note that due to the nature of the AWGN channel, it is not possible to perform a finite-length analysis as for the BEC.

3.3.3 Conclusion

We conclude that the anytime codes developed in the previous section are suitable for transmission over the AWGN channel as well. Even though a finite-length analysis as for the BEC is not possible to perform, the knowledge gained from the analysis of the codes over the BEC allows us to explain the finite-length behavior observed when transmitting over the AWGN channel using short block lengths.

3.4 Comparison With Other Anytime Codes

In this section we describe two other anytime codes proposed in the literature. We compare their structure to the ones proposed in this thesis and outline their main differences.

3.4.1 Comparison with Toeplitz Codes

In [SH11b] the authors consider time-invariant codes with Toeplitz generator and parity-check matrices called *Toeplitz ensemble*. Together with the codes they propose an efficient Maximum-Likelihood decoding algorithm when transmitting over the BEC. The details of the code construction can be found in Appendix A.1. Toeplitz codes are structurally similar to the codes proposed in this thesis in the way that the parity-check matrix is composed of several smaller matrices. Moreover the parity-check matrix has Toeplitz structure as it is the case for our codes. A major difference is however that the parity-check matrix is typically dense, even though we can reduce the density by decreasing the parameter p in the code construction of the Toeplitz ensemble. This is however not desirable for the decoder. Due to the density of the parity-check matrix message-passing decoding is not suitable for Toeplitz codes. Since message-passing algorithms cannot be exploited but an efficient decoding algorithm is only proposed when transmitting over the BEC, there is no obvious way of extending the decoding algorithm to decoding codewords transmitted over other channels than the BEC. Due to these reasons we do not compare our codes to these anytime codes.

3.4.2 Comparison with Spatially Coupled Anytime Codes

In [NARNL15, NARNL13] another approach is taken to the development of practical anytime codes. Here a class of spatially coupled codes (termed SCLDPC anytime codes) are investigated and shown to have anytime properties asymptotically. The details of the code construction, the asymptotic analysis and finite-length estimation can be found in Appendix A.2.

Differences in the Code Structure

SCLDPC codes are just as the codes proposed in this thesis based on protographs. An important difference is however that the protograph in the construction of

SCLDPC codes is random. Whether a variable node at position i is connected to a check node at position j is for SCLDPC codes determined by a probability $P_r(d, \lambda)$, where $d = i - j$ is the distance between the nodes and λ is a design parameter. The probability is chosen to be exponentially decaying with d. Another important structural difference between the codes proposed in this thesis and SCLDPC anytime codes is that for our codes the variable and check node degrees are increasing whereas the degrees of the SCLDPC codes are constant equal to d_v, d_c. For SCLDPC codes as presented in [NARNL15] no encoder is given whereas for the codes presented in this thesis there exists a simple encoder. The anytime exponent of SCLDPC anytime codes is derived to be equal to $\alpha_{SCLDPC} = \lambda d_v$. This is independent of the channel erasure rate. The anytime exponent of the codes proposed in this thesis improves if the channel quality is improved.

Differences in the Interpretation of the Simulations

In this thesis we use the probability of decoding error over time/delay as the performance measure of the codes. When simulating the performance of our codes the protograph is therefore constant for all simulations but the lifting is done using different randomly picked permutation matrices in every new simulation run. The performance curves obtained in this way give a good idea on how a particular code behaves on average over delay. When simulating the performance of the SCLDPC codes as it is done in [NARNL15] apart from the different random lifting, the protograph *itself* is randomly picked according to the parameters of the code. That is, the entire protograph is different for every simulation run. When simulating the performance in this way we have to be very careful when interpreting the results from the averaged simulations since we loose one aspect considering the behavior over delay: A particular choice of a (d_v, d_c) SCLDPC code with parameters λ and γ results in a protograph with fixed connections between variable and check nodes. The number d_v of check nodes connected to a variable node is typically small (in the simulations $d_v = 3$). As an example, some variable node i has connections to check nodes j_1, j_2, j_3 chosen according to the exponential distribution (A.6) ($\gamma = \infty, \lambda = 0.1$) resulting in distances $d_1 = (i - j_1) = 5$, $d_2 = (i - j_2) = 6$ and $d_3 = (i - j_3) = 7$. Then *per construction* the message corresponding to variable node i cannot be decoded until a minimum delay of $d = d_1 = 5$ time steps has passed. And this is independent of the lifting of the code and the channel erasures introduced. The performance curve averaged over many different protographs however shows that this variable node is decodable (although with a high probability of error) already after 1 time step. In the context of a control scenario these fixed delays resulting from the construction of the protograph can have a severe impact on the performance. Note as well that due to the probabilistic choice of the connections and with $\gamma = \infty$ the fixed delay for one variable node can possibly be very large. The difference becomes even more clear when comparing the decoding error performance of the averaged ensemble and the decoding error performance of a particular code. Fig. 3.22 shows the performance averaged over 10000 sim-

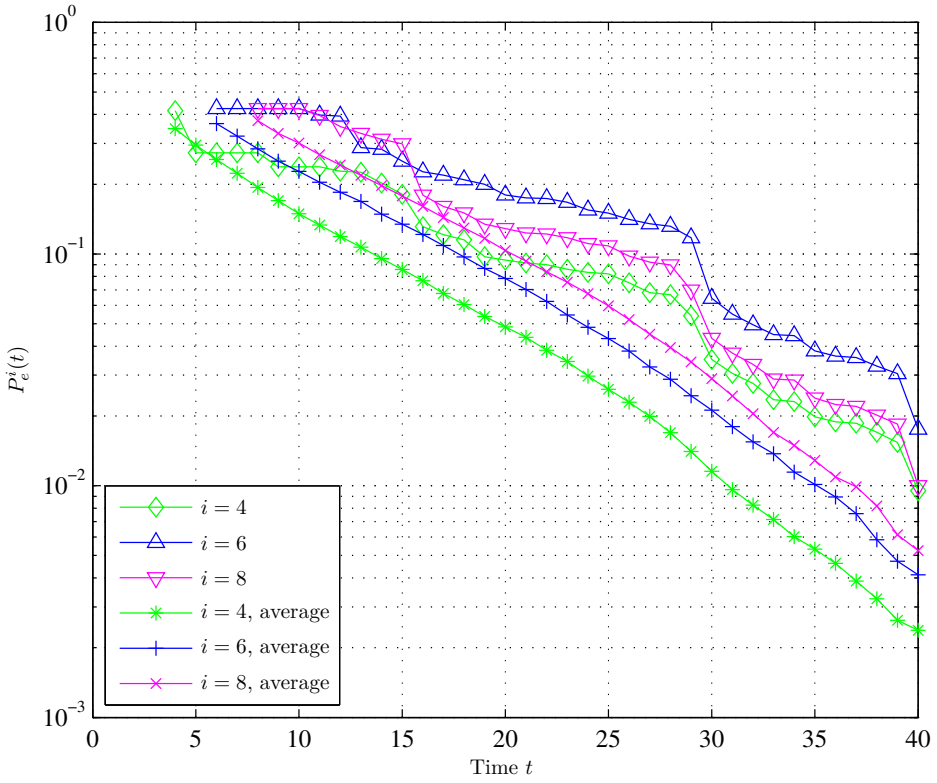


Figure 3.22: Decoding error probability $P_e^i(t)$ when using a $(3, 6)$-SCLDPC anytime code with $k = 40, \lambda = 0.1, \gamma = \infty, \epsilon = 0.42$. Comparison of average behavior of the ensemble and a particular code.

ulation runs of one particular protograph of a $(3, 6)$-SCLDPC anytime code with $\lambda = 0.1, \gamma = \infty, k = 40$ when transmitting over a BEC with $\epsilon = 0.42$. (Note that the lifting is still different in every simulation run). The simulated decoding error probability $P_e^i(t)$ over time is shown for messages $\boldsymbol{x}_4, \boldsymbol{x}_6$ and $\boldsymbol{x}_8$. The lifting of the protograph is different in every simulation run as well as the channel realization, but the protograph is the same. In the same figure we show the performance obtained when averaging over many different protographs of the ensemble as typically done in [NARNL15, ZNARVN15b, NARNL13]. We can see that the probability of decoding error decays for the particular choice of the code similar to the one of the ensemble. However the ensemble average has to be interpreted with caution. For example from the ensemble average we read that on average after a delay of 12 time steps the probability of decoding error of message $\boldsymbol{x}_6$ is lower than 0.1. For the particular code however on average a probability of decoding error below 0.1

can only be obtained for message $\boldsymbol{x}_6$ with a delay of 24 time steps. While such a difference in the delay seems a bit unimportant here, we will see in Chapter 6 when using the codes in a control setup that any delay translates into a longer time during which the system is out of control and into an increased value of the control cost function.

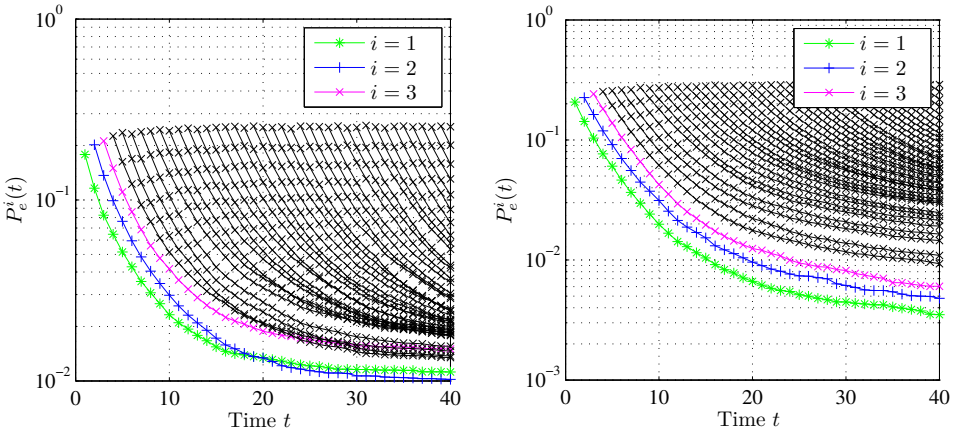
In the same way we have to be careful when interpreting the asymptotic behavior of the SCLDPC codes. The asymptotic analysis is based on the density evolution equation that takes into account the degrees of variable and check nodes but does not take into account the concrete structure as it is done for our codes.

Finite-Length Behavior

Owing to the fact that SCLDPC anytime codes have a far lower variable and check node degree than the codes proposed in this thesis, the probability of cycles in the protograph is significantly reduced. Note however that this does not mean that the finite-length behavior is always close to the asymptotic behavior. In fact erasure floors are observed just as for the codes proposed in this thesis. Again we have to reduce the block length to a very small value in order to be able to simulate the phenomenon. Fig. 3.23a shows the finite-length behavior for a $(3, 6)$-SCLDPC code with $\lambda = 0.3$,$\gamma = \infty$ and block length $k = 3$ when used for transmission over a BEC with $\epsilon = 1/3$. We can see that the erasure probability flattens out in a similar way as in Fig. 3.13. The reason for this are cycles in the parity-check matrix due to the high value of λ. In Fig. 3.23b we see a similar behavior. Here the simulated performance of a $(6, 12)$-SCLDPC code is shown with $\lambda = 0.1$ and $\gamma = \infty$. Again we see an erasure floor instead of an exponential decay for the erasure probability of all messages. Whereas λ is small, in this case the high degrees of variable and check nodes lead to an increased number of cycles. Note that since the error exponent is given as $\alpha = \lambda d_v$ in order to achieve high error exponents with SCLDPC codes we have to choose either a high λ or a high degree d_v.

Differences in the Finite-Length Analysis

The finite-length estimation presented in [NARNL15] is a technique that takes into account the variation of the number of erasures per block of k bits; however, it does not take into account the problems related to cycles in the parity-check matrix since the approach uses the density evolution equations. Applying the density evolution equations implies that the assumption of a cycle-free graph is made. But in general the graph is not cycle-free. The estimation technique can therefore only work well for codes with very few cycles and it can clearly not capture the finite-length behavior related to cycles observed in Fig. 3.23. Since the codes proposed in this thesis contain a lot of cycles the finite-length estimation technique presented in [NARNL15] is not useful in our case.



(a) $(3, 6)$-SCLDPC anytime code with $\lambda = 0.3, \gamma = \infty$.

(b) $(6, 12)$-SCLDPC anytime code with $\lambda = 0.1, \gamma = \infty$.

Figure 3.23: Performance of a (d_v, d_c)-SCLDPC anytime code for $k = 3$ when transmitting over a BEC with $\epsilon = 1/3$.

Comparison of the Error Floors

Due to the differences in the performance evaluation explained above we see little value in comparing our codes with SCLDPC-anytime codes averaged over the ensemble in terms of the behavior over the delay. A valid comparison can probably only be made between the final level of the erasure floors of the probability of decoding error. Fig. 3.24 shows the performance of SCLDPC anytime codes together with the performance obtained for the codes proposed in this thesis, when transmitting messages of size $k = 12$ bits over a BEC with $\epsilon = 1/4$ at rate $R = 1/2$. The operational anytime exponent of our codes is $\alpha_o = -\log_2(0.25) = 2$. In order to obtain SCLDPC-anytime codes with the same exponent α_o we need to choose $\lambda d_v = 2$. We show two choices: $\lambda = 0.5, d_v = 4$ and $\lambda = 2/3, d_v = 3$. We can see that the SCLDPC-codes quickly emerge in an erasure floor, whereas the erasure floor for the codes proposed in this thesis is below 10^{-7}. Judging from [NARNL15] the SCLDPC anytime codes are more suited for higher erasure rates.

3.4.3 Conclusion

We conclude that the anytime codes developed prior [SH11b] and simultaneously [NARNL15] to this work are very interesting approaches. The Toeplitz codes developed in [SH11b] lack however possible extensions to transmission over the AWGN or other channels. As compared with the SCLDPC anytime codes presented in [NARNL15] the lack of an accurate finite-length analysis reflecting the behavior

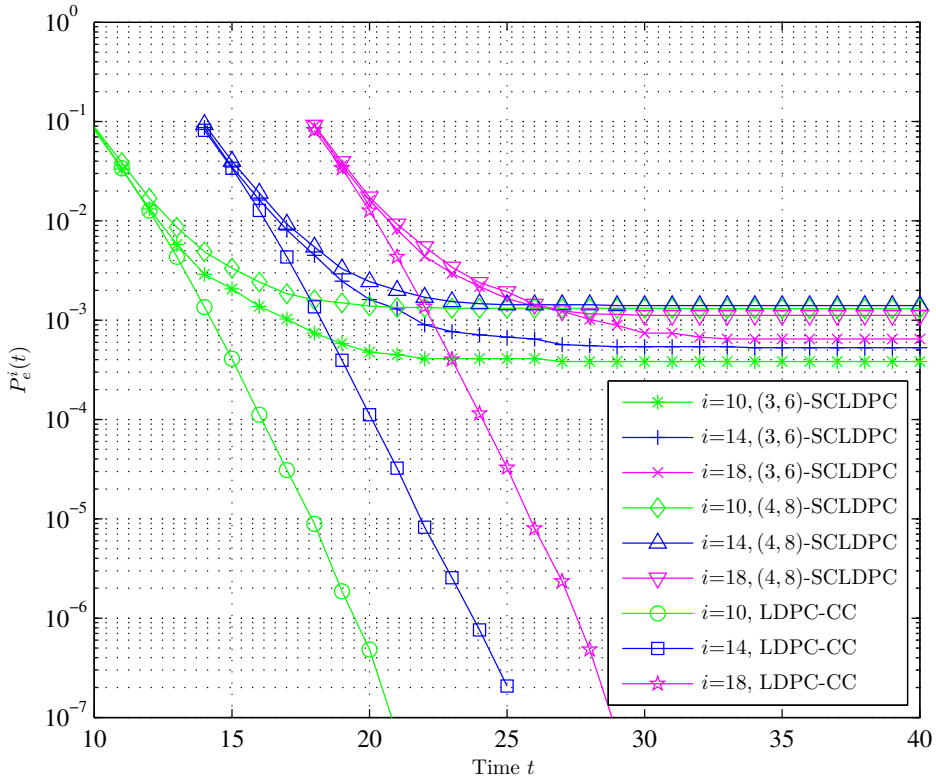


Figure 3.24: Comparison of our proposed codes (LDPC-CC) with two SCLDPC anytime codes with the same anytime exponent: A $(4, 8)$-SCLDPC anytime code with $\lambda = 0.5$ and a $(3, 6)$-SCLDPC anytime code with $\lambda = 2/3$. $\gamma = \infty$. Transmission over a BEC with $\epsilon = 1/4$ using a message length of $k = 12$.

related to cycles and the vague performance evaluation over delay of the codes motivated our choice of further investigating the codes proposed in this thesis.

Chapter 4

Modified Code Structures

In this chapter we propose different modifications of the original code structure presented in Chapter 3 and investigate the effect of the modifications on the performance of the codes. The goal is to show how we can adjust the performance of the code to practical constraints by making changes to the original structure. The constraints considered are delay-sensitivity, an optimized performance for very short message blocks and complexity. The delay sensitivity is determined by the anytime exponent. The larger the anytime exponent is, the faster is the decay of the erasure probability. In a scenario with high delay-sensitivity we therefore need to be able to design codes with large anytime exponents. Our first modification increases the degrees of the variable and check nodes in the protograph. By doing so we can construct codes with a larger anytime exponent. In another scenario where we have severe constraints on the size of the message block length we need to be able to design codes which work even for very short block lengths. A modification of the codes that decreases instead the degrees of the variable and check nodes is proposed in order to achieve a good performance for very short block lengths. If in the same scenario we are allowed to increase the codeword length we can show that an alternative suitable modification is to lower the rate of the code. Severe constraints on the complexity can be met by restricting the memory of the code which is the final modification that we present in this chapter. There are of course many more modifications possible. The main difficulty lies in finding protograph structures with a good balance between the constraints mentioned above, which at the same time have a structure that is still amenable for analysis. For all modifications we analyze the complexity (in terms of the number of edges in the protograph), the asymptotic behavior (in terms of the operational anytime exponent) and the finite-length behavior (in terms of the reachable erasure floor and the probability $P(t)$ of having an increasing information erasure pattern of size ϵkt). Due to reasons of simplicity we only examine the performance for transmission over the BEC. We conjecture a similar behavior when transmitting over the AWGN channel.

4.1 Increasing the Degrees in the Protograph

Systems with a high delay-sensitivity require a coding scheme where the decoder is capable of delivering message estimates with a low erasure probability after a very short delay. If the anytime exponent of the code is high, we can achieve good erasure probability levels already after a short delay. The anytime exponent can be increased for instance by increasing the density of the code. This can be done in many different ways. A straightforward way that can be analyzed easily is to increase the variable and check node degrees by choosing $\boldsymbol{B}_0 = [\theta\ 1]$ and $\boldsymbol{B}_\ell = [\theta\ 0]$ where θ is an integer larger or equal to 1. Note that we only modify the degrees of the information variable nodes in the protograph but not the degrees of the parity variable nodes. The reason for this is that in order to maintain the anytime property we want to keep the upper bound given in (3.26) on the mutual information $I^{\infty}_{Av,t}(i,j)$ for odd j constant. This becomes more clear in the asymptotic analysis.

4.1.1 Complexity

Since the number of edges in the protograph is increased, the complexity of the code is increased. Measuring over a window of size $2kt$ we now have a total number of

$$\#^t_{edges} = \frac{\theta kt(\theta kt+1)}{2} + kt \tag{4.1}$$

edges in the protograph. For large t this is roughly θ^2 times more than in the original structure. It is shown in [SAYK10] that the complexity per iteration of the decoding algorithm scales linearly with the number of edges in the protograph.

4.1.2 Asymptotic Analysis

The analysis of the asymptotic behavior is done exactly the same way as in Section 3.2.1 however when determining the steady-state behavior instead of inserting $B_t(\lceil (j/2) \rceil, j) = 1$ in (3.33) (repeated below) we insert $B_t(\lceil (j/2) \rceil, j) = \theta$:

$$P_{\mathrm{APP},t}(j) = \epsilon \prod_{s=\lceil j/2 \rceil}^{t} \left(1 - I^{\infty}_{Av,t}(s,j)\right)^{B_t(s,j)} \tag{4.2}$$

$$= \epsilon \prod_{s=\lceil j/2 \rceil}^{t} \left(1-(1-\epsilon)\right)^{B_t(s,j)}. \tag{4.3}$$

Here, we have used the fact that $I^{\infty}_{Av,t}(s,j)$ is equal to $(1-\epsilon)$ in steady-state even for the modified code. We then immediately get

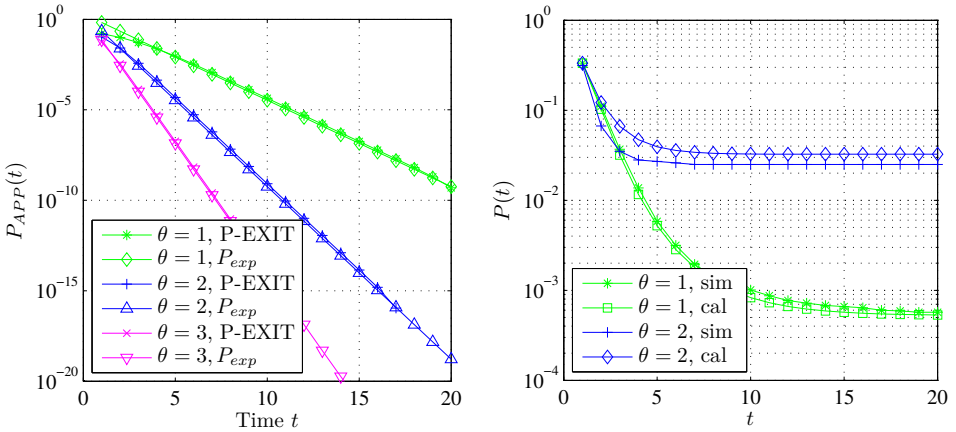
$$P_{\mathrm{APP},t}(j) = \epsilon^{\theta(t-i+1)} \tag{4.4}$$

$$= P_{\mathrm{APP},t+1}(j)\epsilon^{\theta} \tag{4.5}$$

$$= P_{\mathrm{APP},t}(j+2)\epsilon^{\theta}. \tag{4.6}$$
$$\tag{4.7}$$

It follows directly that the anytime exponent of the modified structure is equal to

$$\alpha_o(\theta,\epsilon) = -\theta\log_2(\epsilon). \tag{4.8}$$



(a) P-EXIT analysis and corresponding curve P_{exp} decaying with the calculated anytime exponent. Simulations for $\epsilon = 1/3$.

(b) Simulated and calculated probability $P(t)$ for different values of θ. Here $k = 3, \epsilon_s = 1/3$.

Figure 4.1: P-EXIT analysis and analysis of the probability $P(t)$ for the code with a higher variable and check node degree.

In Fig. 4.1a the P-EXIT analysis of the modified code is shown for three different protographs with different values of θ. In the same plot we show curves decaying with the anytime exponent $\alpha_o(\theta,\epsilon)$ calculated above. We can see that by increasing the density of the code through the parameter θ we can achieve a steeper slope of the curve. The slopes of the decoding erasure probability obtained with the P-EXIT analysis correspond perfectly with the theoretically determined anytime exponents $\alpha_o(\theta=1,\epsilon) = -\log_2(\epsilon), \alpha_o(\theta=2,\epsilon) = -2\log_2(\epsilon)$ and $\alpha_o(\theta=3,\epsilon) = -3\log_2(\epsilon)$.

4.1.3 Finite-Length Behavior

While increasing the degrees in the protograph leads to a better anytime exponent in the asymptotic limit, it has a negative impact on the finite-length behavior: The parity check matrix corresponding to the protograph contains more cycles. An increased number of cycles raises the probability of an increasing erasure event. This is shown in Fig. 4.2. Here we simulate the probability of decoding erasure $P_e^i(t)$ over time for four message blocks with different indices i when transmitting

over a BEC with $\epsilon = 0.25$ using a message length of $k = 12$ bits. We compare the behavior over time for the original structure, that is $\theta = 1$, with modified codes with $\theta = 2$ and $\theta = 3$. We can see that the curves corresponding to the higher degrees drop with a slightly steeper slope. A low erasure rate is reached within a smaller delay than compared to the original code. However due to the increased number of cycles in the codes with a higher degree we see that the erasure floors reached by the codes with $\theta = 2$ and $\theta = 3$ are far higher than the one reached by $\theta = 1$. There clearly is a tradeoff between delay-sensitivity and the erasure floor. Other simulations not shown here reveal that with increasing message length k and/or lower erasure rate ϵ the difference becomes even more obvious.

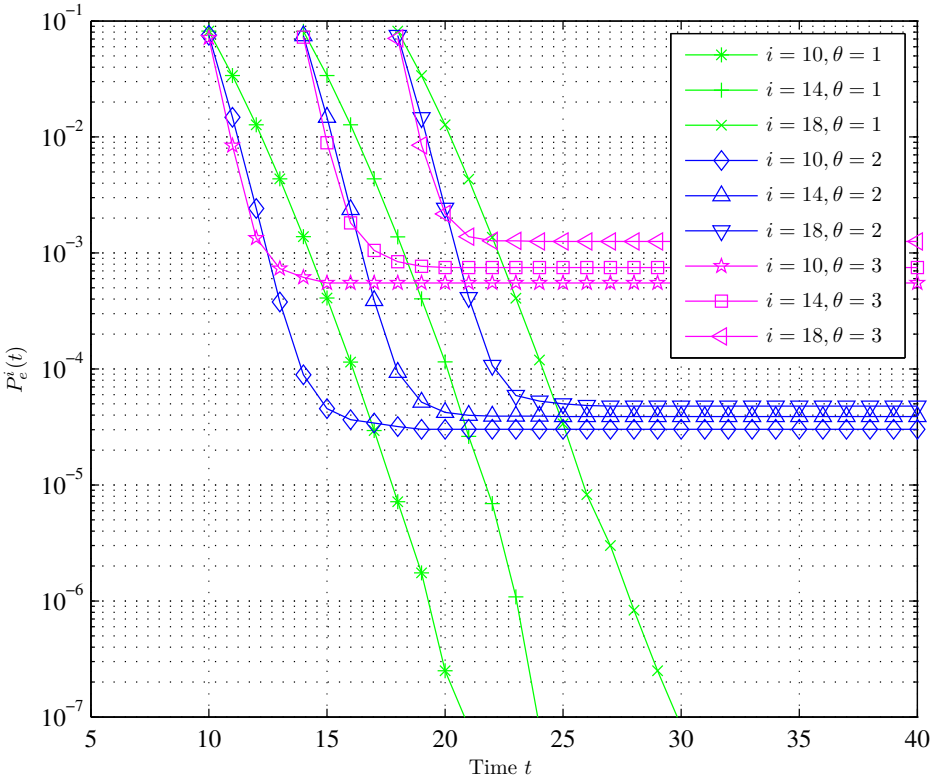


Figure 4.2: Probability of decoding erasure $P_e^i(t)$ over time values of θ. Here $\epsilon = 0.25, k = 12$.

4.1.4 Finite-Length Analysis

The finite-length analysis of the modified scheme follows the analysis for the original structure given in Subsection 3.2.3. However, in order to take into account the increased degrees we need to alter the probability $P(t|t-1)$. This is because a check node C_t^l has now approximately θ times more connections to variable nodes than compared to the original structure. The variable n_t^l denoting the number of erased information bits connected to check node C_t^l is then distributed such that

$$\Pr\{n_t^l = 1 | s_{t-1} = k\epsilon_s(t-1)\} = \theta t \epsilon_s (1-\epsilon_s)^{\theta t - 1}. \tag{4.9}$$

The probability $P(t|t-1)$ is then approximated as

$$P(t|t-1) \approx (1 - t\theta\epsilon_s(1-\epsilon_s)^{t\theta-1})^{k(1-\epsilon_s)}. \tag{4.10}$$

As in the finite-length analysis of the original structure, we assume that the variables n_t^l are independent. Due to the higher degree this assumption becomes however more and more inaccurate with increasing θ. As for the original structure $P(t)$ is obtained recursively as $P(t) = P(t-1)P(t|t-1)$. Fig. 4.1b shows the simulated probability $P(t)$ and the probability obtained through the approximation above for two protographs with different θ. We confirm that by increasing θ it is more likely for the code to get locked into an increasing erasure event since with increasing θ the erasure floor increases.

4.1.5 Conclusion

Increasing the degrees in the protograph by increasing θ leads to a faster decay of the decoding erasure probability over delay. This favourable behavior comes however with a higher complexity and a higher erasure floor. The modification is therefore sensible only if the message length k is large and/or the erasure rate is very low.

4.2 Decreasing the Degrees in the Protograph

In the previous section we have seen how an increase of the degrees of variable and check nodes leads to an increase of the number of cycles and thus to a worse finite-length performance. When transmitting over a channel with a high erasure rate or when using very small block lengths we want to decrease the number of cycles instead in order to obtain a good finite-length performance. This can be done by modifying the protograph in (3.4) by decreasing the number of nonzero entries:

$$\boldsymbol{B}_0 = [1\ 1], \tag{4.11}$$

$$\boldsymbol{B}_1 = \begin{cases} \boldsymbol{B}_i = [1\ 0] & \text{for } \mathrm{mod}(i,\kappa) = 0 \\ \boldsymbol{B}_i = [0\ 0] & \text{otherwise,} \end{cases} \tag{4.12}$$

where κ is an integer. For $\kappa > 1$ we obtain a less dense protograph compared to (3.3). Note that as for the case when we increase the degrees, we do not change the degrees of the parity variable nodes. Again this is to keep the upper bound given in (3.26) on the mutual information $I^{\infty}_{Av,t}(i,j)$ for odd j constant.

4.2.1 Complexity

Decreasing the degrees in the protograph leads to fewer edges in the protograph and therefore to a lower complexity. The number of edges in the protograph is now given as

$$\#^{t}_{edges} = \frac{kt/\kappa(kt/\kappa+1)}{2} + kt. \tag{4.13}$$

For large t this is roughly κ^2 times less than in the original structure.

4.2.2 Asymptotic Analysis

The asymptotic analysis can be done analogously to the one of the original structure with the help of the P-EXIT analysis. Since we keep the parity node degrees constant, $P_{APP,t}(j)$ has the same form as in (4.3). However we now have to take into account that only a fraction $1/\kappa$ of entries between $s = \lceil j/2 \rceil$ and t is different from 0. We therefore get

$$\begin{aligned} P_{\mathrm{APP},t}(j) &= \epsilon^{1/\kappa(t-i+1)} && (4.14)\\ &= P_{\mathrm{APP},t+1}(j)\epsilon^{1/\kappa} && (4.15)\\ &= P_{\mathrm{APP},t}(j+2)\epsilon^{1/\kappa}, && (4.16)\\ & && (4.17) \end{aligned}$$

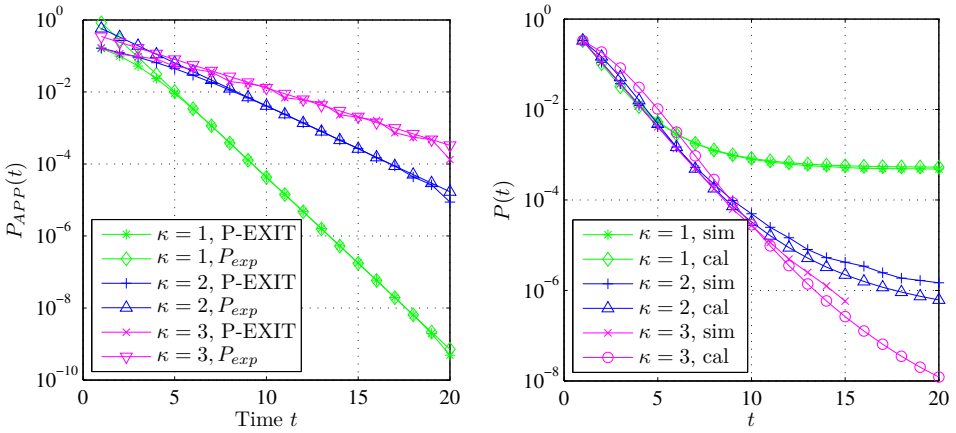
leading to an operational anytime exponent of

$$\alpha_o(\kappa,\epsilon) = -1/\kappa \log_2(\epsilon). \tag{4.18}$$

In Fig. 4.3a the P-EXIT analysis of the original code ($\kappa = 1$) together with two modified versions ($\kappa = 2$ and $\kappa = 3$) is shown for $\epsilon = 1/3$. We can see that the theoretically determined slopes match perfectly with the slopes of the decoding erasure probability obtained through P-EXIT analysis. Moreover we see how the slope becomes flatter when κ is increased. This means that a low probability of decoding erasure can only be reached with a longer delay.

4.2.3 Finite-Length Behavior

Fig. 4.4 shows the behavior of the modified code over time when simulated over a BEC with $\epsilon = 1/3$ using a message length of $k = 12$ bits. Here the probability of decoding erasure $P^{i}_{e}(t)$ is shown for messages with different indices i for different values of κ. We can see that whereas the code with the small κ emerges quickly



(a) P-EXIT analysis and corresponding curve P_{exp} decaying with the calculated anytime exponent for different $\kappa, \epsilon = 1/3$.

(b) Simulated and calculated probability $P(t)$ for different values of κ. Here $k = 3, \epsilon_s = 1/3$.

Figure 4.3: P-EXIT analysis and analysis of the probability $P(t)$ for the code structure with a lower variable and check node degree.

into an erasure floor, continues the decoding erasure probability of the codes with the larger κ to decay. At the same time however the curves corresponding to a larger κ decay with a less steep slope.

4.2.4 Finite-Length Analysis

The finite-length analysis is similar to the one for the original structure. However, in order to take into account the modification we need to alter the probability $P(t|t-1)$. This is because a check node C_t^l has now approximately κ times less connections to variable nodes than compared to the original structure. Let the variable n_t^l denote the number of erased information bits connected to check node C_t^l. The probability that $n_t^l = 1$ is then

$$\Pr\{n_t^l = 1 | s_{t-1} = k\epsilon_s(t-1)\} = 1/\kappa t \epsilon_s (1-\epsilon_s)^{1/\kappa t - 1}. \tag{4.19}$$

The probability $P(t|t-1)$ is then approximated as

$$P(t|t-1) \approx (1 - t/\kappa\epsilon_s(1-\epsilon_s)^{t/\kappa - 1})^{k(1-\epsilon_s)}. \tag{4.20}$$

Again $P(t)$ is obtained recursively as $P(t) = P(t-1)P(t|t-1)$. Fig. 4.3b shows the simulated and the calculated probability $P(t)$ using the approximation above. The curves for the original structure ($\kappa = 1$) and the modified structures ($\kappa = 2$ and $\kappa = 3$) are shown. As we can see the approximation is close to the simulated behavior. Moreover we can verify that for a protograph with a lower variable and

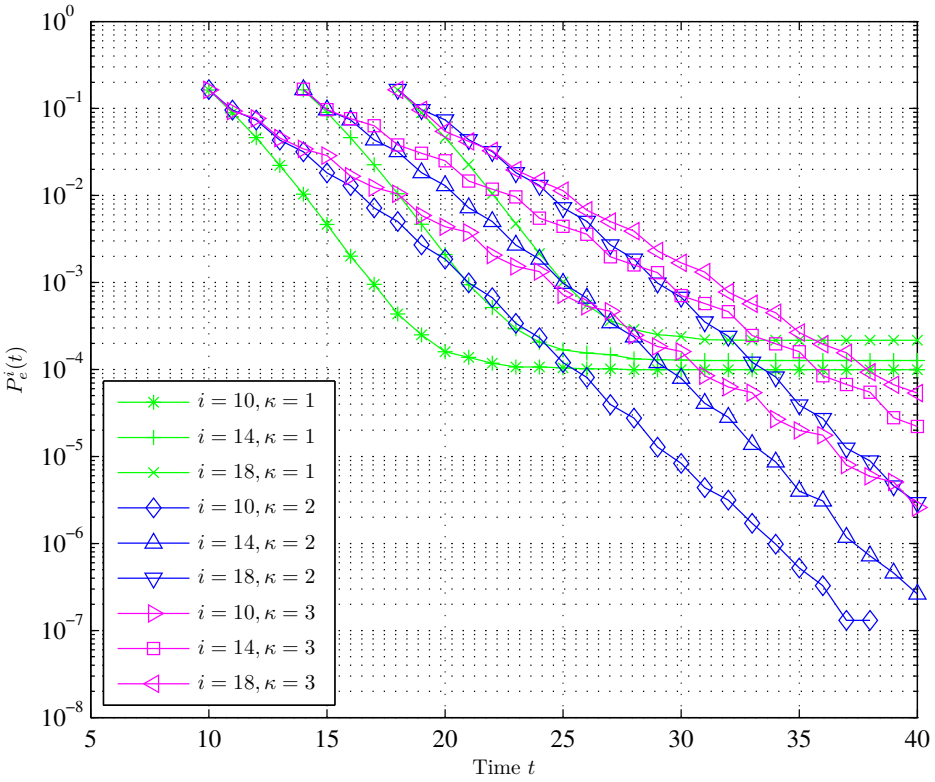


Figure 4.4: Probability of decoding erasure $P_e^i(t)$ over time for $\epsilon = 1/3, k = 12$. Comparison of a code with $\kappa = 1$ and codes with $\kappa = 2$ and $\kappa = 3$.

check node degree the probability of an increasing erasure pattern is lower than for the original structure.

4.2.5 Conclusion

We have seen that decreasing the degrees of the variable and check nodes in the protograph leads to a less steep slope of the probability of decoding erasure over time. This introduces a longer delay to achieve a certain value of the probability of decoding erasure. The average final level of the decoding erasure probability for a message in the stream is however significantly lower for a code with a lower check and variable node degree. The reason for this is that the probability of an increasing erasure pattern is decreased for less dense codes. Moreover the modified codes require a less complex decoder.

4.3 Decreasing the Rate of the Code

We have seen in Section 3.2.5 that the codes with the original structure are not suitable for transmission over channels with poor quality. In an attempt to find codes that can be used even for rather high erasure rates we now investigate the performance of codes with a similar structure to the original but with a lower rate, thus assuming that more bits can be transmitted over the channel per time step t. This could be achieved for instance by a higher bandwidth. We now describe the structure of the lower rate codes that we study. A rate $R = 1/m$ code where m is an integer with $m > 2$ can be obtained from the original structure by modifying B_0 such that we add $m-1$ rows and place exactly two 1s in every new row. B_1 is changed accordingly:

$$\boldsymbol{B}_0 = \begin{bmatrix} 1 & 1 & & & & \\ 1 & 0 & 1 & & & \\ 1 & 0 & 0 & 1 & & \\ 1 & 0 & 0 & 0 & 1 & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \qquad \boldsymbol{B}_1 = \begin{bmatrix} 1 & 0 & \dots \\ 1 & 0 & \dots \\ 1 & 0 & \dots \\ 1 & 0 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix} \tag{4.21}$$

As for the other modifications we again do not change the degree of the variable nodes in the protograph corresponding to code bits. The reason for this becomes obvious in the asymptotic analysis.

4.3.1 Complexity

Decreasing the rate of the code leads to more edges in the protograph. The number of edges in the protograph is now given as

$$\#^t_{edges} = (m-1)\frac{kt(kt+1)}{2} + (m-1)kt. \tag{4.22}$$

For large t this is $m - 1 = 1/R - 1$ times more than in the original structure.

4.3.2 Asymptotic Analysis

The modified structure of the protograph implies that instead of one set of k new check equations per message block we have $(1/R-1)$ sets of k new check equations. By choosing $\boldsymbol{B}_1$ such that the degrees of parity nodes have an unchanged degree equal to 1, we decouple however the check equations in a way that for each row in the protograph the upper bound $I^{\infty}_{Av,t}(i,j) \leq (1-\epsilon)$ is still valid. Therefore in steady-state the only modification we have to make in $P_{APP,t}(j)$ is that the product is now $(1/R-1)$ times larger. We therefore get

$$P_{\mathrm{APP},t}(j) = \epsilon^{(1/R-1)(t-i+1)} \tag{4.23}$$

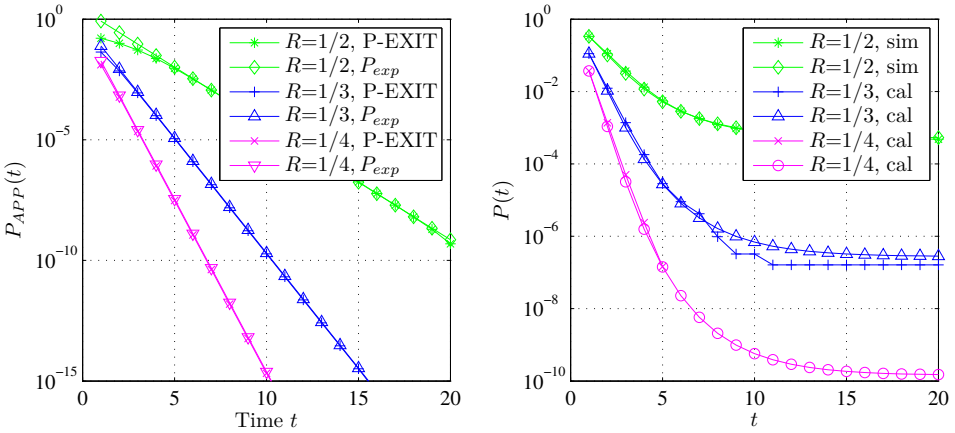
$$= P_{\mathrm{APP},t+1}(j)\epsilon^{(1/R-1)} \tag{4.24}$$

$$= P_{\mathrm{APP},t}(j+2)\epsilon^{(1/R-1)}, \tag{4.25}$$

leading to an operational anytime exponent of

$$\alpha_o(R,\epsilon) = -(1/R-1)\log_2(\epsilon). \tag{4.26}$$

This can be verified in Fig. 4.5a where we plot the asymptotic performance obtained through P-EXIT analysis together with curves decaying with the operational anytime exponent determined above for different rates R. We can see that the anytime exponent can be determined accurately. Being able to transmit more bits per time instance leads moreover to a steeper decay of the asymptotic decoding erasure probability.



(a) P-EXIT analysis and corresponding curve P_{exp} decaying with the calculated anytime exponent for different rates R, $\epsilon = 1/3$.

(b) Simulated and calculated probability $P(t)$ for different values of the rate R. Here $k = 3$ and $\epsilon_s = 1/3$.

Figure 4.5: P-EXIT analysis and analysis of the probability $P(t)$ for different values of the rate R.

4.3.3 Finite-Length Behavior

The finite-length behavior for a code with lower rate is compared to the performance of the original code in Fig. 4.6. Here we plot the simulated probability of decoding erasure $P_e^i(t)$ for three different message indices i for different rates R. As we can see the probability $P_e^i(t)$ corresponding to the code with the lower rate decays both faster over time and reaches a lower erasure floor. With the lower rate code we can therefore cope with erasure rates for which the $R = 1/2$-anytime code is unsuitable.

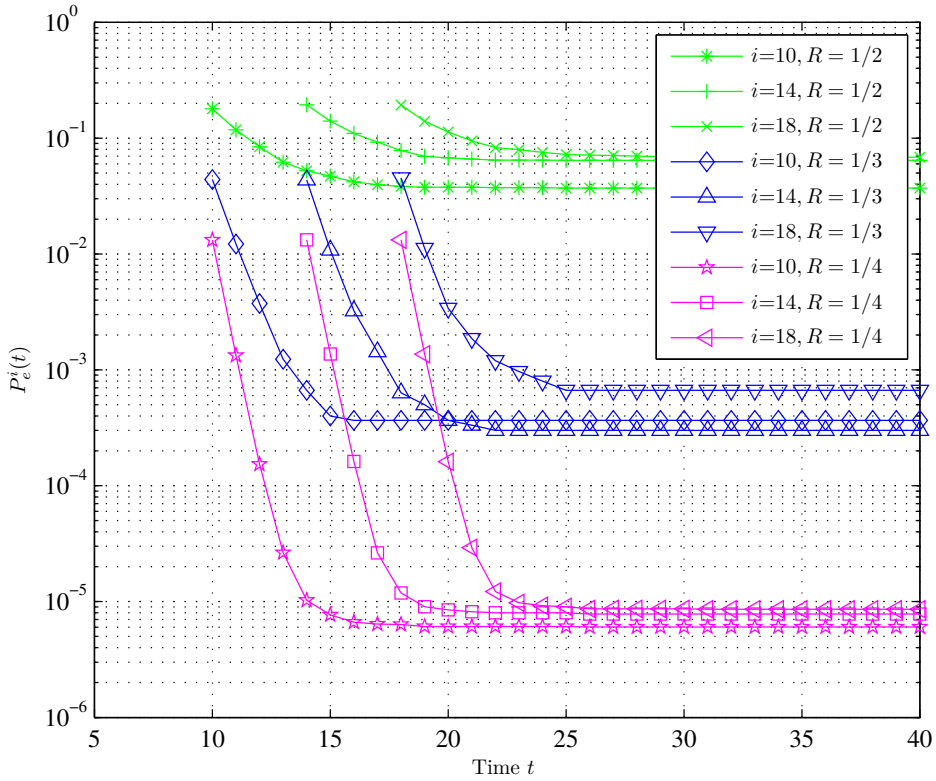


Figure 4.6: Probability of decoding erasure $P_e^i(t)$ over time for $\epsilon = 1/3, k = 3$. Comparison of codes with different rates R.

4.3.4 Finite-Length Analysis

The finite-length analysis of the lower rate codes is performed similar to the analysis of the original structure. However, instead of having only $k - E$ new unerased code bits per time step t as in the original structure, due to the lower rate of the code, in every time step there are now $(1/R - 1)$ times as many unerased code bits to be considered. The probability $P(t)$ is therefore approximated with

$$P(t) \approx P(t=1)^{(1/R-1)} \left[\prod_{i=2}^{t} \left(1 - i\epsilon_s \left(1 - \epsilon_s\right)^{i-1} \right) \right]^{k(1-\epsilon_s)(1/R-1)} \tag{4.27}$$

In Fig. 4.5b we can verify that the above approximation is close to the simulated probability $P(t)$. Here we depict the simulated probability $P(t)$ together with the approximated probability $P(t)$ when transmitting over the static BEC with $\epsilon_s = 1/3$ using a message length of $k = 3$ bits and different rates R. Since the curves flatten

out, we understand that for lower rate codes as for the $R = 1/2$-codes the problem of increasing erasure patterns remains. The probability that such a problem occurs is however lowered when decreasing the rate.

4.3.5 Conclusion

We have proposed a modification of the original structure that allows the transmission of the data at a lower rate. With this modification we can improve both the asymptotic performance, in terms of the operational anytime exponent, as well as the finite-length performance over delay. While we cannot completely eliminate increasing erasure patterns by decreasing the rate we can lower the probability of their appearance significantly. This allows us to transmit over erasure channels with high erasure rates. Note however that a lower rate leads to a decoder of higher complexity.

4.4 Limiting the Memory of the Code

In the original structure and the modified codes proposed so far, the decoder is allowed to work on a parity-check matrix of any size. The occurrence of large decoding windows is however small, since as we have seen in Section 3.2.5, the probability of having a decoding window of size ω decays exponentially in ω. This does however not mean that the decoding window is always small. Even though with a low probability the decoding window might occasionally be very large. Most systems do not tolerate very large matrices not even if only for a short period of time but can only cope with a fixed maximum size d_{max} for encoding and decoding matrices. What most practical systems can tolerate are however a small error probability in the received stream instead. We therefore now design codes that have a fixed maximum size encoding and decoding matrix but do not require the erasure probability to eventually go to zero. Theses constraints are fulfilled by modifying our codes to have band structure with a fixed bandwidth of $m_s \leq d_{max}$. Then decoding can be performed with the sliding window decoder proposed in [LC07] instead of the expanding window decoder proposed in Chapter 3. We will show in the following that if the memory is properly chosen it is no longer possible to have increasing erasure patterns. The code can however only achieve anytime performance up to a maximum delay.

4.4.1 Complexity

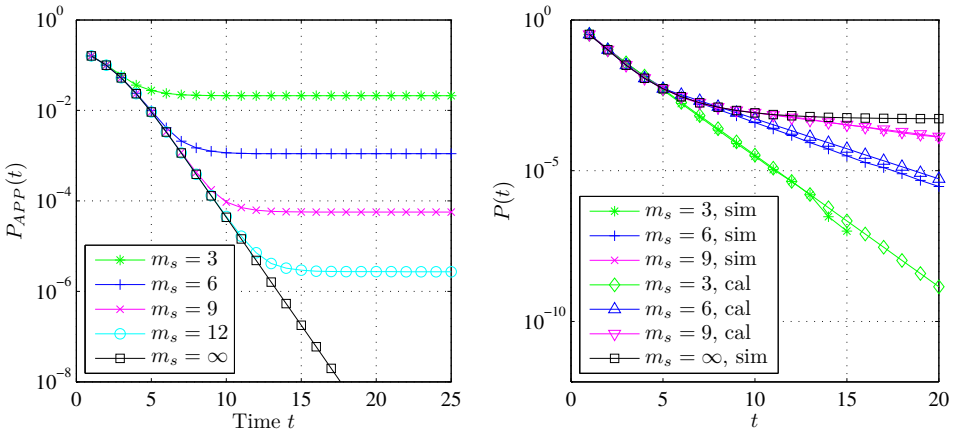
Due to the memory limitation the number of edges in the parity check matrix does not increase over time but stays constant as soon as $t > m_s$. In steady state the number of edges for a code with memory m_s is therefore given as

$$\#^t_{edges} = \frac{km_s(km_s+1)}{2} + km_s. \tag{4.28}$$

This means that by limiting the memory of the code we obtain a constant complexity. This is obviously a huge advantage compared to the complexity of the original structure which is linearly increasing over time.

4.4.2 Asymptotic Analysis

Fig. 4.7a shows the asymptotic analysis of the codes with different values for the memory m_s obtained through P-EXIT analysis. We see the occurrence of an erasure floor. The larger the memory, the lower is the erasure floor. The erasure floor occurs at the time the memory m_s is reached. This is because a message node i is no longer involved in new check equations from the point $i + m_s$ in time onwards. During the time $t = i \dots i + m_s$ the probability of decoding erasure drops with the same anytime exponent as for the original structure. Changing the memory of the code does therefore not change the delay-sensitivity of the structure.



(a) P-EXIT analysis for the modified structure with different memories m_s, $\epsilon = 1/3$.

(b) Simulated and calculated probability $P(t)$ for different values of m_s, ϵ_s=1/3, k=3.

Figure 4.7: Analysis of the protograph structure with memory m_s.

4.4.3 Finite-Length Behavior

Fig. 4.8 shows the finite-length behavior of the memory-limited codes over time obtained through simulations. Here we show the probability $P_e^i(t)$ for different values of the memory m_s when transmitting over a BEC with $\epsilon = 1/3$ using a message length of $k = 12$ bits. The curves corresponding to different memories decay with the same anytime exponent but we can see that the code with the larger memory (and therefore higher complexity) has a lower erasure floor. But whereas the erasure floor for the code with the higher memory is increasing over the message

index i, the erasure floor for the code with the lower memory is constant for all message indices i. In fact there is a maximum memory m_{max} after which a further increase of the memory does not lead to an even lower erasure floor. Finding the maximum memory is subject of Subsection 4.4.4.

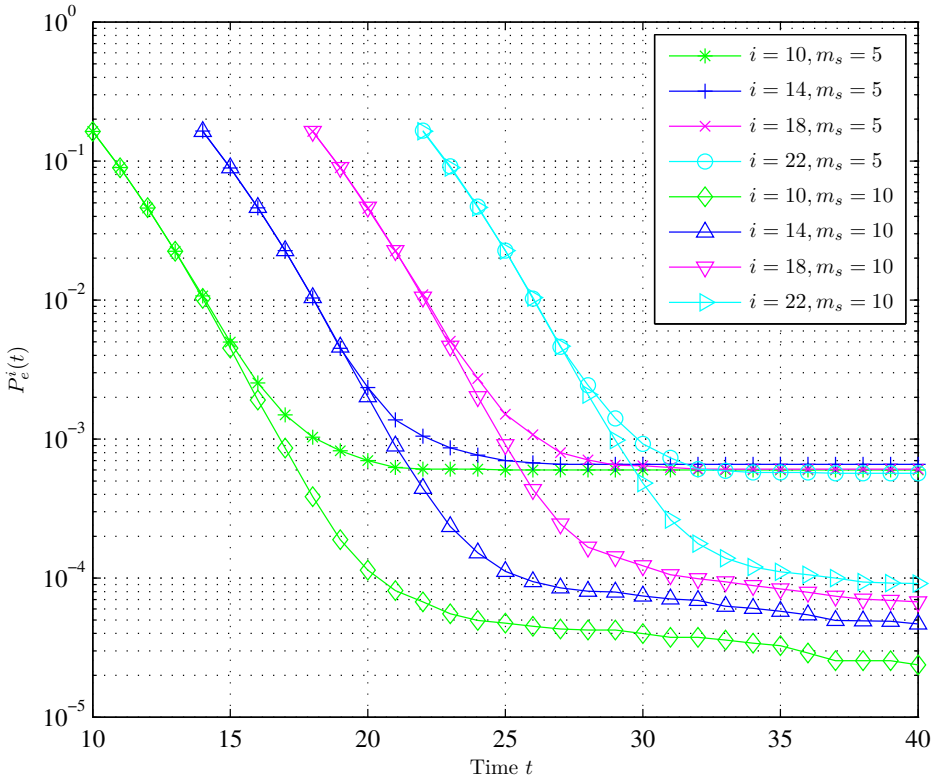


Figure 4.8: Probability of decoding erasure $P_e^i(t)$ over time for different memory constraints m_s. Simulations for $\epsilon = 1/3, k = 12$.

4.4.4 Finite-Length Analysis

The finite-length analysis for the memory-constrained code is done analogously to the finite-length analysis of the original structure. However, due to the limited memory the number of variable nodes connected to a check node C_t^l is limited to m_s:

$$P(t) = P(t|t-1)P(t-1) \tag{4.29}$$

$$\approx P(t=1)\left[\prod_{i=2}^{\min(t,m_s)}\left(1-i\epsilon_s\left(1-\epsilon_s\right)^{i-1}\right)\right]^{k(1-\epsilon_s)} \tag{4.30}$$

Fig. 4.7b shows the simulated probability $P(t)$ together with the approximated probability $P(t)$ for different values of the memory m_s when transmitting over a static BEC with $\epsilon_s = 1/3$ using a message length of $k = 3$ bits. We can verify that the above approximation of $P(t)$ is close to the simulated probability $P(t)$. Furthermore we can see that the further we increase the memory the more we approach the curve obtained for infinite memory. This means that it is more likely to have an increasing erasure pattern for a larger memory than for a smaller memory. This suggests that the choice of the memory is crucial. An optimal way of choosing the memory is given in the following.

Choosing the Right Memory

We have seen that on the one hand in the asymptotic limit an increased memory leads to a lower final erasure floor; on the other hand choosing a larger memory increases the probability of the presence of an increasing erasure pattern, which in turn has a negative influence on the finite-length performance. We therefore conclude that there is a maximum memory $m_{max} > 0$ ($m_{max} \in \mathbb{N}$) after which a further increase of m_s does not lead to a better performance over time. The maximum memory m_{max} can be determined with the help of the probability $P(t)$. Recall from Section 3.2.4 that the turning point is the point where $P(t)$ starts to flatten out and the probability that an erasure pattern increases becomes larger than the probability that it decreases. Considering the turning point is therefore a suitable approach to determine the maximum memory:

$$P(t|t-1) = \frac{P(t)}{P(t-1)} \tag{4.31}$$

$$\approx \frac{P(t=1)\left[\prod_{i=2}^{t}(1-i\epsilon_s(1-\epsilon_s)^{i-1})\right]^{k(1-\epsilon_s)}}{P(t=1)\left[\prod_{i=2}^{t-1}(1-i\epsilon_s(1-\epsilon_s)^{i-1})\right]^{k(1-\epsilon_s)}} \tag{4.32}$$

$$\approx \left[(1-t\epsilon_s(1-\epsilon_s)^{t-1})\right]^{k(1-\epsilon_s)}. \tag{4.33}$$

The maximum memory m_{max} is then given as

$$m_{max} = \max\{t \in \mathbb{N} | \left[(1-m_{max}\epsilon_s(1-\epsilon_s)^{m_{max}-1})\right]^{k(1-\epsilon_s)} \leq 0.5\}. \tag{4.34}$$

The effect of choosing memories below or above m_{max} is illustrated in Fig. 4.9. Here we show the simulated probability of decoding erasure over time for the 10th message block when transmitting over a BEC with $\epsilon = 1/3$ using a message length of $k = 12$ bits. The maximum memory is determined to $m_{max} = 6$. We can see

that increasing the memory up to m_{max} decreases the erasure floor. For memories larger than m_{max} a slightly lower erasure floor is achieved but at the same time the rate at which the decoding erasure probability improves gets smaller. For memories even larger than shown in Fig. 4.9 we observe an increasing erasure floor over the message index i as we did for the code with infinite memory.

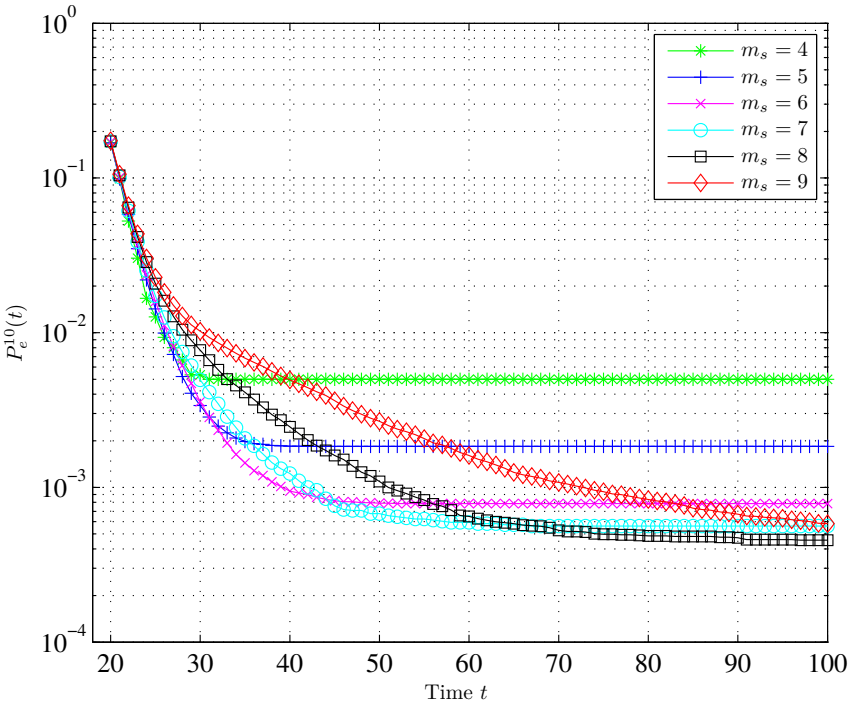


Figure 4.9: Probability of decoding erasure $P_e^{10}(t)$ over time for different memory constraints m_s. Simulations for $\epsilon_s = 1/3, k = 3$.

4.4.5 Conclusion

By limiting the memory of the protograph we can keep the same anytime exponent as for the original structure however with a constant decoding complexity. The anytime property is maintained up to a certain level for all messages equally. This is a significant advantage of this modification. The choice of the memory is however crucial. We have found a maximum memory m_{max} that gives the best performance in terms of the final erasure floor and the decay over the delay. Choosing the memory below or equal to m_{max} leads to a scheme with a constant erasure floor for all message indices. This property is lost when increasing m_s beyond m_{max}.

Chapter 5

Decoding Feedback Protocols

We have seen in the previous chapters that even though the anytime codes proposed in this thesis show excellent performance for large block lengths they suffer from erasure/error floors when used with very small block lengths. The error floors occur due to the existence of increasing error patterns that the decoder fails to decode over time. This means that the transmission over the channel becomes unprotected and potentially gets so poor that control of the plant is no longer guaranteed. In a practical system it is therefore of utmost importance to either eliminate increasing error patterns by changing the code structure or to detect their occurrence and subsequently eliminate them through a feedback procedure. Whereas the focus of the previous chapter was on modifying the code structure in order to decrease the probability of the occurrence of an increasing error pattern, in this chapter the focus is on counteracting their occurrence. To this end we investigate ways to modify, extend, or alter the system in order to be able to use the proposed anytime codes in a practical system setup. The modifications considered is to involve a feedback link; the possibility to halt the input bitstream; or the possibility to transmit a varying number of bits during one time instance. Our main results are three different encoder-decoder protocols that improve the performance of the transmission scheme by counteracting increasing error patterns. At the same time the modifications lead to a reduced complexity of the decoder. In the context of increasing error patterns two probabilities were shown to be useful when investigating and analyzing the performance. These are the probability $P(t|t-1)$ that an error pattern increases when going from time $t-1$ to t and the probability $P(t) = P(t|t-1)P(t-1)$ that there is a maximum size error pattern stretching over t time steps. Both probabilities were introduced in Subsection 3.2.3. Using these probabilities we first devise a mechanism for the decoder to detect an increasing erasure pattern for the BEC. Likewise we propose a technique that can be used for detecting increasing error patterns for the AWGN channel. We investigate the effect of terminating the anytime LDPC-CCs in regular time intervals. Finally we explore three strategies that involve a feedback link. In this chapter we design and analyze algorithms for transmission over either the BEC or the AWGN channel. For simplicity we

sometimes use the common term *increasing error pattern* to describe both the problematic error patterns when transmitting over the AWGN channel and the problematic erasure patterns when considering transmission over the BEC.

5.1 Detecting Growing Error Events

In order to use the proposed codes in practical transmission systems we need a mechanism to detect growing error events since their occurrence causes the decoding complexity to grow linearly over time with no improvement of the decoding error probability. Depending on the model under consideration, the growing error event detection process can be quite different. Both detection processes that we propose here, are based on a threshold test.

5.1.1 Transmission Over the BEC

Given the decoding algorithm in Algorithm 3.1, when transmitting over the BEC, the decoder has at any point in time perfect knowledge about which received bits are known and which bits are unknown. With this information the decoder can determine in every time step the current number s_t of unknown information bits. A simple way to detect an increasing erasure pattern is then to compare this number to a threshold s^{th}. If the current number of unknowns in the received bitstream exceeds the threshold s^{th}, an increasing erasure pattern is detected. Recall from Fig. 3.16 that the probability $P(t|t-1)$ that an erasure pattern increases in size from some size onwards approaches 1 with increasing size of the erasure pattern. The choice of the threshold s^{th} depends on the intended application since there is a tradeoff between the delay of detection and the probability of false alarm. Moreover the choice of the threshold depends on the feedback strategy that follows the detection of a growing erasure pattern. A complex feedback procedure should only be started if the probability of false alarm is quite small. As an example, a good choice of the threshold is obtained by the maximum time t^* for which $P(t|t-1) \leq 0.5$. This turning point was introduced in Subsection 3.2.4. The threshold s^{th} is then chosen as $s^{th} = k\epsilon s^*$, where $s^* = t^*$. Fig. 5.1 shows the probability of false alarm for detecting an increasing erasure pattern using different thresholds on the number of erasures. A detection is categorized as false alarm if the number of erasures decreases to 0 again at some time after the increasing-erasure-event-detector has declared a hit. This clearly means that there is no increasing erasure pattern present. As we can see the probability of false alarm is decreasing with increasing threshold s^{th}. Since a larger number of erasures is more likely observed at a larger delay Fig. 5.1 gives us as well an idea of the tradeoff between the delay of detection and the probability of false alarm. In other words, the longer we wait until we declare a hit or the larger we choose the threshold, the lower is the probability of false alarm. Note that with a fixed threshold it is not possible to miss a detection. We therefore do not need to analyze this probability.

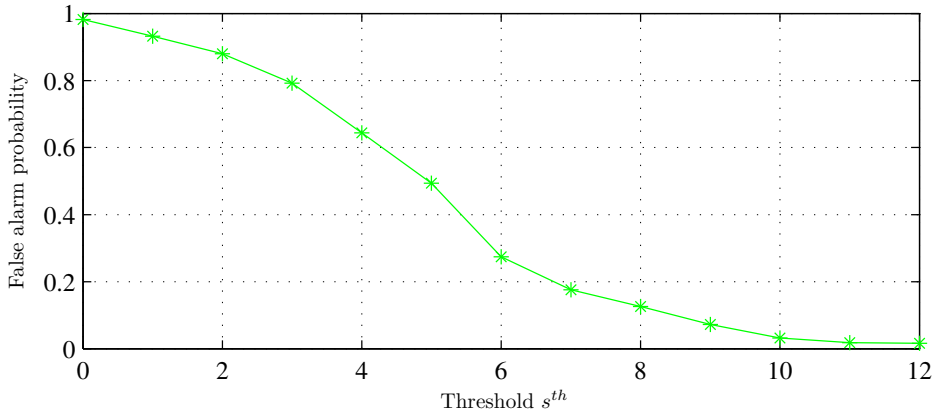


Figure 5.1: Probability of false alarm for different thresholds s^{th} on the number of erasures used for detecting a growing erasure pattern when transmitting over a BEC with $\epsilon = 1/3$ and $k = 3$.

5.1.2 Transmission Over the AWGN Channel

Detecting a growing error event when transmitting over the AWGN channel is more complicated than detecting a growing error event when transmitting over the BEC. When transmitting over the AWGN channel the decoder is only provided with soft bits and channel SNR but has no knowledge about which bits are correct and which bits are wrong. In fact bits that have been decoded correctly at one point in time might be decoded incorrectly at a later point in time. The performance of the anytime codes when used for transmission over the AWGN channel was analyzed in Section 3.3. Let $\rho(j)$ denote the SNR of the decoded bit at index j. Assuming that the LLRs provided by the decoder are Gaussian distributed, that is $p(LLR|b = 1) \propto \mathcal{N}(m_L, \sigma_L)$ then the variance σ_L is given as $\sigma_L = 2m_L$ and the SNR $\rho(i)$ of bit i is therefore given as

$$\rho(i) = |m_L|^2/\sigma_L = |m_L|/2. \tag{5.1}$$

We can estimate the occurrence of an error event by analyzing $\rho(i)$ of the SNRs of the decoded bits. As we know from Section 3.3 in the asymptotic limit the aposteriori SNRs of message $\boldsymbol{x}_j$ is

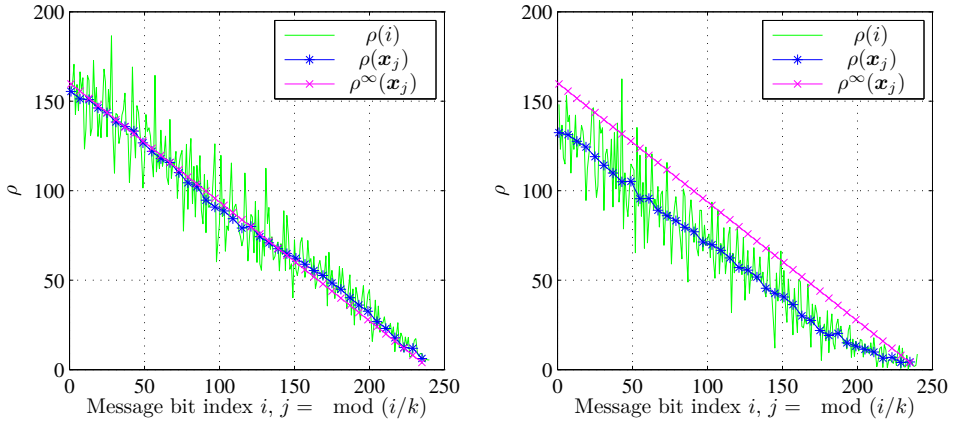
$$\rho_t^{\infty}(\boldsymbol{x}_j) \leq (|N_v(j)| + 1)\rho_{ch}, \tag{5.2}$$

here $|N_v(j)|$ denotes the number of neighbors of variable node j at time t. We can verify through simulations that the upper bound is a good approximation of the SNR and as a consequence an increasing error event can be detected if the linear increase is not maintained over some fixed duration of time. We use the

following quantities to illustrate this: Denote the average SNR for message $\boldsymbol{x}_{t-d^{th}}$ by $\rho(\boldsymbol{x}_{t-d^{th}})$ calculated as

$$\rho(\boldsymbol{x}_{t-d^{th}}) = \frac{1}{k}\sum_{i=1}^{k}\rho(x^i_{t-d^{th}}), \tag{5.3}$$

where $x^i_{t-d^{th}}$ denotes the i-th bit in message $\boldsymbol{x}_{t-d^{th}}$ and $\rho(x^i_{t-d^{th}})$ is the corresponding SNR value determined according to (5.1). Furthermore assume in the following the value of $\rho_t^{\infty}(\boldsymbol{x}_j)$ by taking (5.2) with equality. These quantities are illustrated in Figs. 5.2 and 5.3. Here we simulate the transmission of 40 messages with block length $k = 6$ bits over an AWGN channel with SNR = 3 dB and plot the bit SNR $\rho(i)$ for all message bits $i = (j-1)k + b$, $(j = 1, \ldots, 40, b = 1, \ldots, k)$, the average message SNR $\rho(\boldsymbol{x}_j)$ per message $\boldsymbol{x}_j$ and the asymptotic curve $\rho^{\infty}(\boldsymbol{x}_j)$ obtained if no error event occurs. Four different snapshots are shown, each of them coming from a different simulation. We can see how the observed magnitudes $\rho(j)$ move away from the asymptotic curve $\rho^{\infty}(\boldsymbol{x}_j)$ whenever there are errors remaining in the decoded bitstream. Note that it is not the number of errors but the position of the errors that has the strongest influence on how far away the observed magnitudes are from the asymptotic curve. If there is a large difference between the curves, the probability that the growing error event continues to propagate is very high. The

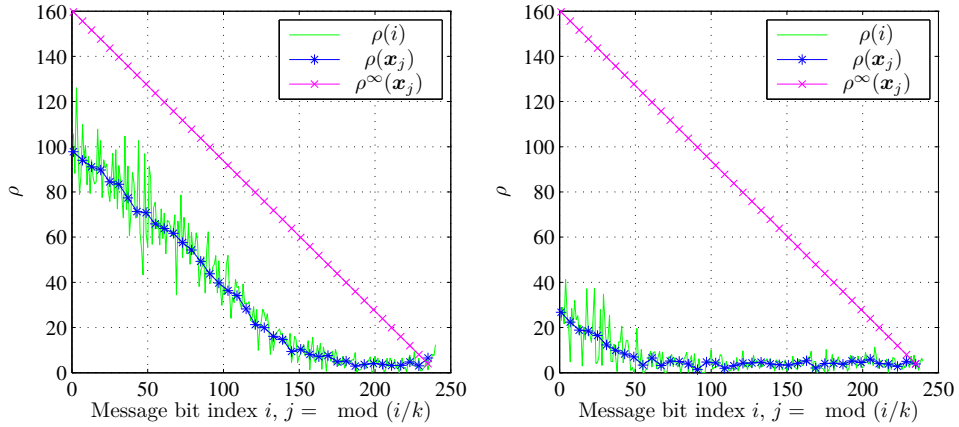


(a) No growing error event present. 0 errors in the decoded stream.

(b) Starting phase of a growing error event. 7 errors in the decoded stream.

Figure 5.2: Snapshots of measured absolute message reliability for SNR = 3 dB, $k = 6$ at $t = 40$.

error detection mechanism is a threshold test described as follows: Denote by d^{th} the delay from time t where we want to perform the threshold test and denote by $\rho^{th}(\boldsymbol{x}_d^{th})$ the value of the threshold that we want to compare to. The threshold is



(a) Growing error event present. 9 errors in the decoded stream. Still possible to be resolved.

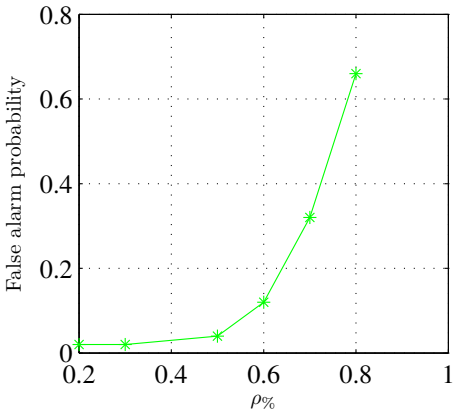
(b) Growing error event present. 46 errors in the decoded stream. Very unlikely to be resolved.

Figure 5.3: Snapshot of measured absolute message reliability for SNR = 3 dB, $k = 6$ at $t = 40$.

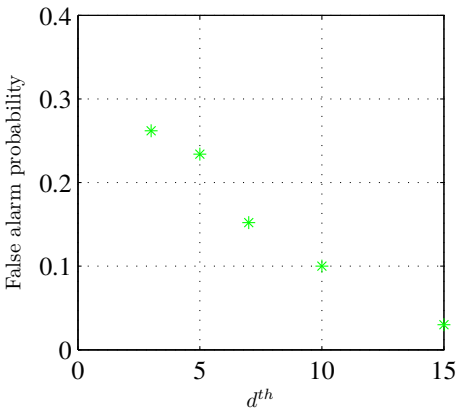
defined using a parameter $\rho_{\%}$ and the asymptotic limit ρ^{∞}

$$\rho^{th}(\boldsymbol{x}_d^{th}) = \rho_{\%}\rho_t^{\infty}(d^{th}). \tag{5.4}$$

For example $\rho_{\%} = 0.5$ means that the threshold ρ^{th} is set to 50% of the value $\rho^{\infty}(\boldsymbol{x}_{t-d^{th}})$ obtained at delay d^{th} if no error event occurs. In order to detect an increasing error event in every time step the average SNR of all messages $i \leq t$ is determined. A hit is declared if the value of $\rho(\boldsymbol{x}_{t-d^{th}})$ is below the threshold $\rho^{th}(\boldsymbol{x}_{d^{th}})$. As for the error event detection for the BEC the parameters $\rho_{\%}$ and d^{th} should be chosen depending on the application and the kind of feedback strategy that follows the detection of a growing error event. As for the BEC we can analyze the false alarm probability. A detection is declared as false alarm if at some point after the detection the number of errors in the decoded bitstream is equal to 0 again. Fig. 5.4 shows the probability of false alarm as a function of $\rho_{\%}$ and d^{th}. In Fig. 5.4a we illustrate how the probability of false alarm behaves for different choices of the parameter $\rho_{\%}$. We can see that the larger we choose the value $\rho_{\%}$ the closer is the threshold to the value of $\rho_{d^{th}}^{\infty}$ and the more often an error event is declared. This increases the false alarm probability. Fig. 5.4b shows the dependence of the false alarm probability on the decision delay d^{th}. The longer we wait until we make a decision on whether an error event has occurred or not, the lower is the probability of false alarm. Recall however that in the context of anytime coding the system is very sensitive to long delays. Again if the value of $\rho_{\%}$ is not chosen too low it is not possible to miss the detection of an error event.



(a) Tradeoff of the probability of false alarm versus percentages $\rho_{\%}$ of the ideal value used as threshold.



(b) Tradeoff of the probability of false alarm versus decision delay using a 50% threshold.

Figure 5.4: False alarm probability for different thresholds for detection of growing error events when transmitting over an AWGN channel with SNR = 3 dB and $k = 6$.

5.1.3 Conclusion

We have presented a method to detect increasing erasure patterns when transmitting over the BEC and we have determined a suitable value for the threshold. For the AWGN channel we have shown that the average absolute reliability per message together with a thresholds is a good way of detecting an increasing error pattern. We have shown the effect of changing the threshold. The methods used for increasing error event detection presented here are used in the following sections whenever necessary.

5.2 Terminating the Codes

It has been shown that terminated LDPC-CCs have an excellent performance on the erasure channel [LSCZ10]. Terminated codes as opposed to unterminated codes have a tail of τ symbols that forces the encoder to the zero state at the end of the encoding process. As compared to unterminated codes the performance can therefore be improved. Please refer to Section 2.5.4 for a complete description of the termination process. The anytime codes that we have considered so far are however not terminated. It is interesting to see how well suited it is to terminate the codes in the context of anytime coding. In order to allow for termination we need to alter the system setup in a way that the input bitstream can be halted. During the time the input bitstream is halted, the transmitter transmits the termination

bits. After that the input bitstream is resumed. We could halt the input bitstream upon a detection of a growing error event and thereafter start the termination phase. However we can as well halt the input bitstream in regular time intervals and thus avoid the feedback link. This is the setup we are going to investigate here. After a termination phase we choose not to involve previously transmitted bits in future parity check equations since this allows us to reduce the complexity of the system significantly. Thus we obtain a band diagonal structure for the parity check matrix. Whereas these advantages are desired properties, we will see that imposing a block structure on top of the anytime codes has a strong influence on the performance of the codes over time and delay. We now investigate the performance of the anytime codes when the input stream is halted after the transmission of a fixed number of L blocks. We always choose the following $\tau = m_s + 1$ blocks for terminating the LDPC-CC. Both transmission over the BEC and transmission over the AWGN channel are considered. The rate R_t obtained when taking into account the transmission of the $\tau = m_s + 1$ termination bits is then given as

$$R_t = \frac{Lk}{L2k + m_s 2k}. \tag{5.5}$$

In the following we illustrate the performance of terminated anytime codes. This is done first when transmitting over the BEC and subsequently when transmitting over the AWGN channel.

5.2.1 Transmission Over the BEC

Fig. 5.5 shows the performance over time of a terminated code with $L = 15, m_s = 10, \tau = 11$ when transmitting over a BEC with $\epsilon = 0.35$. For the ease of simulation and for clarification we use a very small block length of $k = 3$ bits. The decoding erasure probability $P_e^i(t)$ of the messages $\boldsymbol{x}_1 \dots \boldsymbol{x}_5$ is depicted over time. The code blocks with indices $i = L, \dots, L + m_s$ do not contain message bits but only code termination bits. For a block length as short as $k = 3$ bits we observe an increasing erasure floor. By terminating the codes we can however see how the erasure floor is lowered. A clearer picture of the development during the termination phase is given in Fig. 5.6a. Here we show the probability of decoding erasure of all L messages at different time steps during the termination phase. Each curve corresponds to a different time step where $t = L + i$ means that the receiver has received i packets of the overall $m_s + 1$ packets of termination bits. We can see that the probability of decoding erasure is only very slightly improved with every reception of a new block of termination bits. Only when the final termination bits have been received the probability of decoding erasure improves abruptly to a low value. In Fig. 5.6b we can see that when increasing the block length $k = 3$ to $k = 6$ bits the effect of the termination becomes even more evident. Again the erasure floor is significantly reduced only at the end of the termination phase. Figs. 5.7a and 5.7b investigate the average probability $\overline{P_e^{i=1\dots L}(L)}$ for different interval lengths L after which the code is terminated and the resulting rate R_t: The average

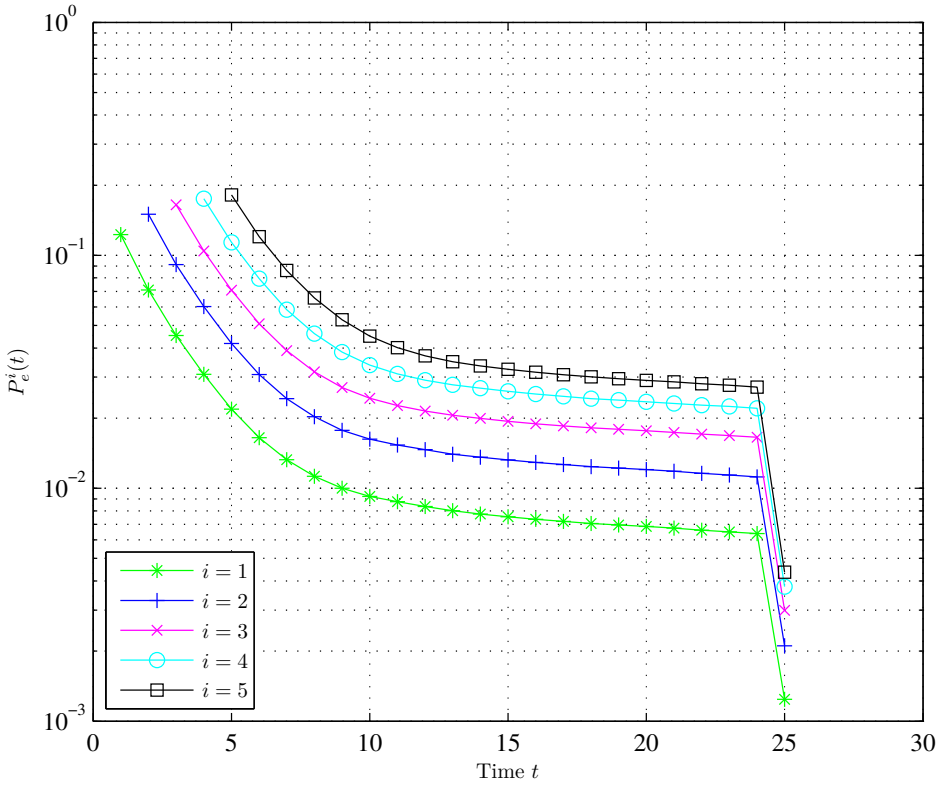
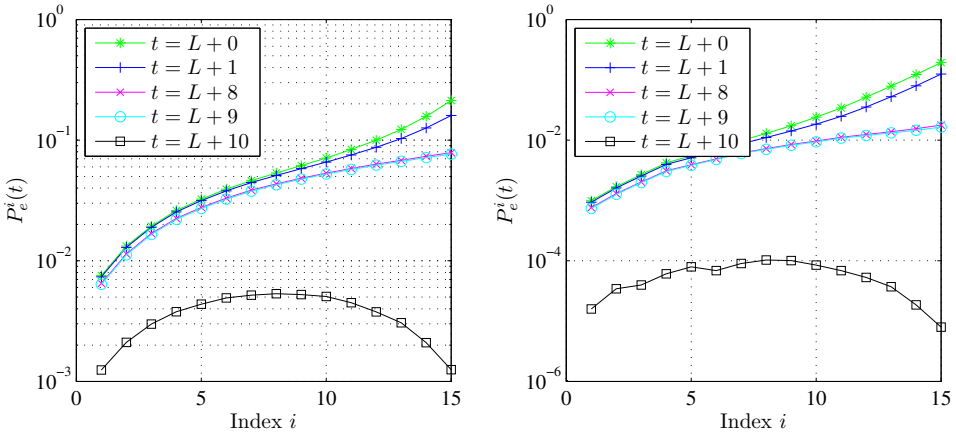


Figure 5.5: $P_e^i(t)$ for message indices $i = 1, \ldots, 5$ over time. $L = 15, m_s = 10, \tau = 11, k = 3, \epsilon = 0.35$.

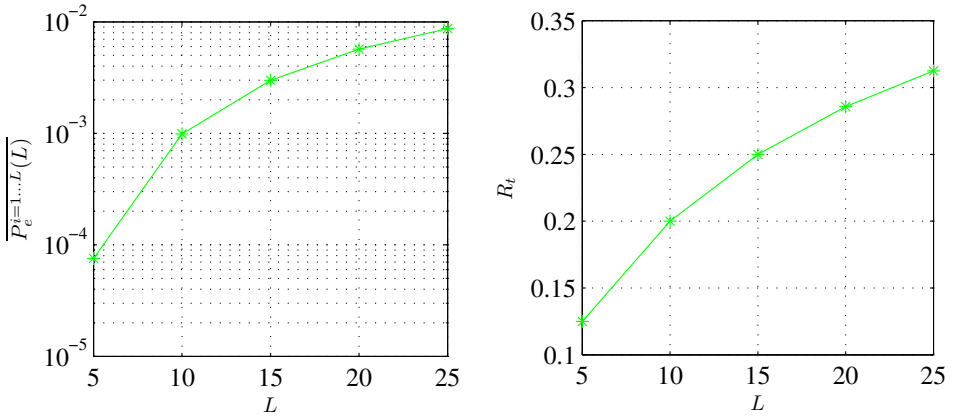
probability of decoding erasure at the end of the termination phase increases with increasing L while at the same time the rate R_t increases. We conclude that a very low decoding erasure probability P_e can be reached by terminating the code very frequently using a rather long termination length. This however reduces the rate R_t significantly. Note that a good improvement of $\overline{P_e^{i=1\ldots L}(L)}$ is only obtained when the termination interval length L is smaller than the termination length τ. In the context of anytime coding we are interested in an exponential improvement of the performance over time and index. A long delay until the performance is improved combined with no incremental improvement on the way violates the idea of anytime codes. Fig. 5.8 shows the probability of decoding erasure over time for all message blocks transmitted until termination at $L = 5$ for different m_s and thus different termination lengths $\tau = m_s + 1$. We can see that for the code with $m_s = 15$ during an interval of approximately 5 time steps there is virtually no decrease of the decoding erasure probability for any of the message blocks. But the probability



(a) Block length $k = 3$.

(b) Block length $k = 6$.

Figure 5.6: $P_e^i(t)$ for $t = L, \ldots, L+m_s$ where here $L = 15, m_s = 10, \tau = 11, \epsilon = 0.35$.

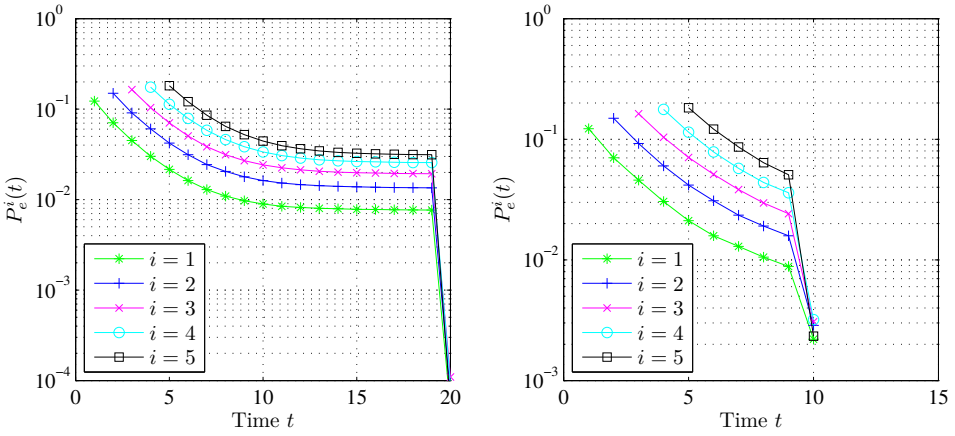


(a) Tradeoff $\overline{P_e^{i=1\ldots L}(L)}$ vs L, with $m_s = 15, \tau = 16$. $k = 3, \epsilon = 0.35$.

(b) Tradeoff R_t vs L, with $m_s = 15, \tau = 16$. $k = 3, \epsilon = 0.35$.

Figure 5.7: Probability of decoding error and corresponding code rate after termination R_t for different lengths of L.

of decoding erasure at the end of the termination phase is low. For the code with $m_s = 5$ there is an incremental decrease of the decoding erasure probability of all messages during the entire termination phase. However the decoding erasure probability at the end of the termination phase is not as low as for the code with $m_s = 15$. This shows that the more we back off from the anytime property the better the final decoding erasure probability gets at the end of the termination phase.



(a) $m_s = 15, \tau = 16, R_t = 0.125$. (b) $m_s = 5, \tau = 6, R_t = 0.25$.

Figure 5.8: Probability of decoding error over time for the messages $i = 1, \ldots, 5$ when terminating the code at $L = 5$ with different termination lengths $\tau = m_s + 1$. $k = 3, \epsilon = 0.35$.

5.2.2 Transmission over the AWGN Channel

Fig. 5.9 shows the performance of the terminated codes when transmitting over the AWGN channel with SNR = 3 dB. For $m_s = 10$ and $L = 15$ we illustrate how the performance over the index improves with the reception of a new block of bits coming from the termination phase. Very similar to the observations made for the BEC we see how the error probability decreases significantly when the final code block corresponding to the termination phase has arrived. In Fig. 5.10 we can see similar to Fig. 5.8 how the final error probability at the end of the termination phase is improved if we back off from the anytime property.

5.2.3 Conclusion

By terminating the anytime codes in regular time intervals we can significantly reduce the complexity of the encoder/decoder since both encoder and decoder only

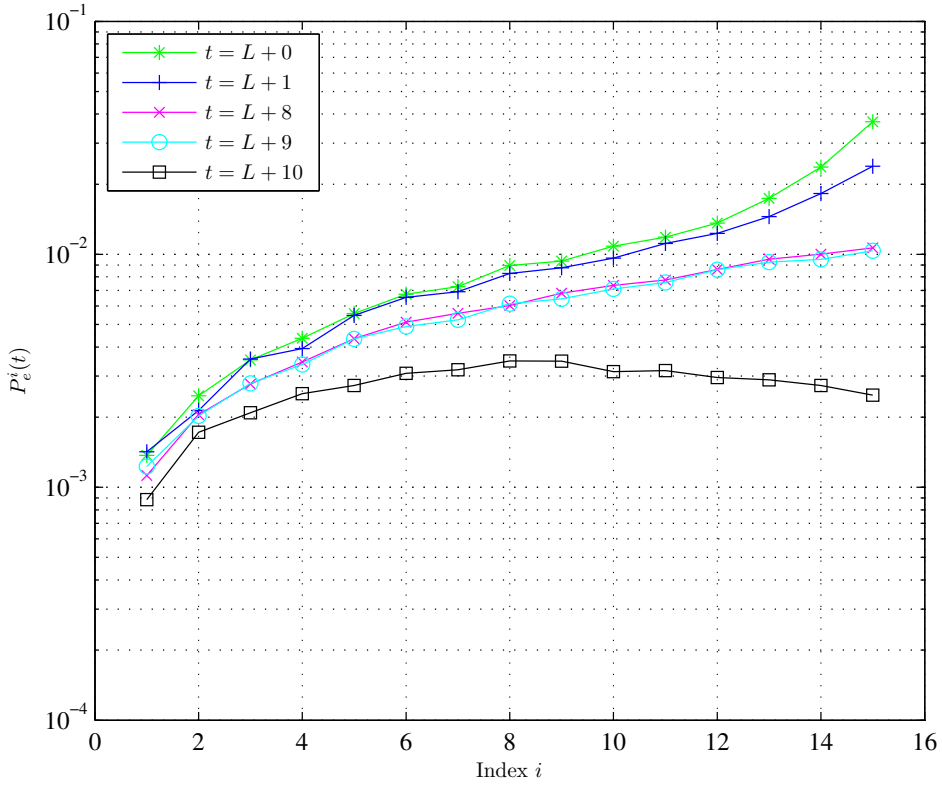
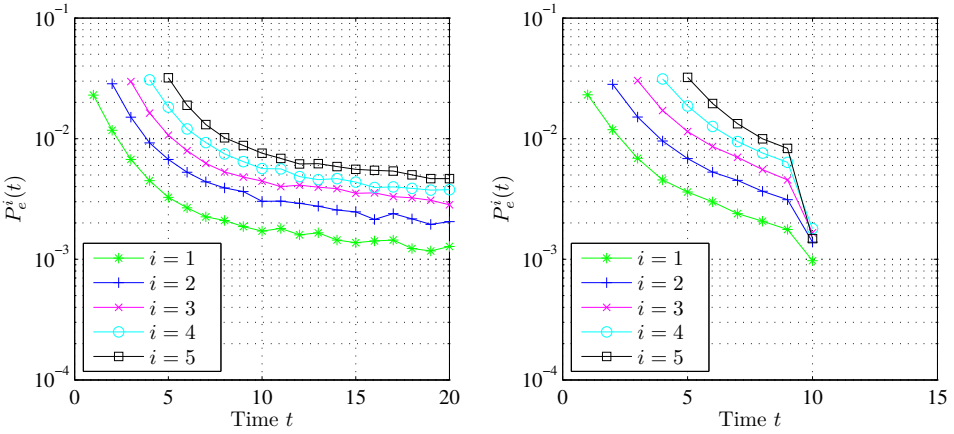


Figure 5.9: $P_e^i(t)$ for $t = L, \ldots, L+m_s$ where here $L = 15, m_s = 10, k = 3, \text{SNR} = 3$ dB. The final rate is equal to $R_t = 0.3$.

have to work on streams involving a fixed number L of messages/codewords. The final average probability of error per ensemble of L message blocks is severely reduced if the parameters m_s and L are chosen properly. This comes however at the expense of a rate reduction and by backing off from the anytime properties: The input bitstream has to be halted and the transmission of the termination bits is not incrementally improving the performance as we are aiming for with anytime codes. We conclude that in general terminating the codes is not suitable in the context of anytime coding.

5.3 Feedback Strategies

By introducing a feedback link from the decoder to the encoder we can both reduce the encoding and decoding complexities of the system and at the same time be able to detect and react on increasing error patterns. Below we present different ways



(a) $m_s = 15, \tau = 16, R_t = 0.125$.

(b) $m_s = 5, \tau = 16, R_t = 0.25$.

Figure 5.10: Probability of decoding error over time for the messages $i = 1, \ldots, 5$ when terminating the code at $L = 5$ with different $\tau = m_s + 1$. $k = 3, \text{SNR} = 3$ dB.

of using the feedback channel. All strategies proposed in the following require a feedback link and the possibility for the transmitter to temporarily transmit more bits than there are in a standard codeword. The latter property could instead be achieved by halting the input stream as it is done in the case where we terminate the code or by allocating additional bandwidth in the network. However, we choose to look at the decoding process from a different angle since in the following strategies we either keep the additional amount of data sent at one time instance very small or we take into account the delay obtained by transmitting a longer codeword. Note that the feedback is not used to inform the transmitter about whether a single transmission was successful or not. As typically done in the analysis of automatic-request-repeat (ARQ) systems, we assume that the feedback link is error-free. The effect of feedback errors in ARQ systems has been extensively investigated for noisy channels in [BEV14] and references therein. In general the feedback channel is better protected and thus provides a lower erasure probability. As long as the feedback erasure probability is of the order 10^{-2} or lower, the impact is minimal. For simplicity we make the assumption of an error-free feedback link.

5.3.1 Using Knowledge of Error Positions

The following subsection is based on [GRTS14] and therefore contains partly verbatim extracts of the paper. When transmitting over the BEC the decoder has at any point in time perfect knowledge about which decoded bits are still unknown. The exact position of the unknowns can be exploited to improve the performance of the code. Since the decoder of a system involving an AWGN channel does not have

this knowledge the following feedback strategy is only suitable when transmitting over the BEC. The knowledge about the exact positions of the unknowns is used in the following way: Whenever an increasing erasure pattern is detected (see Section 5.1.1 for how to detect an increasing erasure pattern) the decoder determines the smallest index j of an undecodable information bit (the undecodable information bit with the largest decoding delay) and informs the transmitter by feeding back the index. The size of the feedback messages depends on practical constraints on the maximum window size allowed. The feedback scheme is detailed in Algorithm 5.1. The feedback scheme results in an encoder and decoder pair with time varying

Algorithm 5.1 Feedback scheme for the BEC

Decoder side:

1: In every time step t the decoder determines the information stopping set size s_t of the overall bit stream $\tilde{\boldsymbol{y}}_{[1,t]}$.
2: If s_t exceeds a threshold $s_t > s^{th} = s^* \epsilon k$, the decoder flags the need for feedback.
3: The decoder determines the index j of the earliest undecodable information bit and feeds it back to the transmitter.

Encoder side:

1: In case the feedback link is activated at time t, then at time $t+1$ the transmitter sends both the current word $\boldsymbol{y}_{t+1}$ and the value of the unknown bit at index j over the channel.
2: The encoder updates the generator matrix to include only bits from the earliest unknown bit onwards. Accordingly the decoder updates its parity check matrix.
3: If the feedback link is inactive the encoder automatically shortens the generator matrix to include only bits up to an expected memory (m_{em}) of m_s. The decoder updates its parity check matrix accordingly.

encoding and decoding windows. Hence, the parity check matrix is no longer lower diagonal but band diagonal with a varying bandwidth. The expected memory m_{em} is a predetermined value that determines the size of the window in the parity check matrix that the encoder/decoder pair works on in case the feedback link is not activated. Therefore, if the encoder does not get a feedback request it can assume that bits that fall out of the expected memory decoding window are known. These bits are then no longer included in the encoding process in the following steps and the encoding/decoding window is shifted. However, if the feedback link is activated the encoding and decoding matrices grow and can get larger than the expected memory decoding window since the decoding window is increased such that an unknown bit cannot be erroneously purged from the decoding window. The choice of the expected memory m_{em} depends on k, ϵ and the threshold s^* as well as on the number of feedback bits allowed per channel transmission. The amount of feedback can be significantly reduced by increasing the block length k.

Finite-Length Behavior

Fig. 5.11 shows the finite-length average decoding performance $P_e^i(t)$ over time for blocks $i = 10, 14, \ldots, 26$, when feedback is introduced. Information blocks of size $k = 3$ bits are transmitted over a BEC with $\epsilon = 1/3$, and feedback is used as described above with a threshold of $s^* = 5$. The expected memory is set to $m_{em} = 10$. Comparing with Fig. 3.13 we can see that the code now has anytime properties. The decoding performance is independent of the position of the code block in the stream and the decoding error probability decays exponentially. An averaged feedback-request frequency of $f_{FB} = 0.027$ requests per channel transmission was measured over $T = 40$ transmissions. Here f_{FB} is defined as the frequency at which a feedback request is transmitted. The erasure pattern set size is closely

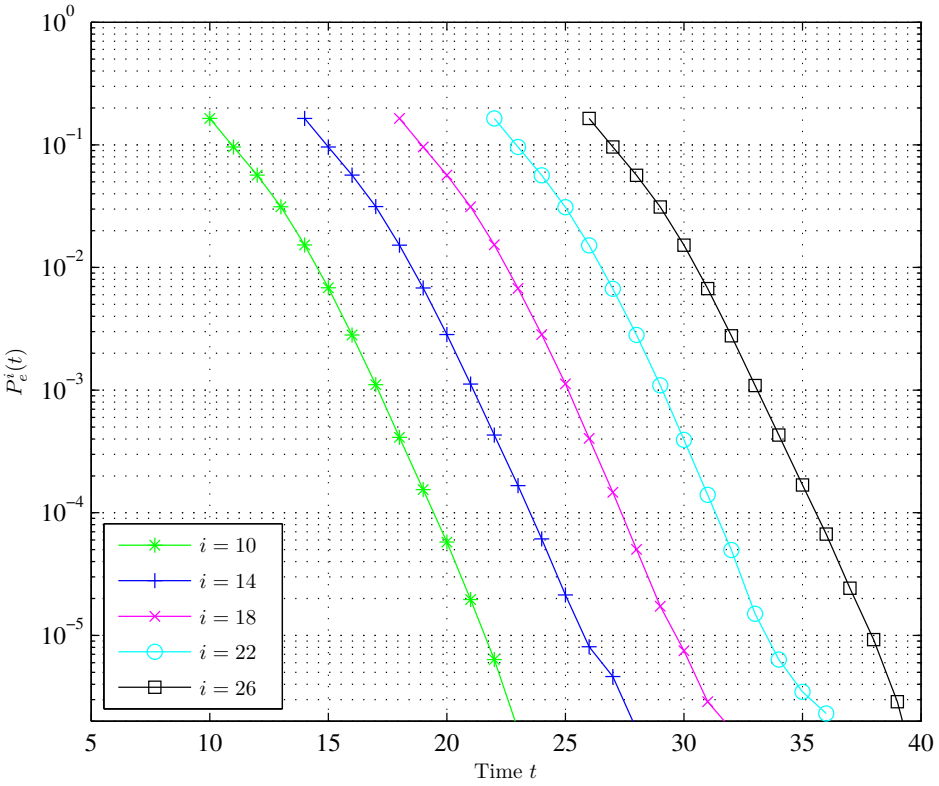
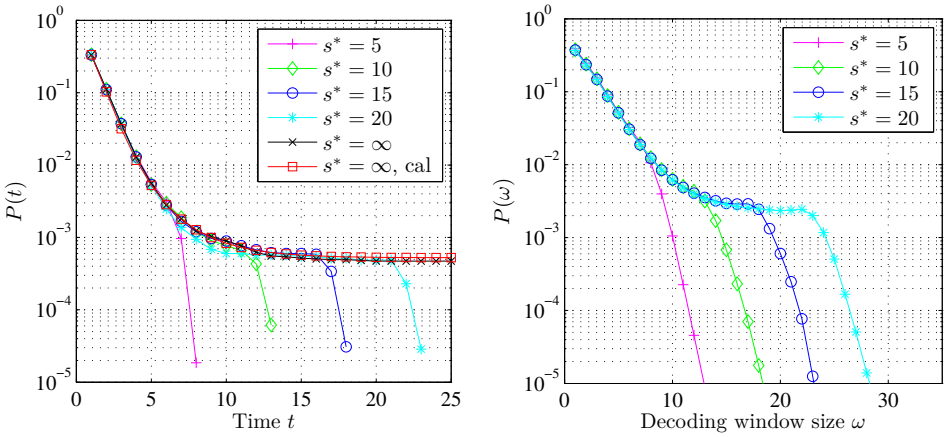


Figure 5.11: Finite-length average decoding performance P_e^i of the blocks with indices $i = 10, 14, \ldots 26$ when introducing feedback. Retransmission if $s_t > s^{th} = s^* \epsilon k$, where $s^* = 5$. Transmission over a BEC with $\epsilon = 1/3$. Information block length $k = 3$ bits. Perfect feedback with a measured average frequency of $f_{FB} = 0.027$ requests per channel transmission over $T = 40$ transmissions.

related to the decoding window size since the decoding window is always chosen such that all blocks containing erasures are included in the decoding window. That means as well that the probability $P(t)$ of having Et erasures at time t is closely related to the histogram of the decoding window size. Fig. 5.12a shows the simulated probability $P(t)$ for different thresholds s^* together with the approximated probability $P(t)$ for $s^{th} = \infty$ as shown earlier in Fig. 3.16a. As we can see the threshold determines how long the $P(t)$ follows the curve with infinite threshold. The probability of an erasure pattern set with size larger than the one corresponding to the threshold chosen decreases rapidly. The corresponding averaged feedback frequencies are shown in Fig. 5.13a as a function of s^*, and in Fig. 5.13b as a function of k. As expected the feedback frequency decreases with increasing threshold but saturates just as the probability $P(t)$. Fig. 5.12b shows the histogram of the decoding window size for the scenario considered in Fig. 5.12a. We can observe a very similar behavior for the decoding window size histogram as for the probability $P(t)$. That is, the probability of a window of size larger than the threshold s^* decreases rapidly. The decoding erasure probability over time follows a similar shape as $P(t)$ and the decoding window size. We can observe in Fig. 5.12b that

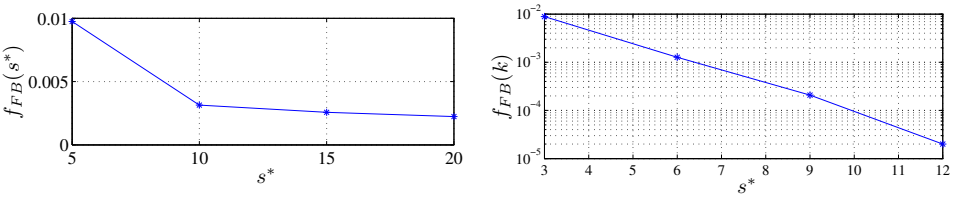


(a) Simulated and calculated $P(t)$.

(b) Decoding window histogram $P(w)$.

Figure 5.12: Probabilities when transmitting over the BEC with $\epsilon_s = 1/3$ using feedback with different thresholds $s^{th} = s^*\epsilon_s k$. Block length $k = 3$.

a smooth curve (without a plateau) can be obtained if we choose the threshold properly. In Fig 5.14a and Fig 5.14b we can further observe the effect of increasing k. Thus, by increasing k and choosing a proper threshold we can approach the asymptotic performance obtained without feedback. In order to achieve a smooth decay of the decoding error probability over time we need to choose the threshold such that $P(t)$ does not saturate since this would result in a similar plateau of the decoding erasure probability. A criterion for choosing the threshold could for



(a) Different thresholds s^{th} but same block length $k = 3$.

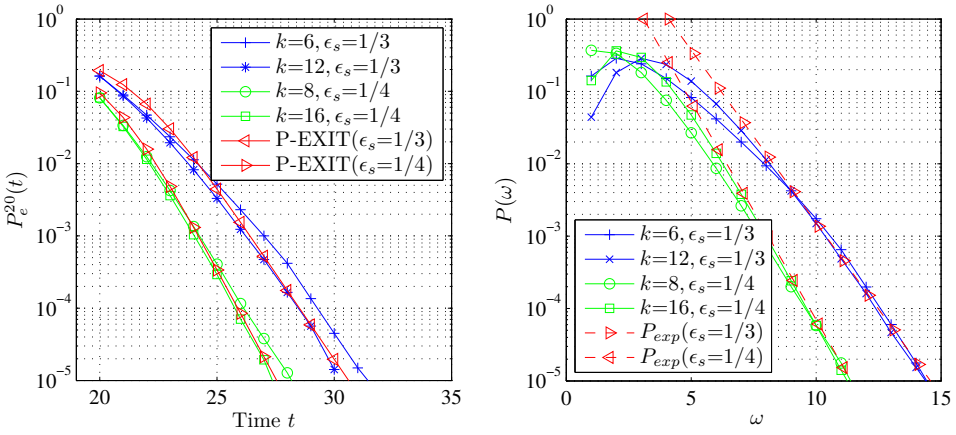
(b) Different block lengths, same feedback threshold $s^* = 5$.

Figure 5.13: Feedback frequencies for different thresholds s^{th} and block lengths k. Transmission over the BEC with $\epsilon_s = 1/3$.

example be to choose the turning point for s^*, as described in Section 3.2.4, corresponding to the time t where the curve $P(t)$ flattens out severely. Fig. 5.14a shows the decoding erasure probability of block $i = 20$ over time together with the curve obtained through asymptotic analysis (without feedback) for different k and ϵ_s. As we can see finite-length and asymptotic curve are very close. Fig. 5.14b shows the histogram $P(\omega)$ over the decoding window size ω when increasing the block length k but keeping the feedback threshold constant to $s^* = 5$ for all curves with $\epsilon = 1/3$ and to $s^* = 10$ for all curves with $\epsilon = 1/4$. We note that for large k the histograms converge to a common curve. This is because with increasing k the overall behavior is dominated by the average behavior and not by extreme cases such as receiving a block with no erasures at all or receiving a block containing only erasures. As we can see the probability of a large window decays rapidly. In fact the code with feedback now has anytime properties and the probability of a window of size ω decays exponentially over s from some point onwards. The error exponent is equal to the one determined in Section 3.2.1 as we can verify when comparing to the curves $P_{exp}(\epsilon)$ shown in the figure. The exponential decay of $P(\omega)$ indicates that the decoding complexity is significantly reduced using this kind of feedback strategy since large decoding windows become very unlikely. The feedback frequencies for different block lengths k are shown in Fig. 5.13b where we measured the fraction of times the feedback link was activated over a time horizon of $T = 80$ time instances. We see that we can decrease the feedback frequency by increasing the block length k while keeping the threshold constant. As seen before by adapting the threshold to k we can lower the feedback frequency even further; however note that at the same time this increases the complexity of the system. Based on the previous figures we can make an educated choice for the expected memory m_{em}. We can for example choose the memory m_{em} to be two times the size of the threshold chosen.

Conclusion

We have seen that with the preceding feedback strategy and a proper choice of the threshold, we can successfully eliminate increasing erasure patterns and achieve



(a) Decoding error performance of the 20th block over time.

(b) Probability $P(\omega)$ of a decoding window of size ω.

Figure 5.14: Transmission with feedback for different block lengths k. Feedback thresholds $s^*(\epsilon_s = 1/3) = 5$ and $s^*(\epsilon_s = 1/4) = 10$.

anytime behavior over all delays and all indices. The decoding complexity is significantly reduced since the probability of an encoding and decoding matrix of size ω is decreasing exponentially over ω with an error exponent equal to $\alpha_o = -\log_2(\epsilon)$. But note that even if the probability is decreasing exponentially it is still possible that the matrices grow large for a short time. The complexity might therefore on very rare occasions be very large. But if the system tolerates such cases we can achieve anytime behavior over all times with the help of this kind of feedback strategy.

5.3.2 Adapting the Rate of the Code

We have seen in Section 4.3 that decreasing the rate of the code reduces the probability of an increasing error event. While we do not want to transmit permanently at a lower rate we now want to investigate the effect of temporarily decreasing the rate in order to potentially resolve an increasing error pattern. The rate of the code is then variable and the change of the rate is done on the fly based on feedback provided by the decoder. The resulting parity check matrix of the code is then time varying. Furthermore we introduce an element of randomness in the code construction. We propose two different strategies on how to set the rate of the code. The first strategy is for transmission over the BEC only. The second strategy is suitable for transmission over either the BEC or the AWGN channel. The way the code is constructed is the same for both strategies. We therefore first explain the code construction depending on the current rate R_t and all previous rates R_i.

Thereafter we devise the two different strategies on how these rates are chosen.

Code Construction

The message length of the variable rate coding scheme is constant equal to k for all messages $\boldsymbol{x}_i$ but the codeword length n_i of codeword $\boldsymbol{y}_i$ is variable and depends on the rate $R_i = k/n_i$ assigned to the i-th message. Given that the rate is changing over time so is the protograph, that is

$$\boldsymbol{B}_{[1,t]} = \begin{bmatrix} \boldsymbol{B}_0(1) & & & & & \\ \boldsymbol{B}_1(2) & \boldsymbol{B}_0(2) & & & & \\ \boldsymbol{B}_2(3) & \boldsymbol{B}_1(3) & \boldsymbol{B}_0(3) & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ \boldsymbol{B}_{t-1}(t) & \boldsymbol{B}_{t-2}(t) & \dots & \boldsymbol{B}_1(t) & \boldsymbol{B}_0(t) \end{bmatrix}, \tag{5.6}$$

where the matrix $\boldsymbol{B}_0(t)$ above depends on the rate R_t assigned to the current message only, with a basic structure given as

$$\boldsymbol{B}_0(t) = \begin{bmatrix} 1 & 1 & & & & \\ 1 & 0 & 1 & & & \\ 1 & 0 & 0 & 1 & & \\ \vdots & \vdots & & & \ddots & \\ 1 & 0 & \dots & \dots & 0 & 1 \end{bmatrix} \tag{5.7}$$

and the size equal to $1/R_t - 1 \times 1/R_t$. The size of the matrices $\boldsymbol{B}_i(t)$ with $i > 0$ depends on the rate R_i assigned to message $\boldsymbol{x}_i$ and the rate R_t assigned to the current message $\boldsymbol{x}_t$: Their basic structure is partly random given as

$$\boldsymbol{B}_i(t) = \begin{bmatrix} p_{i,t} & 0 & 0 & \dots \\ p_{i,t} & 0 & 0 & \dots \\ p_{i,t} & 0 & 0 & \dots \\ \vdots & \vdots & & \end{bmatrix} \tag{5.8}$$

where $p_{i,t}$ indicates that the corresponding entry is equal to 1 with probability $p_{i,t}$. The size of $\boldsymbol{B}_i(t)$ depends on R_i and R_t and is given as $(1/R_i - 1) \times 1/R_t$. We choose to make the probability $p_{i,t}$ that the first column in the submatrix $\boldsymbol{B}_i(t)$ is equal to 1, rate depended by setting

$$p_{i,t} = R_t/\gamma, \tag{5.9}$$

where $\gamma \geq 1$ is an integer. With this choice of $p_{i,t}$ the variable nodes corresponding to a message $\boldsymbol{x}_i$ encoded with a very high rate are strongly connected to the variable nodes of the current message $\boldsymbol{x}_t$; and the variable nodes corresponding to a message $\boldsymbol{x}_i$ encoded with a low rate are loosely connected to the variable nodes corresponding to the current message $\boldsymbol{x}_t$. The parameter γ allows for further modification. We now explain two different feedback protocols that make use of the above code structure.

Strategy 1: Transmission over the BEC

The first algorithm is applicable to transmission over the BEC only since it involves knowing the exact number of unknowns in the received bitstream. The feedback algorithm is given in Algorithm 5.2. The motivation for this structure is that since

Algorithm 5.2 Feedback scheme for the BEC

Decoder side:

1: In every time step t the decoder determines the number of unknown information bits s_t in the overall bit stream $\tilde{\boldsymbol{y}}_{[1,t]}$ received up to time t.
2: Depending on the value of s_t, the decoder sends a request for changing the rate.
3: If $s_t \geq is^*\epsilon k$ and $s_t < (i+1)s^*\epsilon k$ the rate is set to $R_{t+1} = \frac{k}{(i+2)k}$.
4: If $s_t < s^*\epsilon k$ the codeword length n_t is decreased to $n_t = \min(2k, n_{t-1} - k$ meaning that the rate is increased to $R_{t+1} = \min(0.5, \frac{1}{(1/R_t - 1)})$.

Encoder side:

1: The encoding/decoding matrix is modified according to the requested rate of the decoder.

according to the feedback protocol the rate is high only if the number of erasures is low, the most recent message can be safely reconnected to many previous messages with a low risk of contributing to an increasing erasure pattern. If the number of erasures is high, the risk of contributing to an increasing erasure pattern is high and the current message is therefore only reconnected to a lower number of previous messages, thus limiting the number of cycles introduced into the graph. Note that as for the original structure only information variable nodes are reconnected. We now investigate the performance of such a rate adaptive code design.

Finite-Length Analysis

In order to address the problem analytically we restrict ourselves to the transmission over the static BEC with erasure rate ϵ_s, that means *exactly* $E = \epsilon_s k$ erasures are introduced by the channel in a block of k bits. Fig. 5.15 shows the simulated probability $P(t)$ of the probability of an erasure pattern containing exactly Et unknown information bits when transmitting over a static BEC with erasure rate $\epsilon_s = 3/4$ and $k = 4$ for different values of the threshold s^*. We can see that there is a drop in the probability $P(t)$ whenever a multiple of the threshold is reached and the rate is lowered. The way the probability $P(t)$ decays depends strongly on the choice of the threshold s^*. In other words, by decreasing the threshold s^* we can achieve a larger drop of the probability $P(t)$. It is therefore advantageous to have a small threshold s^*. Note however that for a small s^* the feedback link is activated very often. With the proposed feedback scheme we cannot entirely remove increasing erasure patterns. This is because even with a code with a very

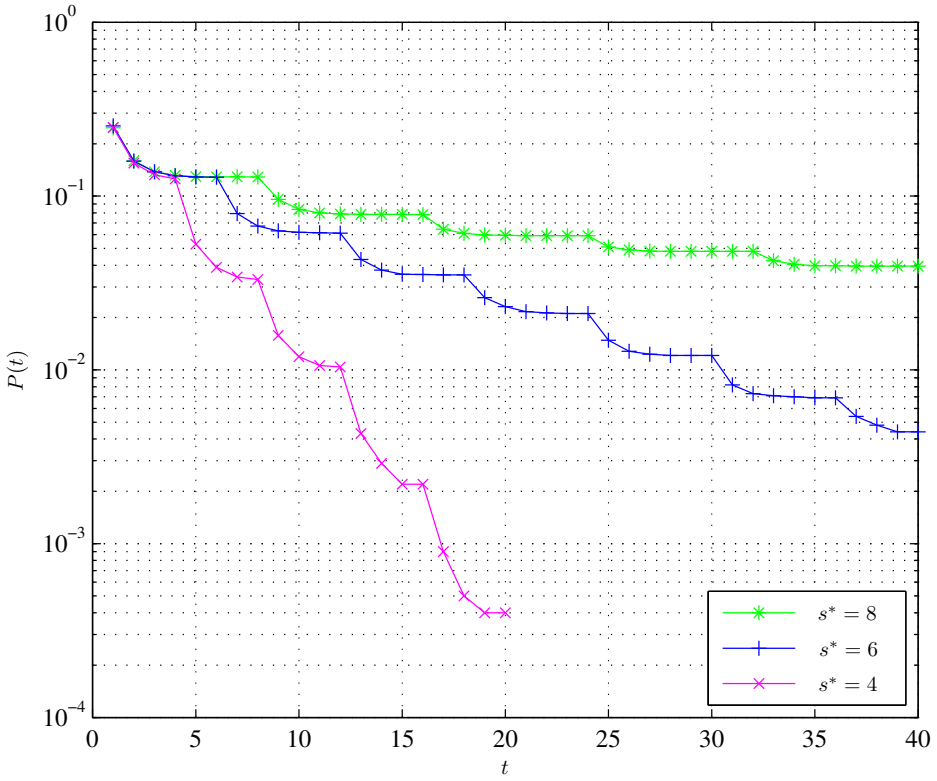


Figure 5.15: Simulated probability $P(t)$ for $\epsilon_s = 3/4, k = 4$ and different s^*. Feedback strategy according to Algorithm 5.2: Rate decrease whenever the number of unknowns exceeds multiples of the thresholds $s^{th} = \epsilon_s k s^*$.

low rate we encounter the problem of increasing erasure patterns. However, we can see in Fig. 5.15 that by decreasing the rate when the number of erasures exceeds a small threshold we can reduce the probability of occurrence of an increasing erasure pattern. We now derive an approximation of the probability $P(t)$ of having an erasure event of size Et at time t. The probability $P(t)$ depends on the value of s^* and on the value of $p_{i,t} = p_t = R_t/\gamma$ (here we only analyze $\gamma = 1$). For $t \in [0...s^*]$ is the probability $P(t)$ the same as for the original case. That is

$$P(t) = P(0) \prod_{i=0}^{t \leq s^*} \left[1 - i\epsilon_s(1-\epsilon_s)^{i-1}\right]^{k(1-\epsilon)} . \tag{5.10}$$

In order to calculate the probability $P(t)$ for $t \in [s^* + 1, \ldots, 2s^*]$ we use the results obtained in Section 4.2 on protographs with reduced degrees and the results obtained in Section 4.3 for lower rate codes. Denote as in Subsection 3.2.3

by n_t^ℓ the number of erased information bits connected to check node C_t^ℓ, where $C_t^1, \ldots, C_t^{(1/R_t-1)k}$ are the new check nodes introduced at time t. Since $\boldsymbol{B}_0(t)$ is deterministic but $\boldsymbol{B}_0(t)$ partly random, a check node C_t^ℓ has deterministic connections to exactly $(t - \lfloor t/s^* \rfloor s^*)$ information nodes and additionally $(\lfloor t/s^* \rfloor s^*)$ possible connections coming from the random part, where a connection exists with probability p_t. Check node C_t^ℓ has then on average $\bar{c}_t = (t - \lfloor t/s^* \rfloor s^* + p_t \lfloor t/s^* \rfloor s^*)$ connections. The probability that only known bits are connected to C_t^ℓ except for exactly one connected erasure, is then given similar to (3.38) as

$$\Pr\{n_t^\ell = 1 | s_{t-1} = E(t-1)\} = \bar{c}_t \epsilon_s \left(1 - \epsilon_s\right)^{\bar{c}_t - 1}. \tag{5.11}$$

Note that due to the variable rate the number of check nodes is now rate dependent and equal to $(1/R_t - 1)k$. Under the assumption that the variables n_t^ℓ are independent, the probability $P(t|t-1)$ is given as

$$P(t|t-1) \approx \prod_{l=1}^{(k-E)(1/R_t-1)} \Pr\{n_t^l \neq 1 | s_{t-1} = E(t-1)\} \tag{5.12}$$

$$\approx \left(1 - \bar{c}_t \epsilon_s \left(1 - \epsilon_s\right)^{\bar{c}_t - 1}\right)^{(1/R_t-1)(k-E)}. \tag{5.13}$$

The probability $P(t)$ is then given recursively as $P(t) = P(t-1)P(t|t-1)$, where $P(1)$ is given in (3.46). The resulting analytic term is an approximation of the actual performance due to the assumption of independence and due to the averaging over the number of information bits connected to a check node. Fig. 5.16 shows the accuracy for the approximation for two different cases. As can be seen the approximation is good for small erasure patterns and gets increasingly inaccurate for large patterns. While we cannot expect the variable rate code to solve the problem of increasing erasure patterns entirely, we can however show that the probability of an increasing erasure pattern is significantly reduced by allowing rate adaptations. Moreover we conjecture that an adaptive code can keep up anytime properties with higher probability than a fixed rate code.

Simulations

In Fig. 5.17 we show the performance in terms of the decoding erasure probability $P_e^i(t)$, of the variable rate code with the feedback strategy given in Algorithm 5.2 when transmitting over a BEC with $\epsilon = 0.6$ using a message length of $k = 5$. In the same plot we show the performance of a fixed rate anytime code with a rate equal to the average rate of the variable rate code. In order to make a fair comparison we show the performance over time in bits. Thus we compare the codes in steps where on average the same number of bits was transmitted over the channel for the two different codes. We can see in Fig. 5.17 that the fixed rate code is not suitable in this case since the decoding erasure probability of all messages shown quickly emerges into an erasure floor. This is due to the poor channel quality combined

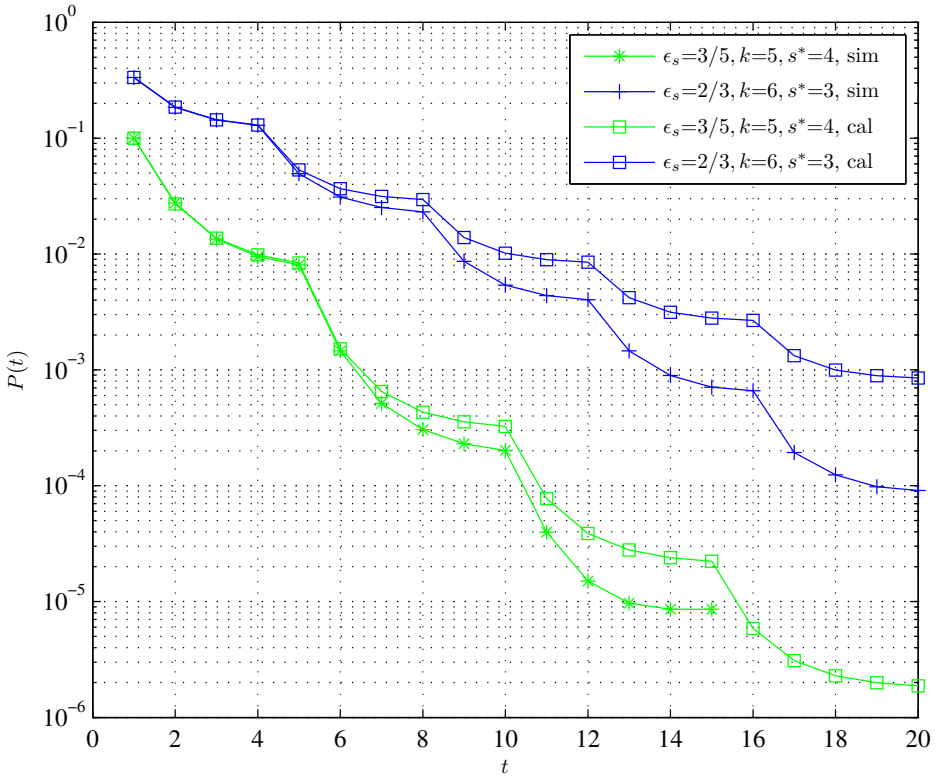


Figure 5.16: Simulated probability $P(t)$ for different ϵ_s, k and s^*. Feedback strategy according to Algorithm 5.2: Rate decrease whenever the number of unknowns exceeds multiples of the thresholds $s^{th} = \epsilon_s k s^*$.

with a short block length. The variable rate code however shows a significantly better decay of the decoding erasure probability of the messages considered. The variable rate code is therefore more suitable.

We conclude that the possibility to adapt the rate of the anytime code on-the-fly significantly improves the performance. The feedback scheme proposed here is to some extent analytically describable. It is however only applicable to transmission over the BEC. The next strategy that we propose can be used for both transmission over the BEC and the AWGN channel.

Strategy 2: Transmission Over the BEC or the AWGN Channel

The feedback protocol described in the previous subsection requires the transmission of the desired rate R_{t+1} for the next message $\boldsymbol{x}_{t+1}$ on the feedback link. This

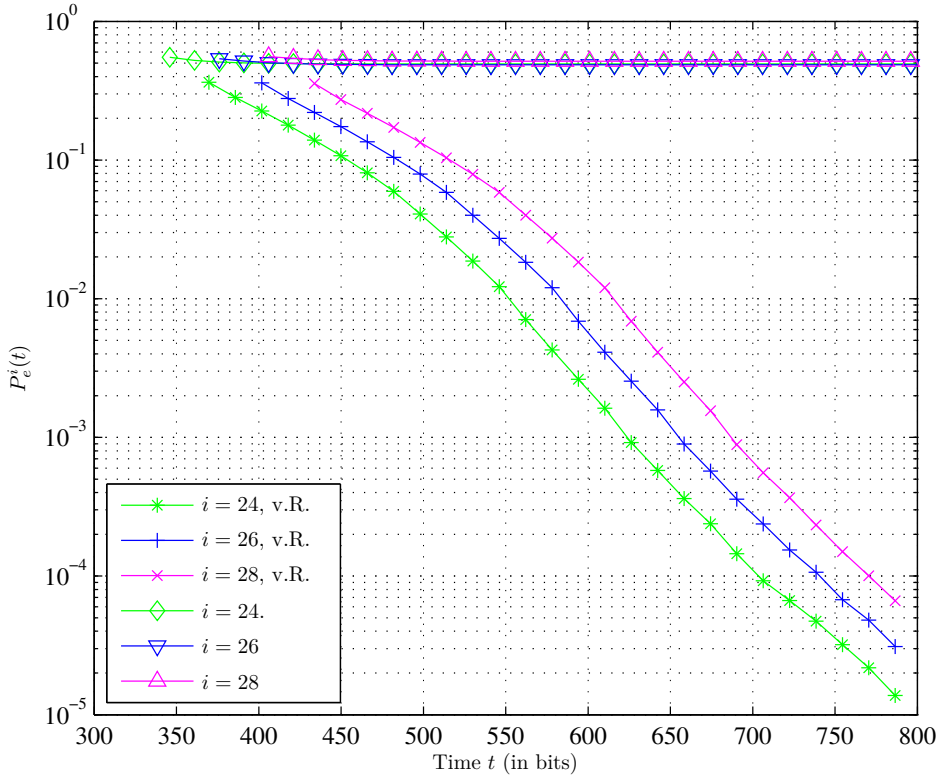


Figure 5.17: Behavior over time for $\epsilon = 0.6, k = 5$. Comparison of a variable rate code with $s^* = 4$ and an average final rate of $\bar{R} = 0.31$ (after $T = 60$ time steps) with a fixed rate code with $R = 1/3$. Feedback strategy according to Algorithm 5.2. "v.R." stands for variable rate.

involves transmitting several bits. An alternative feedback strategy, that only involves the transmission of a single bit, is given in Algorithm 5.3. This strategy works as well for transmission over the AWGN channel since the threshold test does not depend on the exact number of unknowns in the received stream.

Finite-Length Analysis

Recall that the finite-length analysis presented in Section 3.2.3 simplifies the actual situation by assuming that an error event grows constantly. That is, we neglect the cases where the number of errors increases, then stops increasing or even decreases and then increases again. With the feedback strategy described in Algorithm 5.3 this situation is the most common. A finite-length analysis for the BEC along

Algorithm 5.3 Feedback scheme for the BEC and the AWGN

Decoder side:

1: In every time step t an error event detector checks the occurrence of an increasing error event.
2: If an increasing error event is detected, the decoder sends a request to decrease the rate.
3: If no increasing error event is detected, the decoder sends a request to increase the rate.

Encoder side:

1: The encoding/decoding matrix is modified according to the request of the decoder.
2: If the request is to decrease the rate, the rate changes from $R_t = k/n_t$ to $R_{t+1} = k/(n_t + k)$.
3: If the request is to increase the rate, the rate changes from $R_t = k/n_t$ to $R_{t+1} = \min(0.5, k/(n_t - k))$.

the lines of (3.45) is therefore not very accurate. We do however show through simulations that the proposed feedback scheme has excellent performance over time.

Simulations

We now simulate the performance of the codes over time with the feedback strategy given in Algorithm 5.3 when transmitting over either the BEC or the AWGN channel. Fig. 5.18 shows the performance of the variable rate anytime code with threshold $s^* = 5$ together with the performance obtained when transmitting with a fixed rate anytime code. The rate of the fixed rate anytime code is chosen to $R = 1/3$ which is approximately equal to the average rate ($\bar{R} = 0.38$) obtained with the variable rate anytime code during a transmission of $T = 100$ messages. The performance is shown in terms of the decoding erasure probability $P_e^i(t)$ for different messages i over the number of transmitted bits when transmitting over a BEC with $\epsilon = 0.6$ using a message length of $k = 5$ bits. For the variable rate anytime code the average number of transmitted bits per time step is chosen. As we can see the fixed rate anytime code does not have anytime properties and quickly looses its error correction capabilities. The decoding erasure probability emerges quickly into an erasure floor. The variable rate anytime code on the other hand has an exponential decay of the decoding error probability at least down to 10^{-5}. Moreover the decoding error performance is independent of the message index i. Fig. 5.19 shows a similar plot when transmitting over the AWGN channel with SNR = 3dB. Here we show the decoding error probability $P_e^i(t)$ for different messages i using a variable or fixed rate code with message length $k = 6$ bits. We can see that with the variable rate anytime code we can achieve anytime behavior at an SNR where the fixed rate anytime code transmitting the same number of bits

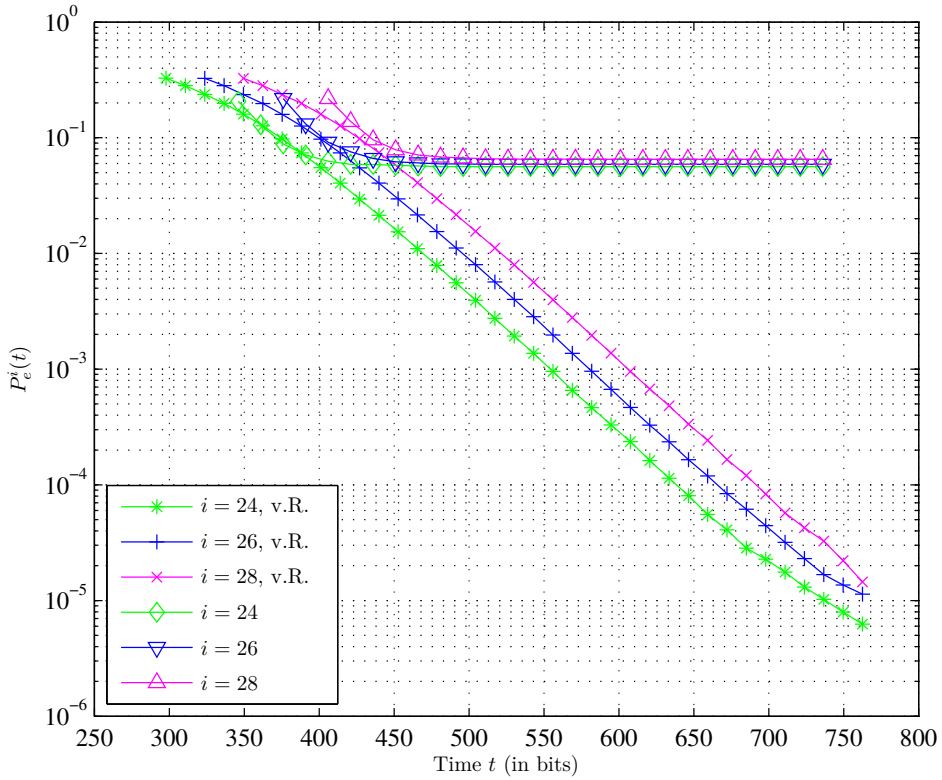


Figure 5.18: Decoding erasure probability $P_e^i(t)$ over time for messages $i = 24, 26, 28$ when transmitting over a BEC with $\epsilon = 0.5$ using either a fixed rate code with $R = 1/3$ or a variable rate code with feedback strategy according to Algorithm 5.3 and $s^* = 5, \gamma = 2$. Message length $k = 5$. The variable rate code has a measured rate of $\bar{R} = 0.38$.

over the channel as the variable rate anytime code on average, is quickly emerging into an increasing error floor. We conclude that when transmitting over the BEC or the AWGN channel, the possibility of adaptively changing the rate of the code improves the decoding error performance significantly.

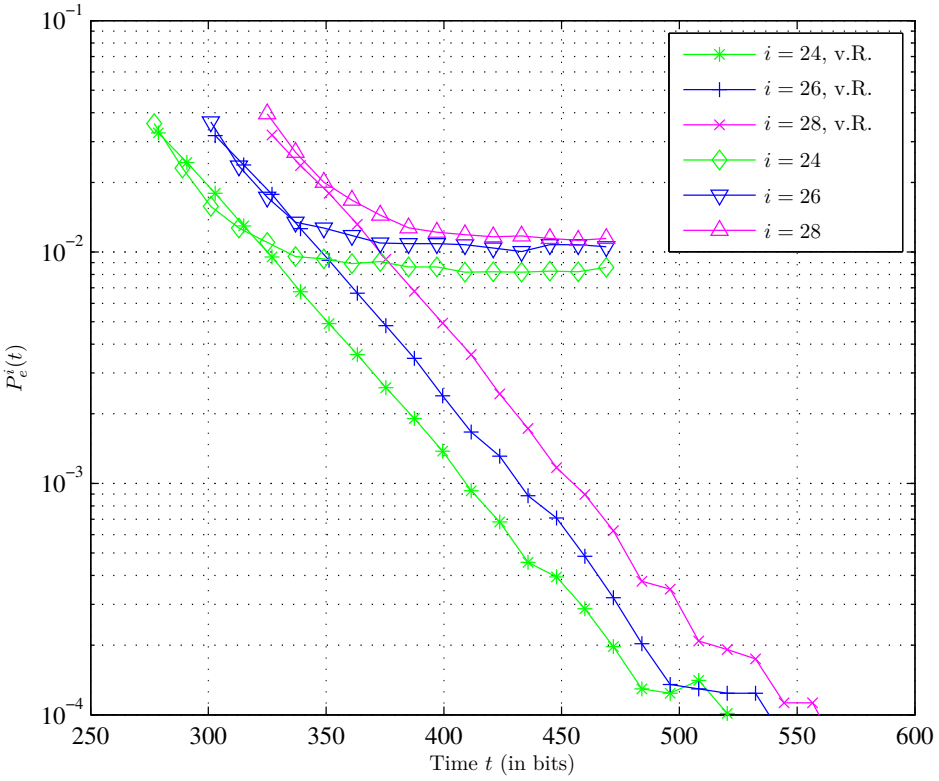


Figure 5.19: Behavior over time for SNR $= 3$dB, $k = 6$. Comparison of a variable rate (v.R.) code with $\rho_{\%} = 0.5, d^{th} = 7$ with a fixed rate code. The average rate of the variable rate code after $T = 50$ time steps is $\bar{R} = 0.49$. The fixed rate code has a rate of $R = 1/2$. Feedback strategy according to Algorithm 5.3.

Chapter 6

Applications of Anytime Codes in Automatic Control

In this chapter we apply the codes developed in the previous chapters in order to protect the messages when stabilizing an unstable plant over a noisy channel. The main focus of this chapter is therefore on the control context as highlighted in Fig. 6.1. We compare the performance of the overall system when using anytime codes with the performance obtained when using block codes. As the performance index we choose the variance of the observed signal. We demonstrate the advantage of the anytime codes over block codes for the case when only a very limited number of bits can be transmitted over the channel per system update. The plant and observer that we consider here have certain characteristics such that the control problem is Linear-Quadratic-Gaussian (LQG). After a brief explanation of the overall system setup we describe the system components in more detail for the case when block codes are used for error protection. In the subsequent section the system components are specified for the case when anytime codes are used for error control. The main differences lie in the way the observations are quantized and reconstructed, how the encoder makes use of the information given by the observer, and how the state estimator is designed. In the analysis and performance evaluation we derive for both the transmission scheme using block codes and the scheme using anytime codes the cost function and a constraint on the number of quantization bits. In the last section we show the benefit from using anytime codes as compared to block codes. The results illustrate the relation between the parameters such as available resources, dynamics of the system, resolution of the quantizer and the code rate. Due to analytical tractability we only consider transmission over the BEC. While in anytime theory the signal to be transmitted is often seen as a stream of bits, in control theory bits are often grouped into packets. Instead of loosing single bits a data loss is then the loss of an entire packet. To bridge this gap we use results from [PPV10] to convert bit erasure rates from the BEC considered in communication theory, into packet erasure rates when given the erasure rate of the underlying BEC. Note that in this chapter we only provide a qualitative comparison of using

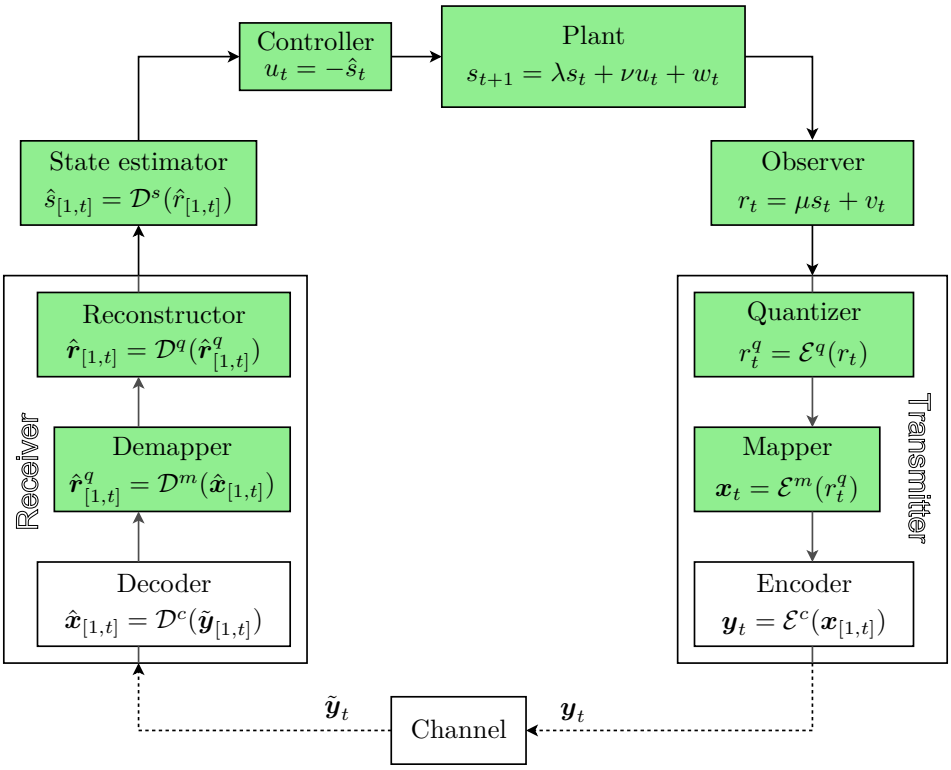


Figure 6.1: System overview with focus on the control problem.

either anytime codes or block codes, due to various approximations and assumptions made.

6.1 System Overview

Fig. 6.1 depicts the overall system setup. Here we briefly describe the different system components in the diagram. A more detailed description is given later since the implementation of the system components depends on the type of channel code used.

Plant and Observer

We consider a plant, modeled as a discrete-time, scalar LTI system where both process and measurement noise are modelled as additive white Gaussian noise. The state update equation and observer state are given by:

$$s_{t+1} = \lambda s_t + \nu u_t + w_t \tag{6.1}$$

$$r_t = \mu s_t + v_t, \tag{6.2}$$

where s_t is the state of the system, r_t is the observed signal and u_t is the control signal. The additive terms w_t and v_t are independent discrete-time Gaussian white noise processes representing process and measurement noise. Both are zero-mean with variance σ_w^2 and σ_v^2, respectively. The variables λ, ν and μ are the state, input and output coefficients. The plant is unstable if for the variable λ we have $|\lambda| > 1$. Without loss of generality, we let $\mu = \nu = 1$ for the sake of a more compact notation. We only consider scalar variables. The problem of controlling a plant as described above is frequently studied in control theory and termed the LQG problem. Whereas the plant description is (almost) independent of the channel code we want to use, the way the observed signal is processed is very much dependent on the type of channel code used. When using a block code the differences in observation processing result in the formulation of an *effective* state update equation.

Quantizer and Mapper

The signal r_t observed by the observer is a real value that in order to be processed needs to be quantized. This is done by the quantizer. The quantization level r_t^q that the observed signal r_t is quantized to is mapped into a binary vector $\boldsymbol{x}_t$. This is the message that we want to transmit over the channel. The quantizer depends on the channel code used.

Encoder

The message $\boldsymbol{x}_t$ is subject to the encoder. Whether all previous messages $\boldsymbol{x}_{[1,t-1]}$ are relevant when determining the codeword $\boldsymbol{y}_t = \mathcal{E}^c(\boldsymbol{x}_{[1,t]})$ depends on the type of channel code used. The modulation scheme applied in this thesis is Binary-Phase-Shift-Keying (BPSK) independent of the channel code used. Note that there is no direct feedback from the decoder to the encoder. The encoder has therefore no direct knowledge about whether a transmission was successful or not.

Channel

In this chapter we only consider transmission over the BEC with erasure rate ϵ. We assume that the channel is such that only a very limited number n_{max} of bits can be transmitted over the channel per time step t. This could for instance be the case when there are many other processes close by that are controlled simultaneously and every system is assigned a very narrow bandwidth only.

Decoder

In every time step the decoder tries to decode the codeword $\tilde{\boldsymbol{y}}_t$. Whether all other previous codewords are decoded as well depends on the channel code used. Thus we obtain an estimate of the current message $\hat{\boldsymbol{x}}_t$ and potentially all previous messages as well.

Demapper and Reconstructor

The demapper performs the inverse operation of the mapper and converts all binary messages $\hat{\boldsymbol{x}}_{[1,t]}$ back into an integer $\hat{\boldsymbol{r}}_{[1,t]}$. The demapping from $\hat{\boldsymbol{x}}_i$ to r_i^q is only possible if the estimated message $\hat{\boldsymbol{x}}_i$ does not contain any erasures. For all messages $\hat{\boldsymbol{x}}_i$ where the demapper is able to find the corresponding value r_i^q the reconstructor tries to determine the real value $\hat{r}_i$. Whether this is possible or not depends on the channel code used.

Estimator and Controller

Based on the reconstructed value(s) the state estimator estimates depending on the channel code used either only the current state $\hat{s}_t$ or as well all previous states $\hat{s}_{[1,t-1]}$. The control signal u_t is set to the value $-\hat{s}_t$ if an estimate of the state s_t is available. Otherwise $u_t = 0$.

Performance Evaluation Index

In order to evaluate the performance of the system involving both the control components and the communication components we consider the output variance as performance index

$$J = \lim_{t\to\infty} \sup \mathbb{E}[r_t^2], \tag{6.3}$$

already introduced in Section 2.3. The LQG problem is then the problem of finding the control signal for a plant given in (6.2) that minimizes the quadratic cost function J subject to the constraint that u_t only depends (strictly) causally on the output signal $r_{[1,t-1]}$ and possibly on its previous values $u_{[1,t-1]}$.

6.2 Transmission with Block Codes

Given the basic system setup described in the previous section we now detail the implementation of the system components when using a block code for error protection.

6.2.1 Observer

Anytime codes and block codes are designed for different purposes: An anytime code is supposed to capture the dynamics of a quickly evolving plant with an increasing reliability over delay. A block code is supposed to protect the messages separately such that each message can be recovered from its corresponding codeword with very high reliability even when the codeword is corrupted. For this to work well a block code needs a sufficiently large block length. Since we consider the scenario where only a very limited number of bits n_{max} can be transmitted per observed state, in order to make a fair comparison we cannot simply use the block code in the same way as the anytime code. This would result in a codeword which is far too short.

Instead we only process a fraction of the observed states and can thus transmit a longer codeword by spreading the transmission over several time steps. The different processing schemes are illustrated in Fig. 6.2. Whereas for the anytime code every

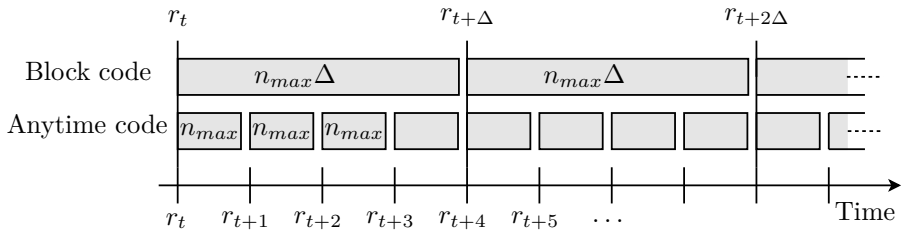


Figure 6.2: Observation processing for block code and anytime code. An observation r_t is made in every time step but for the block code only every Δ-th observation is processed. Here $\Delta = 4$. Between two time steps a maximum of n_{max} bits can be transmitted over the channel.

observation r_t is processed, quantized and encoded into a codeword of length n_{max}, for the block code only every Δ-th observation is processed. All other observations are discarded. In this way we are able to transmit a codeword containing $n_{max}\Delta$ bits with the block code. By omitting observations the transmitter has a coarser view on the plant. This can be described with an *effective* state update equation as seen by the transmitter.

6.2.2 Effective State Update Equations for the Block Code

We have seen in the previous section that by summarizing several blocks of size n_{max} into one codeword we can increase the length of the overall codeword. A new codeword is determined every Δ-th time step based on the current observed state. The message containing the information about the current state is encoded using a block code with a message length of k_{BC} and a codeword length of Δn_t bits. Since only n_{max} bits can be transmitted over the channel during one time instance, it takes Δ time steps to transmit the entire codeword.

The resulting *effective* state update equation for the block code is derived as follows: Starting with the state s_t at time t the state update equation when $u_t = 0$ can be written as follows

$$s_{t+i} = \lambda^i s_t + \sum_{j=0}^{i-1} \lambda^j w_{t+i-j-1}. \tag{6.4}$$

Since for the block code a control signal can only be applied every Δ time steps the update equation is given as

$$s_{t+\Delta} = \lambda^\Delta s_t + u_t + \tilde{w}_{t+\Delta}, \tag{6.5}$$

where $\tilde{w}_{t+\Delta} = \sum_{i=0}^{\Delta-1} \lambda^i w_{t+\Delta-i-1}$, and the observed signal is given as

$$r_{t+\Delta} = s_{t+\Delta} + v_{t+\Delta}. \tag{6.6}$$

In the following we will use the terms $x_{t+\Delta}, v_{t+\Delta}, \tilde{w}_{t+\Delta}, u_{t+\Delta}$ and $y_{t+\Delta}$ when we talk about the variables updated in time steps of Δ. The measurement noise $v_{t+\Delta}$ has the same distribution as v_t, namely $v_{t+\Delta} \sim \mathcal{N}(0, \sigma_v^2)$. The effective system noise $\tilde{w}_{t+\Delta}$ is zero-mean but with a variance given as

$$\mathbb{E}[\tilde{w}_{t+\Delta}^2] = \sum_{i=0}^{\Delta-1} \lambda^{2i} \sigma_w^2. \tag{6.7}$$

6.2.3 Quantizer and Mapper

Since packets can be lost as no retransmission is foreseen, a predictive quantizer (or any other quantizer with memory) would end up in an uncertain state once a packet is lost. Instead, we consider quantization of the absolute state such that the state is known (within the resolution of the quantizer) once the corresponding packet is received. Furthermore, for this choice of quantizer recovering outdated states will not improve the system performance. We will later see that this is not the case when using an anytime code. For simplicity, the quantizer that we apply here is a symmetric uniform quantizer with dynamic range $-q_{max} \dots q_{max}$. It should be noted that we do not try to optimize the quantizer here other than by choosing proper limits according to the number of quantization bits that we are allowed to use. The observed signal $r_{t+\Delta}$ that is quantized is not uniformly distributed. Using a uniform quantizer for non-uniform data is surely not optimal but makes the analysis easier. Changing the quantizer to e.g. a logarithmic quantizer should obviously improve the performance but it does not change the general point we are trying to make when comparing block and anytime codes in this context.

In order to find the proper limits of the quantizer and in order to determine the signal-to-quantization-noise ratio (SQNR) we need to make an assumption on the distribution of $s_{t+\Delta}$. Given that $u_t = 0$, we can see from (6.5) that both $s_{t+\Delta}$ and $r_{t+\Delta}$ are Gaussian distributed with zero mean. Denote the variance of $r_{t+\Delta}$ by $\sigma_{r+\Delta}^2$. When setting the upper and lower limit $-q_{max}$ and q_{max} of the uniform quantizer we make use of the fact that for a Gaussian distributed variable with variance σ^2 we have 99.7 percent of the area of the probability density function contained between $\pm 3\sigma$. That is, if we choose

$$q_{max} = \frac{1}{2} q_{99.7} = 3\sigma_{y+\Delta}, \tag{6.8}$$

then 99.7 of all values that $r_{t+\Delta}$ takes are within the interval $[-\frac{q_{99.7}}{2}, +\frac{q_{99.7}}{2}]$. Given this interval and the number of quantization bits k_{BC}, we can determine the size of the quantization interval by dividing the range by the number of distinct quantization levels describable with k_{BC} bits:

$$\delta = \frac{q_{99.7}}{2^{k_{BC}}} = \frac{6\sigma_{y+\Delta}}{2^{k_{BC}}}, \tag{6.9}$$

where we have inserted (6.8). The operation of the quantizer $r_t^q = \mathcal{E}^q(r_t)$ is then given as

$$r_t^q = -q_{max} + \frac{\left(i+\frac{1}{2}\right) q_{max}}{2^{k_{BC}-1}} \text{ if } r_t \in \left[-q_{max}+\frac{iq_{max}}{2^{k_{BC}-1}}, -q_{max}+\frac{(i+1)q_{max}}{2^{k_{BC}-1}}\right), \tag{6.10}$$

where i is the quantization index. After quantization the quantized signal is subject to the mapper. The binary message $\boldsymbol{x}_t$ is obtained from the integer i by simply converting i to its binary representation. The demapper performs the inverse operation. Note that since the block code operates on messages separately, the demapper only demaps a single message, that is $\hat{r}_{t+\Delta}^q = \mathcal{D}^m(\hat{x}_{t+\Delta})$. Similarly the reconstructor only reconstructs a single value: $\hat{r}_{t+\Delta} = \mathcal{D}^q(r_{t+\Delta}^q)$.

SQNR

Now using the assumption that the quantization error (denoted by n_t) is uniformly distributed, for the quantization noise we can calculate $\mathbb{E}[n_t^2] = \delta^2/12$, and we can determine the SQNR to be

$$\text{SQNR} = \frac{\mathbb{E}[r_{t+\Delta}^2]}{\mathbb{E}[n_t^2]} = \frac{(2^{k_{BC}})^2}{3}. \tag{6.11}$$

Note that for this choice of quantizer the SQNR only depends on the number of quantization bits.

6.2.4 Encoder and Decoder

We use a standard block encoder and decoder for error protection of the message $\boldsymbol{x}_t$ when transmitting over the BEC with erasure rate ϵ_{BEC}. Instead of employing a specific block code we use general results on finite-length block codes obtained in [PPV10]. From Theorem 37, in [PPV10] we know that for the BEC with erasure probability ϵ_{BEC} there exists an $(n, M, \epsilon_{packet})$ block code (maximal probability of error) such that

$$\epsilon_{packet} \leq \sum_{i=0}^{n} \binom{n}{i} \epsilon_{BEC}^i (1-\epsilon_{BEC})^{n-i} 2^{-[n-i-\log_2(M-1)]^+} \tag{6.12}$$

is fulfilled. Here n is the block length and $M = 2^{Rn}$ is the number of codewords. The result holds for block lengths above approximately 100 bits. Using this result we can now determine an upper bound on the probability ϵ_{packet} that a block codeword is dropped.

6.2.5 State Estimator and Controller

For the design of the state estimator and the controller we make use of the results obtained in [CLSZ13]. Here the authors investigate the LQG control problem as described in Subsection 6.1 in a networked control system with a realistic channel. The channel model takes into account packet loss, signal quantization, code rate limitations and delay. Fig. 6.3 shows the setup they consider. Packets of arbitrary size are transmitted over the channel. Packet erasures are modeled by a Bernoulli process $\gamma_t \in \{0, 1\}$. A packet that is sent over the channel is either received correctly ($\gamma_t = 1$) or it is dropped ($\gamma_t = 0$). A packet drop occurs with probability ϵ_{packet} at each packet transmission, independently of previous events. Accepted packets are correct with very high probability. Based on the information available the state estimator then produces an estimate $\hat{\xi}_t = [\hat{x}_{t-d+1}, \ldots, \hat{x}_t]^T$ which is fed to the state feedback generator which sets the control signal $u_t = L\hat{\xi}_t$, where L is a variable. The variable n_t in Fig. 6.3 represents the quantization noise. If the resolution of the quantization is sufficiently fine, n_t can be modeled as a zero-mean additive random process with identically distributed uncorrelated samples of power $\sigma_n^2 = \mathbb{E}[n_t^2]$. The channel model furthermore takes into account a transmission/processing delay of d time steps. The code rate limitation is considered by an upper bound on the SQNR $\alpha = \mathbb{E}[r_t^2]/\sigma_n^2$. For this setup the authors propose a simple (constant gain) estimator where they can derive the following condition for which the optimal value J^* of the cost is finite:

$$\frac{1 - \epsilon_{packet}}{1 + \frac{\lambda^{2d}}{\alpha}} > 1 - \frac{1}{\lambda^2}. \tag{6.13}$$

Moreover, the optimal value of L is found to be $L = -\lambda$. The constant gain estimator is further detailed below.

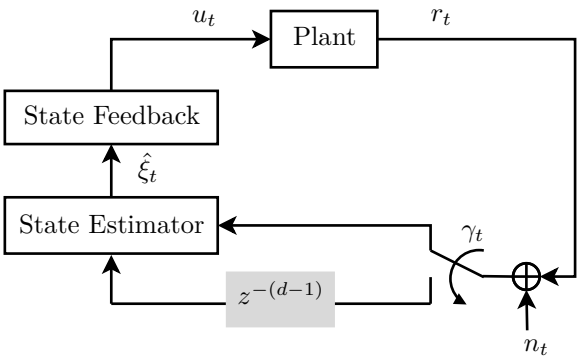


Figure 6.3: Networked control scheme as in [CLSZ13]. The delay unit $z^{-(d-1)}$ is gray shaded since we do only consider the setup for $d = 1$.

Processing and transmission delays for the transmission with block codes are in our model reflected by the parameter Δ in the *effective* state update equations.

Investigating additional delays is out of the scope of this thesis. We therefore use the results in [CLSZ13] for $d = 1$: The constant gain estimator has then the following form

$$\hat{x}_t = \hat{x}_{t-1} + u_{t-1} + \gamma_{t-1} G(\gamma_{t-1}(r_{t-1} + n_{t-1}) - \hat{x}_{t-1}), \tag{6.14}$$

where G is the predictor gain. The estimator is motivated as follows: If a packet is lost, i.e. $\gamma_t = 0$, then the estimator updates its state using the model only. If a packet is received correctly, i.e. $\gamma_t = 1$, the estimate is adjusted by a correction term, based on the output innovation, similarly to a Kalman filter. This scheme does not yield the optimal Kalman filter; it allows however for an explicit computation of the performance index J. The optimal value G^* is the solution of a set of coupled Riccati equations. The details are omitted here. Note that the channel in the model considered in Fig. 6.3 is a packet erasure channel and not the BEC.

6.3 Transmission with Anytime Codes

Given the basic system setup in Fig. 6.1 we now detail the implementation of the system components when using an anytime code for error protection. The plant usually considered in anytime theory is slightly different from the plant considered in control theory. The difference lies in the process and measurement noise variables w_t and v_t. In anytime theory these processes are assumed to be *bounded*, i.e. $||w_t||_\infty < \frac{W}{2}$ and $||v_t||_\infty < \frac{V}{2}$. Without the boundedness it is not clear how one can stabilize the system with anytime codes without additional assumptions on the channel [SH11c]. By choosing W and V large compared to σ_w and σ_v the practical importance of this constraint is however very low. In the simulations we therefore use unbounded noise processes.

6.3.1 Observer

We have seen that an anytime code is capable of working on potentially very small block lengths. We therefore try to capture the dynamics of the system at the same pace as the system evolves. The observation processing was already explained in Subsection 6.2.1. As depicted in Fig. 6.2 the anytime code processes all observations r_t.

6.3.2 Quantizer and Mapper

When using an anytime code for error-protection we typically do not use a simple uniform quantizer as we did when using block codes. The reason for this is that the goal when using the anytime structure is to be able to track the plant no matter what value r_t takes. The anytime property implies however as well that (even though with a very low probability) a reliable estimate of a particular state might be available only with a very long delay. During this time the system is uncontrolled and the system state could grow out of the limits of a e.g. uniform quantizer. A

quantizer that is able to quantize any value $r_t \in \mathbb{R}$ is the δ-lattice based quantizer. A δ-lattice based quantizer [SM06] is a map that maps an input r_t to an integer i. Fig. 6.4 depicts the mapping. Each of the bins is of size δ and a value r_t that falls into one of these bins is assigned to the center of the bin. The quantizer is L-regularly-labeled if there are L bin labels that repeat periodically. When using k quantization bits we have $L = 2^k$. The quantize operation is then given as

$$r_t^q = \lfloor r_t/\delta \rfloor \mod 2^k. \tag{6.15}$$

The binary message $\boldsymbol{x}_t$ is obtained from the integer r_t^q by simply converting r_t^q to its binary representation. The demapper $\mathcal{D}^m(\hat{\boldsymbol{x}}_{[1,t]})$ performs the inverse operation for all messages that are decodable.

Due to the periodicity of the quantizer, in order to reconstruct a value, we need to know in which segment of the quantizer the value r_t^q lies. We specify the segment by an offset o_t. The reconstructed value $\hat{r}_t$ is then obtained as

$$\hat{r}_t = \hat{r}_t^q \delta + \delta/2 + \hat{o}_t. \tag{6.16}$$

Given $\hat{r}_t^q$ we know that $r_t \in \{\hat{r}_t \pm \delta/2\}$. Note that the offset o_t is not transmitted over the channel, instead, it has to be inferred from all previous values $\hat{r}_{[1,t]}^q$. This is done together with the state estimation which is explained in detail later. Under

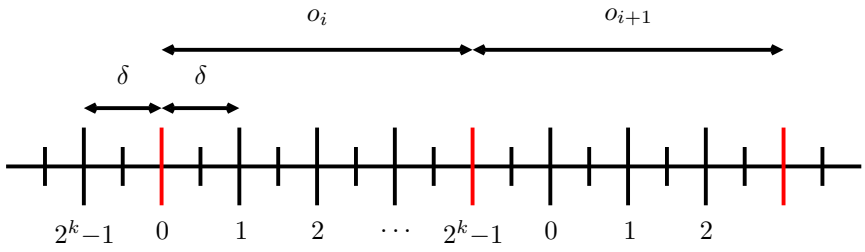


Figure 6.4: L-regularly-labeled δ-based quantizer with $L = 2^k$.

the assumption that the offset o_t is known, the quantizer is a uniform quantizer with bin width δ and the quantization noise power is therefore given as $\sigma_n^2 = \delta^2/12$.

6.3.3 Encoder and Decoder

The message $\boldsymbol{x}_t$ is encoded with an anytime code $\mathcal{E}^c(\boldsymbol{x}_{[1,t]})$ and the resulting codeword $\boldsymbol{y}_t$ is transmitted over the channel. The encoding and decoding process of an anytime code have been described in previous chapters. We use a systematic LDPC convolutional anytime code as proposed earlier with different rates.

6.3.4 State Estimator and Controller

Since the anytime decoder provides improved message estimates $\hat{\boldsymbol{x}}_{[1,t]}$ in every time step t, the state estimator uses the information on *all* reconstructed values to

update the estimates of *all* states seen so far. In the following we will see that the state estimator is built on the assumption that the noise processes v_t and w_t are bounded. The boundaries assumed in the simulations are $W = 6\sigma_w$ and $V = 6\sigma_v$ (the values of W and V are chosen such that 99.7% of all values are within the bounds). For all decodable messages $\hat{\boldsymbol{x}}_i$ the reconstructor can find the value $\hat{r}_i^q$. In order to obtain a real number $\hat{r}_i = \hat{r}_i^q\delta + \delta/2 + \hat{o}_i$ however we need to know the offset $\hat{o}_i$. The offset is not transmitted over the channel. We can infer the offset by investigating the range of possible values the signal $\hat{r}_i^q$ can take. This is done using a hypercuboidal filter [SH11c]. In the case of scalar measurements a hypercuboidal filter is just a region of the form $x \in \{\mathcal{R}|a \leq x \leq b\}$. With this filter the receiver bounds the set of all possible states that are consistent with the estimates of the quantized measurements. We assume that the initial state s_0 has bounded support. Starting from this we can determine the range of possible values of all states $s_i \in [\hat{s}_i^{min}, \hat{s}_i^{max}]$ and observations $r_i \in [\hat{r}_i^{min}, \hat{r}_i^{max}]$, the so-called uncertainty intervals, by iterating upper and lower bounds on s_i and r_i through the state update equation and measurement equation. An estimate $\hat{s}_i$ of the state or its observation $\hat{r}_i$ is then the middle of its possible range

$$\hat{s}_i = \frac{1}{2}(\hat{s}_i^{min} + \hat{s}_i^{max}) \tag{6.17}$$

$$\hat{r}_i = \frac{1}{2}(\hat{r}_i^{min} + \hat{r}_i^{max}). \tag{6.18}$$

The offset $\hat{o}_i$ can be determined accurately whenever the possible range $[\hat{r}_i^{min}, \hat{r}_i^{max}]$ of values for $\hat{r}_i$ lies *within* the range of L quantization intervals corresponding to the *same* offset value of the quantizer (see Fig. 6.4 for the quantizer). If this is the case then

$$\hat{o}_i = \left\lceil \frac{(\hat{r}_i^{min} - r_i^q\delta - \delta/2)}{2^k\delta} \right\rceil 2^k\delta. \tag{6.19}$$

Given the offset $\hat{o}_i$ the state estimate $\hat{s}_i$ is updated to $\hat{s}_i = \hat{r}_i$. Note that the estimate $\hat{r}_i = r_i^q\delta + \delta/2 + \hat{o}_i$ is known with an uncertainty corresponding to the quantization error only. If the offset is known the uncertainty interval therefore shrinks dramatically. For messages where the corresponding offset is not known the accuracy of the estimate depends on how far in the past the message lies which offset is known. This is further detailed below. First we study the case where none of the messages up to time i is decodable. Then

$$\hat{r}_i^{q,min} = \hat{s}_i^{min} - \frac{V}{2}, \tag{6.20}$$

where

$$\hat{s}_i^{min} = \lambda\hat{s}_{i-1}^{min} - \frac{W}{2}. \tag{6.21}$$

Here we assumed that the control signal $u_t = 0$. Similarly,

$$\hat{r}_i^{q,max} = \hat{s}_i^{max} + \frac{V}{2}, \tag{6.22}$$

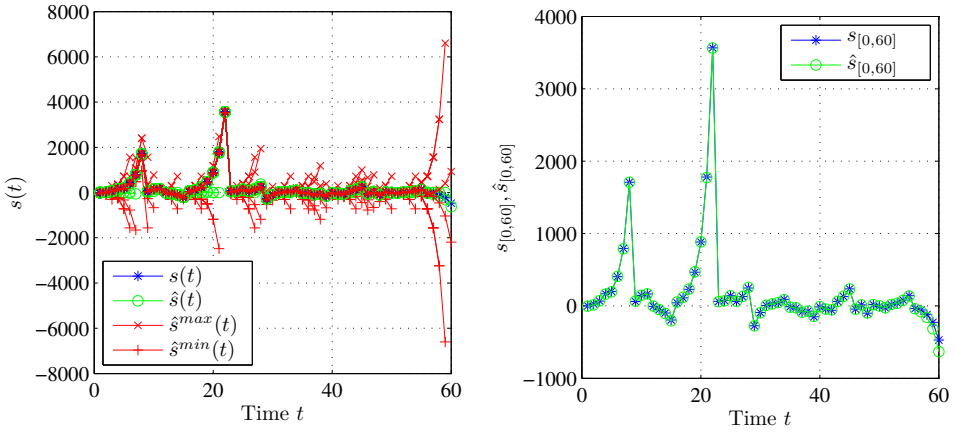
where

$$\hat{s}_i^{max} = \lambda \hat{s}_{i-1}^{max} + \frac{W}{2}. \tag{6.23}$$

We can see from these equations that the uncertainty range grows exponentially. Moreover, if $\hat{s}_0^{min} = -\hat{s}_0^{max}$ then the uncertainty interval grows symmetrically, and consequently, for all estimates $\hat{s}_i$, we have $\hat{s}_i = 0$, and therefore the control signal u_t is equal to 0. This means that for the case where none of the received messages is decodable the system evolves uncontrolled. Now assume that the first message $\boldsymbol{x}_1$ is decodable. Then if the bounds of s_0 are known the offset $\hat{o}_1$ can be determined accurately. If the quantizer is properly designed and the noise processes are assumed to be truncated we can determine the offset $\hat{o}_2$ as soon as the message $\boldsymbol{x}_2$ is decodable, and so forth. The state estimate $\hat{s}_t$ is therefore as good as the last state estimate $\hat{s}_i$ where the offset is known. The control signal is given as $u_t = -\hat{s}_t$. In Fig. 6.5 we show a snapshot of a simulation at time $t = 60$ when controlling an unstable plant with $\lambda = 2$ (process noise standard deviation $\sigma_v = 5$) that is observed in noise (measurement noise standard deviation $\sigma_w = 15$). The observed measurement is quantized using $k_{AC} = 20$ bits and transmitted over a BEC with $\epsilon = 0.3$ using an $R = 1/2$-anytime code. The state estimator and controller follow the process described above. We can see that the plant is stabilized. In Fig. 6.5a the evolution of the state $s(t)$ is shown together with the evolution of the state estimate $\hat{s}(t)$ and the uncertainty range $[s_i^{min}, s_i^{max}]$ for $0 \leq i \leq t$. We can see how the uncertainty range temporarily grows exponentially until it suddenly shrinks dramatically again. Fig. 6.5b shows the state $s_{[0,t]}$ and the estimated state $\hat{s}_{[0,t]}$ at time $t = 60$. We confirm that the estimated state is very close to the actual state. Only for small delays with respect to the current time $t = 60$ the estimated state $\hat{s}(t)$ is not yet very close to the state $s(t)$. The intervals where $s(t)$ is growing exponentially correspond to the times where the state estimate was only known with high uncertainty. With sufficient delay the decoder and state estimator are however capable to estimate the state properly.

6.4 Analysis and Performance Evaluation for Block Codes

After having specified the different ways the entire system is designed when using either a block code or an anytime code we now want to evaluate the performance. In this section we analyze the system by determining the number of quantization bits necessary to stabilize the plant and the control cost function when using block codes for error protection. Section 6.5 is concerned with the performance evaluation when transmitting using an anytime code. Given the observation processing scheme for block codes depicted in Fig. 6.2 and the effective state update equation (6.5) we now find constraints on the maximum packet erasure rate, the BEC erasure rate and the resolution of the quantization necessary to be able to control the plant.



(a) State $s(t)$ and estimated state $\hat{s}(t)$ evolution over time together with the uncertainty range.

(b) Snapshot of state $s_{[0,t]}$ and estimated state $\hat{s}_{[0,t]}$ at time $t = 60$.

Figure 6.5: Snapshot of state evolution over time for an unstable system with $\lambda = 2, \sigma_v = 5, \sigma_w = 15$ when using an anytime code with $R = 0.5, k = 20$ for error-protection over a BEC with $\epsilon = 0.3$.

6.4.1 Constraint on the Erasure Rate

Applying the result from [CLSZ13] given in (6.13) and taking into account the *effective* state update equation given in (6.5) yields the following result on the packet erasure rate ϵ_{packet}: The optimal value J^* of the control cost is finite if and only if

$$\frac{1-\epsilon_{packet}}{1+\frac{\lambda^{2\Delta}}{SQNR}} > 1 - \frac{1}{\lambda^{2\Delta}}. \tag{6.24}$$

Inserting $SQNR = (2^{k_{BC}})^2/3$ as given in (6.11) we get

$$\epsilon_{packet} < \frac{\frac{(2^{k_{BC}})^2}{3} + \lambda^{2\Delta} - \lambda^{4\Delta}}{\lambda^2 \frac{(2^{k_{BC}})^2}{3}}. \tag{6.25}$$

Furthermore by using (6.12) with equality we can determine the relation for the BEC erasure rate ϵ_{BEC} instead of the packet erasure rate ϵ_{packet}:

$$\sum_{i=0}^{n_{BC}} \binom{n_{BC}}{i} \epsilon_{BEC}^i (1-\epsilon_{BEC})^{n_{BC}-i} 2^{-[n_{BC}-i-\log_2(2^{k_{BC}}-1)]^+} < \frac{\frac{(2^{k_{BC}})^2}{3} + \lambda^{2\Delta} - \lambda^{4\Delta}}{\lambda^2 \frac{(2^{k_{BC}})^2}{3}}. \tag{6.26}$$

Note that since (6.12) is an upper bound we might be able to stabilize plants even for higher erasure rates ϵ_{BEC}. Note moreover that (6.12) is a theoretical result

about the best possible block code. It does not provide us with a suitable block code.

With the above inequality we can judge whether there exists a (n_{BC}, k_{BC}) block code that we can use to encode observations every Δ time steps and transmit over a BEC with erasure rate ϵ_{BEC} and achieve a finite optimal control cost J^* when trying to stabilize a plant with parameter λ using the cheap estimator given in [CLSZ13]. This inequality is used later when determining the parameters of the block code.

6.4.2 Constraint on the Number of Quantization Bits

From (6.25) we can conclude a constraint on the number of quantization bits that is independent of the erasure rate: In order for the packet erasure rate to be positive we need

$$\frac{(2^{k_{BC}})^2}{3} + \lambda^{2\Delta} - \lambda^{4\Delta} > 0, \tag{6.27}$$

which is equivalent to

$$k_{BC} > \frac{1}{2}\log_2(3(\lambda^{4\Delta} - \lambda^{2\Delta})). \tag{6.28}$$

This inequality is used later when determining the parameters of the block code when comparing anytime codes with block codes.

6.4.3 Cost Function

We now derive the cost function $J = \lim_{t\to\infty} \mathbb{E}[r_t^2]$. As described in Section 6.2 the decoder tries to decode the received codeword every Δ time steps. If the codeword cannot be decoded perfectly it is discarded and the control signal is set equal to 0. If the codeword can be decoded perfectly a control signal is applied. Assume that if the packet is not discarded based on the received codeword the controller can control the system with an error denoted by $\mathbb{E}[x_{e0}^2]$. From (6.5) we can obtain $\mathbb{E}[s_{t+\Delta}^2]$ depending on how many packets are lost.

$$\mathbb{E}[s_{t+\Delta}^2|1 \text{ packet lost}] = \lambda^{4\Delta}\mathbb{E}[x_{e0}^2] + (1 + \lambda^{2\Delta})\sigma_{\tilde{w}}^2) \tag{6.29}$$

$$\mathbb{E}[s_{t+\Delta}^2|2 \text{ packets lost}] = \lambda^{6\Delta}\mathbb{E}[x_{e0}^2] + (1 + \lambda^{2\Delta} + \lambda^{4\Delta})\sigma_{\tilde{w}}^2) \tag{6.30}$$

$$\dots$$

$$\mathbb{E}[s_{t+\Delta}^2|i \text{ packets lost}] = \lambda^{(i+1)2\Delta}\mathbb{E}[x_{e0}^2] + \sum_{j=0}^{2i} \lambda^{j\Delta}\sigma_{\tilde{w}}^2) \tag{6.31}$$

where $\sigma_{\tilde{w}}^2 = \sum_{i=0}^{\Delta-1} \lambda^{2i}\sigma_w^2$. The probability that i packets are lost in a row is equal to ϵ_{packet}^i. As soon as a packet is received correctly we have

$$\mathbb{E}[s_t^2|0 \text{ packets lost}] = \lambda^{2\Delta}\mathbb{E}[x_{e0}^2] + \sigma_{\tilde{w}}^2. \tag{6.32}$$

This happens with probability $1-\epsilon_{packet}$. It follows that

$$\mathbb{E}[s_t^2] = (1-\epsilon_{packet}\mathbb{E}[s_t^2|0 \text{ packet lost}] + \sum_{j=1}^{\infty} \mathbb{E}[s_{t+\Delta}^2|j \text{ packets lost}]\epsilon_{packet}^j. \quad (6.33)$$

Here we restrict however the infinite sum above to only contain 2 terms since we assume that the system is prevented from loosing more than 2 packets in a row. One of the reasons why we need this constraint is that since there are no retransmissions, if several packets are lost in a row the observed value $r_{t+\Delta}$ might exceed the range of the quantizer to the extend that a control signal equal to $\pm q_{max}$ can not steer the state back to 0. For low erasure rates this restriction has very little impact.

The energy of the estimation error x_{e0} is determined as follows:

$$x_{e0} = \hat{x}_t - s_t = y_t^q - s_t = (y_t + n_t) - s_t = s_t + n_t + v_t - s_t = n_t + v_t, \quad (6.34)$$

and then

$$\mathbb{E}[x_{e0}^2] = \sigma_v^2 + \frac{\delta^2}{12}. \quad (6.35)$$

Given the packet erasure rate and system parameters we can now determine the approximate cost function for the block code by inserting the above into (6.33):

$$J \approx (1-\epsilon_{packet})(\lambda^{2\Delta}\left(\sigma_v^2+\frac{\delta^2}{12}\right)+\sigma_{\tilde{w}}^2+\sum_{j=1}^{2}\epsilon_{packet}^j\lambda^{(j+1)2\Delta}\left(\sigma_v^2+\frac{\delta^2}{12}\right)+\sum_{i=0}^{2j}\lambda^{i\Delta}\sigma_{\tilde{w}}^2)+\sigma_v^2. \quad (6.36)$$

In the following figures we use the value of the packet erasure rate when (6.12) is fulfilled with equality. Fig. 6.6 shows the calculated control cost for two different unstable plants when the channel between observer and controller is a BEC with erasure rate $\epsilon_{BEC} = \epsilon$. The codeword of the block code consists of $n_{BC} = 100$ bits and this codeword is transmitted over the channel using Δ time steps. The plant is either controlled every $\Delta = 5$ time steps (sending 20 bits per time step) or every $\Delta = 20$ time steps (sending 5 bits per time step). The number of quantization bits is determined using (6.28). We can see that the control cost is constant over all ϵ for the plant with a low coefficient λ. The reason for this is that for a plant with rather low dynamics, a loss of a packet does not have a severe impact. The control cost can however be lowered if the plant is controlled in shorter time intervals ($\Delta = 5$). For a plant with higher dynamics $\lambda = 1.7$ we can see in Fig. 6.6 that the control cost increases rapidly for an increasing erasure rate ϵ if the plant is controlled in long time intervals ($\Delta = 20$). If the plant is controlled more often a flat curve can be achieved even for the plant with $\lambda = 1.7$.

6.4.4 Conclusion

We have seen that a block code in the context of networked control shows a good performance as long as the dynamic of the plant is low and the number of bits that can be transmitted over the channel per time step is high.

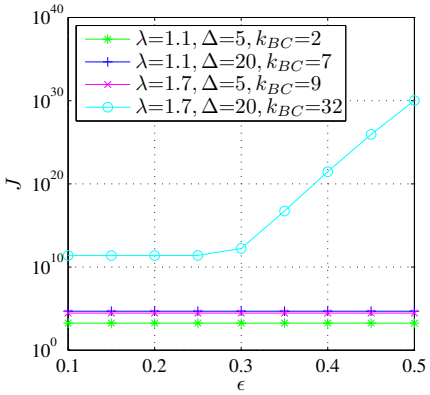


Figure 6.6: Calculated control cost J for different channel erasure rates ϵ, different plant dynamics λ and different time updates Δ when using a block code with $n_{BC} = 100$ bits per codeword. $\sigma_v = 5, \sigma_w = 15$.

6.5 Analysis and Performance Evaluation for Anytime Codes

In this section we study the behavior of the system when using anytime codes instead of block codes for error protection. As for the block codes we analyze the number of quantization bits necessary to stabilize the plant and we derive the cost function.

6.5.1 Constraint on the Number of Quantization Bits

A constraint on the number of quantization bits is derived by inspecting the δ-based quantizer given in Section 6.3.2 pictured in Fig. 6.4. If we use a quantizer with bin width $\delta > V$ then, if at any time t we have $r_t \in (-\frac{\delta}{2}, \frac{\delta}{2})$ and the controller is aware of it, then for the state we know $s_t \in (-\frac{\delta}{2} - \frac{V}{2}, \frac{\delta}{2} + \frac{V}{2})$). If $u_t = 0$ in the next time step $s_{t+1} \in (-|\lambda|\frac{\delta+V}{2} - \frac{W}{2}, |\lambda|\frac{\delta+V}{2} + \frac{W}{2})$ and therefore $r_{t+1} \in (-|\lambda|\frac{\delta+V}{2} - \frac{W+V}{2}, |\lambda|\frac{\delta+V}{2} + \frac{W+V}{2})$. Hence r_{t+1} can lie in at most one of $\frac{|\lambda|(\delta+V)+W+V}{\delta}$ possible bins. Then we need the number of quantization levels $2^{k_{AC}}$ to be larger than the number of possible bins. That is

$$2^k_{AC} > \frac{|\lambda|(\delta + V) + W + V}{\delta}. \tag{6.37}$$

It follows that the number of quantization bits has to fulfill

$$k_{AC} > \log_2 \left(\frac{|\lambda|(\delta + V) + W + V}{\delta} \right). \tag{6.38}$$

It was shown in [SH11c] that it is possible to stabilize an unstable system in the mean-squared sense with an (R, α)-anytime code provided

$$R > R_n = \frac{1}{n}\log_2(|\lambda|), \qquad \alpha > \alpha_n = \frac{2}{n}\log_2(|\lambda|) \tag{6.39}$$

By choosing the number of quantization bits according to (6.38) the rate constraint is therefore always fulfilled.

6.5.2 Cost Function

For the anytime code the energy $\mathbb{E}[s_{t+1}^2]$ and therefore as well the control cost given in (6.3) depends on what is known about the previous states. To keep track of this knowledge we analyze the control cost function depending on a decoding window of size ω. Recall that the decoding window is defined as the window containing all bits from the current codeword back to the codeword containing the first erroneous message bit in the stream. For simplicity we moreover assume that given a decoding window of size ω, *none* of the messages within this window is decodable. This is a strong simplification that is however necessary in order to derive simple analytical expressions for the control cost.

Assume that the decoding window is of size 0, this means that $\hat{s}_t$ is accurately known. Then similarly as for the block code we can determine the energy of $\mathbb{E}[s_{t+1}^2]$ as

$$\mathbb{E}[s_{t+1}^2|\omega = 0] = \lambda^2\left(\sigma_v^2 + \frac{\delta^2}{12}\right) + \sigma_w^2. \tag{6.40}$$

For a decoding window of size 1, the message block 1 time step in the past contains an error and is therefore not decodable. Given $\hat{s}_t = \hat{r}_t$ the estimate $\hat{s}_{t+1}$ is then equal to 0 according to the calculation

$$\hat{s}_{t+1} = \frac{1}{2}(\hat{s}_{min,t+1} + \hat{s}_{max,t+1}) = \ldots = \lambda\hat{r}_t - \lambda\hat{s}_t = 0, \tag{6.41}$$

where we have inserted the terms for $\hat{s}_{min,t+1}$ and $\hat{s}_{max,t+1}$ from (6.21) and (6.23). It follows that $\mathbb{E}[s_{t+2}^2]$ for a decoding window of size 1 is equal to

$$\mathbb{E}[s_{t+2}^2|\omega = 1] = \lambda^2\mathbb{E}[s_{t+1}^2] + \sigma_w^2. \tag{6.42}$$

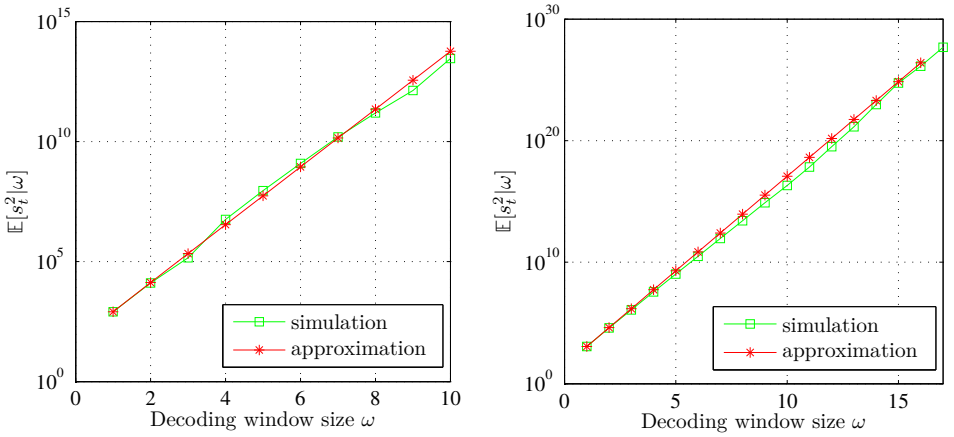
If the decoding window is of size 2, then r_{t+1} is not decodable. Under the assumption that r_{t+2} is not decodable neither we get

$$\mathbb{E}[s_{t+3}^2|\omega = 2] = \lambda^2\mathbb{E}[s_{t+2}^2] + \sigma_w^2. \tag{6.43}$$

For a decoding window of size ω the general term is then given as

$$\hat{\mathbb{E}}[s_{t+\omega+1}^2|\omega] \leq \lambda^{2w}\mathbb{E}[s_{t+\omega}^2|\omega - 1] + \sigma_w^2 = \sum_{i=0}^{\omega}\lambda^{2\omega}\left(\left(\sigma_v^2 + \frac{\delta^2}{12}\right) + \sigma_w^2\right). \tag{6.44}$$

The inequality comes from the fact that in general we cannot make the assumption that *all* messages within the window of size ω are not decodable. We can see that the energy increases approximately exponentially over the decoding window size ω with a rate of λ^2. The reason for this is that, since we assume that none of the states within the decoding window is decodable, for a decoding window size of ω the last control signal applied that is not equal to 0 is $\omega + 1$ time steps in the past. During the time ω the system is therefore uncontrolled and the state s_t grows exponentially. We can compare the upper bound to the value of $\mathbb{E}[s_t^2|\omega]$ obtained through simulations. Fig. 6.7 shows the upper bound together with the simulated value of $\mathbb{E}[s_t^2|\omega]$ when controlling two unstable plants with different characteristics over an erasure channel with $\epsilon = 0.2$ using an $R = 1/2$-anytime code and a quantizer with $\delta = 3$. As we can see the simulated behavior is very close to the upper bound determined in (6.44).



(a) $\mathbb{E}[s_t^2|\omega]$ for $\lambda = 4, \sigma_w = 20, \sigma_v = 5$.

(b) $\mathbb{E}[s_t^2|\omega]$ for $\lambda = 6, \sigma_w = 15, \sigma_v = 5$.

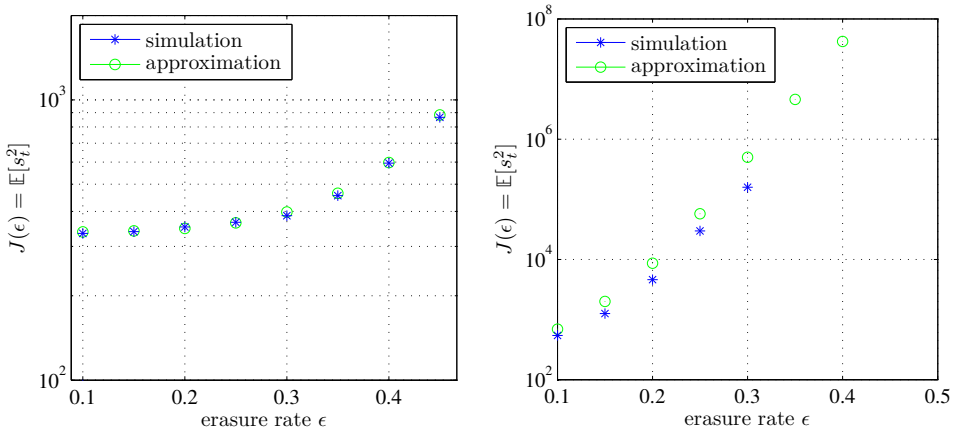
Figure 6.7: Energy statistics $\mathbb{E}[s_t^2|\omega]$ depending on the decoding window size ω for an $R = 1/2$-anytime code with $k = 10$, and quantizer with $\delta = 3$, when controlling an unstable plant over an erasure channel with $\epsilon = 0.2$.

With the above we can now determine an approximation of the cost function as follows

$$J = \mathbb{E}[s_t^2] + \sigma_v^2 \approx \sum_{w=0}^{\infty} P(\omega) \left(\lambda^{2(w+1)} \mathbb{E}[s_{e0}^t] + \sum_{i=0}^{w} \lambda^{2i} \sigma_w^2 \right) + \sigma_v^2, \tag{6.45}$$

where $P(\omega)$ is given in (3.50). Figs. 6.8a and 6.8b show the simulated control cost together with the approximation determined above for systems with different λ and codes with different rates. Since we only want to investigate the effect of the control cost, we do not consider erasure rates where an increasing erasure event occurs

with high probability. The average control cost $\mathbb{E}(s_t^2)$ is measured over $T = 30$ time instances for different channel erasure rates ϵ when controlling an unstable plant with $\lambda = 2, \sigma_v = 5, \sigma_w = 15$ using an anytime code for error protection. For the low-rate code ($R = 0.25$) the probability of an increasing erasure event is very low and the calculated control cost is very close to the simulated one. Moreover the cost does not increase significantly for increasing erasure rates since the low rate code is capable of maintaining good error correction properties. For the code of higher rate ($R = 0.5$) the probability of an increasing erasure event is higher and dominates the system performance for larger erasure rates ϵ. For low erasure rates the calculated cost is close to the simulated cost. The cost increases significantly for increasing erasure rates since the anytime code gradually looses its error correction capabilities.



(a) Code and quantizer with $k = 5, R = 0.25, \delta = 6.1$.

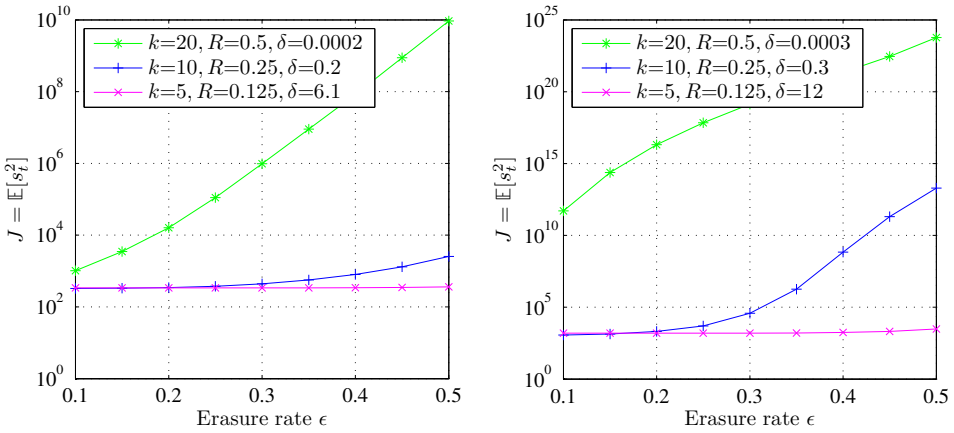
(b) Code and quantizer with $k = 10, R = 0.5, \delta = 3$.

Figure 6.8: Simulated and calculated control cost for different erasure rates ϵ for a system with $\lambda = 2, \sigma_v = 5, \sigma_w = 15$ measured over $T = 30$ time instances.

6.5.3 Quantization Resolution and Code Rate

In Fig. 6.9 we investigate the effect of changing the rate of the code while keeping the number of bits in a codeword constant (here $n_{AC} = 40$). The quantization interval δ is chosen such that it fulfills (6.38). We can see that for a low rate code using fewer quantization bits the control cost is generally lower than for a high rate code with more bits spend on quantization. Moreover we notice that the lower the rate of the code the less sensitive the control cost is to an increased erasure rate. From the figures we conclude that in terms of the control cost a good error protection is more important than a fine quantization. By comparing the two

figures we furthermore note that this becomes even more important for a system with higher dynamics.



(a) System with low dynamics $\lambda = 2$.

(b) System with high dynamics $\lambda = 6$.

Figure 6.9: Comparison of control cost when using an anytime code with $n_{AC} = 40$ but different rates R_{AC}.

6.5.4 Conclusion

We conclude that the control cost of the system using an anytime code can be reduced by decreasing the rate of the code. It is possible to achieve an almost constant control cost over all erasure rates up to $\epsilon = 0.5$ by decreasing the rate. The rate decrease can be achieved by lowering the number of message bits while keeping the number of transmitted code bits constant. We could see that in the considered setup a very coarse quantization was enough to be able to control the system and we conclude that much of the complexity should therefore be shifted to the channel code.

6.6 Comparing Anytime Codes with Block Codes

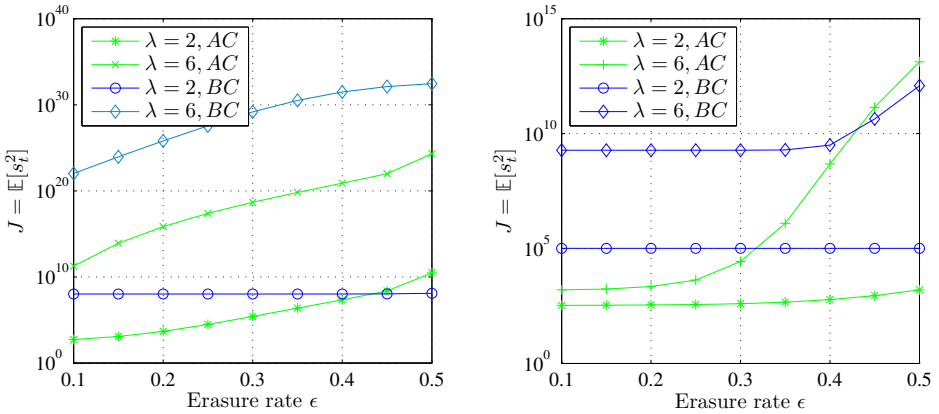
We now want to compare the performance of a system using anytime codes with a system using block codes. The comparison is done in terms of the control cost function. Since the control cost calculations involve a couple of assumptions and approximations the comparison is to be seen as an indication only. Given a BEC with an erasure rate ϵ, the process and measurement noise variances σ_v^2 and σ_w^2, and a constraint on the number n_{max} of bits that can be transmitted at maximum during one time step, we determine the parameters of the anytime code and of the block code in order to calculate the control cost function.

6.6.1 Anytime Code Parameters

The anytime code parameters are found as follows: The codeword length n_{AC} is chosen equal to n_{max}. We pick a message length $k_{AC} = n_{max}/i$ such that i is an integer and determine the minimum quantization interval size δ that fulfills (6.38), where for the state estimator we set $V = 6\sigma_v$ and $W = 6\sigma_w$. The rate of the anytime code is then given as $R_{AC} = k_{AC}/n_{AC}$.

6.6.2 Block Code Parameters

The block code parameters are found as follows: The block code codeword length n_{BC} is always set to 100 bits. The reason for this is that from this codeword length onwards equation (6.12) is valid. The update interval Δ is determined as $\Delta = 100/n_{max}$. The message length k_{BC} is then chosen such that it fulfills (6.28).



(a) $n_{max} = 10$ bits. Anytime code of rate $R_{AC} = 0.5$ with $k_{AC} = 5$ with $\delta = 6.1$ ($\lambda = 2$) or $\delta = 12$ ($\lambda = 6$). Block code with $\Delta = 10, n_{BC} = 100, k_{BC} = 51$.

(b) $n_{max} = 20$. Anytime code of rate $R_{AC} = 0.25$ with $k_{AC} = 5$ with $\delta = 6.1$ ($\lambda = 2$) or $\delta = 12$ ($\lambda = 6$). Block code with $\Delta = 5, n_{BC} = 100, k_{BC} = 25$.

Figure 6.10: Comparison of the calculated control cost when using an anytime code or a block code. System with $\sigma_v = 5, \sigma_w = 15$.

6.6.3 Comparison

Fig. 6.10 compares the calculated control cost of a system using an anytime code with a system using a block code with the parameters obtained according to the previous description. The figure illustrates the relation between the available resources (given by n_{max}), the dynamics of the system (described with λ), the resolution of the quantizer (given by k_{AC} and k_{BC}) and the code rate (R_{AC}, R_{BC}). We can see in both figures that for the a system with low dynamics ($\lambda = 2$) the anytime code

has a slightly better performance for almost all rates. For a system with higher dynamics ($\lambda = 6$) the difference is more profound at least for low or moderate erasure rates. The figures show simulations for a different maximum number of transmitted bits n_{max} per time step. We notice that for a higher n_{max} both the system with the anytime code as well as the system using a block code are improved. Due to the increased number of bits transmitted over the channel in one time step, the block code is capable of updating the state observation in shorter time intervals and therefore shows a rather good performance even for the system with higher dynamics.

6.6.4 Conclusion

We conclude that for a system where the number of bits that can be transmitted over the channel in one time step is limited and the plant that has to be controlled is highly dynamic the system largely benefits in terms of the control cost from using an anytime code for error protection instead of a block code.

Chapter 7

Conclusion

7.1 Summary and Concluding Remarks

In this thesis we have studied anytime codes and their application in a control setup. We have analyzed both theoretical aspects and practical considerations. We summarize our results and conclusions in the following.

In Chapter 3, we constructed error-correcting codes with anytime properties. These codes are protograph-based LDPC-CCs with infinite memory. We have shown with the help of P-EXIT analysis that over the BEC in the asymptotic limit of infinite block length these codes have anytime properties when decoded with an expanding-window decoder. By simulating the codes for very short block lengths, we identified fundamental problems of the code structure for finite block lengths and we presented a combinatorial approach to analyze the finite-length behavior. We conclude that for anytime codes it is of utmost importance that for the code design we cannot just consider the behavior of the ensemble average in the limit of infinite block length but have to pay much more attention to the effects occurring for small block lengths. The analysis given in this chapter provides a useful approach for the design of the codes with different parameters. When transmitting over the AWGN channel, we observed a very similar behavior for finite block lengths as when transmitting over the BEC. We therefore conclude that the design approaches made for the BEC are useful even when designing codes for transmission over the AWGN channel and probably other channels as well.

In Chapter 4, we modified the proposed code structure in several ways in order to take different practical constraints into account. The modifications involved the increase or decrease of the density of the protograph, the decrease of the rate of the code and the limitation of the memory in the protograph. By analyzing both the asymptotic and the finite-length performance we could show the advantages and disadvantages of the different modifications. We conclude that there is a tradeoff between the delay-sensitivity and reliability of the coding scheme, where through modifications of the code structure we can decide the impact of the one or the other. The discussion showed that the proposed codes have a high degree of flexibility. The

codes can therefore possibly be adapted to various other constraints than the ones shown here while still being amenable for analysis.

In Chapter 5, we investigated different approaches to improve the finite-length behavior of the anytime codes by detecting problematic behavior and reacting on it. Terminating the codes in regular time intervals was shown not to be very promising due to the loss of the anytime decoding structure of the code. Allowing the decoder to feedback information to the encoder on the other hand, was demonstrated to lead to a significant performance improvement. A feedback strategy using the knowledge of the erasure positions was proposed for transmission over the BEC. Two other feedback protocols were described for the BEC or the AWGN channel, leading to a variable rate coding scheme which allows transmission with high reliability even over low-quality channels. We expect, that the progressive improvement of the reliability over delay in connection with a feedback link from the receiver to the transmitter is not only interesting in the context of automatic control but as well in other areas, where the receiver can trade off reliability with delay-sensitivity constraints.

Finally, in Chapter 6, we integrated the error-correcting codes in a networked control system. We specified the different components of the overall system. By comparing the control performance using the codes proposed in this thesis with the control performance using block codes we showed the superiority of anytime codes when used to stabilize highly unstable systems with severe resource constraints. We conclude that especially in a setup where communication resources are very limited, the use of anytime codes instead of block codes can significantly improve the performance of the overall system. The importance of choosing a proper error-correcting code to secure the communication in a networked control system is thus underlined.

7.2 Future Work

There are many questions open in the field of coding for networked control systems. Here we only mention some of them:

- In this thesis we have only investigated the behavior of the anytime codes for the BEC and the AWGN channel, a natural question to ask is how these codes perform on other channels. Recently, in [TNDT13a] the performance was studied when transmitting over a Rayleigh fading channel. Other anytime codes have been studied for the relay channel in [NARNL14].

- The flexibility of the code structure allows for many other modifications than the ones proposed in this thesis. It is interesting to see what changes in the structure might lead to an even better finite-length behavior. Recently, in [ZNARVN15a] the effect of introducing an element of randomness in the proposed code structure was shown to be promising.

- In this thesis, there are no attempts to remove small cycles from the parity-check matrix after construction. It would be interesting to see how big the impact of a cycle removing algorithm is on the performance of the code for finite block lengths.
- In this thesis we investigated the behavior of the proposed codes for UDP-like protocols with and without a feedback link that could be used to transmit data. It is interesting to see how good the performance can be when tailoring the proposed codes to transmission over TCP-like protocols.
- In Chapter 6 we made the first attempt to integrate anytime codes in an LQG control setup. Further investigating the interplay of quantizer, controller, encoder etc. is a very interesting direction of research since relating the error-correcting properties more closely to various other parameters in the control setup is expected to allow for a much better overall understanding and design of a NCSs.
- The anytime codes are so far only employed to ensure a reliable communication between the observer and the controller, assuming that the controller and the plant are connected via a noise free link. A further step would be to investigate the performance of the overall system when even the link between controller and plant is noisy and has to be protected using error-control schemes.
- In this thesis, we focused on a networked control setup as intended application. There are probably many more applications possible, where progressive reliability/refinement with delay is a relevant aspect. As an example we could imagine searching for an object in images from the Internet. The algorithm starts on a low-quality version of the image. It can stop early if an object is likely not to be found and continues to download and search a refined version of the image while the probability that the object is present is sufficiently high.

Chapter A

Appendix

A.1 Toeplitz Ensemble

In [SH11b] the authors consider time-invariant codes with Toeplitz generator and parity check matrices called *Toeplitz ensemble*.

A.1.1 Code Construction

The ensemble of time-invariant codes with parity check matrices $\boldsymbol{H}_{n,R}$, where n is the codeword length and R is the rate of the code, is obtained as follows: $\boldsymbol{H}_{n,R}$ has Toeplitz structure with

$$\boldsymbol{H}_{n,R}^{TZ} = \begin{bmatrix} \boldsymbol{H}_1 & & & & \\ \boldsymbol{H}_2 & \boldsymbol{H}_1 & & & \\ \vdots & \vdots & \ddots & & \\ \boldsymbol{H}_\tau & \vdots & \ddots & \boldsymbol{H}_1 & \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \tag{A.1}$$

where $\boldsymbol{H}_{i,j}$ is of size $n(1-R) \times n$, $\boldsymbol{H}_1$ is any full rank binary matrix and for $\tau \geq 2$, the entries of $\boldsymbol{H}_\tau$ are chosen i.i.d. according to Bernoulli(p), i.e. each entry is equal to 1 with probability p and 0 otherwise. Since $\boldsymbol{H}_1$ is of full rank, so is the leading principal minor $\boldsymbol{H}_{n,R}^t$ of size $n(1-R)t \times nt$ of $\boldsymbol{H}_{n,R}$.

A.1.2 Decoding

The authors in [SH11b] propose a decoding algorithm suitable when transmitting over the BEC. The algorithm performs maximum likelihood decoding by solving a set of equations. Denote by $\boldsymbol{H}_{n,R}^t$ the submatrix in $\boldsymbol{H}_{n,R}^{TZ}$ consisting of the rows $1 \dots n(1-R)t$ and the columns $1 \dots nt$. Since $\boldsymbol{H}_1$ is full rank, so is $\boldsymbol{H}_{n,R}^t$ for all t. Consider an arbitrary decoding instant t, assume that for the transmitted

codeword $\boldsymbol{y} = [\boldsymbol{y}_1^T, \dots \boldsymbol{y}_t^T]$ the corresponding channel output is $\tilde{\boldsymbol{y}} = [\tilde{\boldsymbol{y}}_1^T, \dots \tilde{\boldsymbol{y}}_t^T]$. Let $\tilde{\boldsymbol{y}}_e$ denote the erasures in $\tilde{\boldsymbol{y}}$ and let $\boldsymbol{H}_e$ denote the columns of $\boldsymbol{H}_{n,R}^t$ that correspond to the positions of the erasures. Denote the unerased entries of $\tilde{\boldsymbol{y}}$ as $\tilde{\boldsymbol{y}}_{ue}$ and let $\boldsymbol{H}_{ue}$ denote the columns of $\boldsymbol{H}_{n,R}^t$ excluding $\boldsymbol{H}_e$. We have the following parity check condition on $\tilde{\boldsymbol{y}}_e$:

$$\boldsymbol{H}_e \tilde{\boldsymbol{y}}_e = \boldsymbol{H}_{ue} \tilde{\boldsymbol{y}}_{ue}. \tag{A.2}$$

All unerased entries $\tilde{\boldsymbol{y}}_{ue}$ are known at the decoder and therefore the right hand side $\boldsymbol{s} := \boldsymbol{H}_{ue}\tilde{\boldsymbol{y}}_{ue}$ above is known. Maximum likelihood decoding means therefore solving the linear equation $\boldsymbol{H}_e\tilde{\boldsymbol{y}}_e = \boldsymbol{s}$. Now most likely we cannot recover the entire vector $\tilde{\boldsymbol{y}}_e$ by solving the above equation since $\boldsymbol{H}_e$ has lower-triangular structure and will typically not have full column rank. This is not necessary neither since for the anytime property we only need an exponential improvement from the last time step. And therefore we do not need to solve for *all* $\tilde{\boldsymbol{y}}_e$. We can write $\boldsymbol{H}_e\tilde{\boldsymbol{y}}_e = \boldsymbol{s}$ as

$$\begin{bmatrix} \boldsymbol{H}_{e,11} & 0 \\ \boldsymbol{H}_{e,21} & \boldsymbol{H}_{e,22} \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{y}}_{e,1} \\ \tilde{\boldsymbol{y}}_{e,2} \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_1 \\ \boldsymbol{s}_2 \end{bmatrix}, \tag{A.3}$$

where $\tilde{\boldsymbol{y}}_{e,1}$ correspond to the values of $\tilde{\boldsymbol{y}}_e$ up to time t^* and $\tilde{\boldsymbol{y}}_{e,2}$ correspond to the values of $\tilde{\boldsymbol{y}}_e$ from $t^* + 1 \dots t$. By multiplying both sides with $\mathrm{diag}(\boldsymbol{I}, \boldsymbol{H}_{e,22})$ we get

$$\begin{bmatrix} \boldsymbol{H}_{e,11} \\ \boldsymbol{H}_{e,22}^{\perp}\boldsymbol{H}_{e,21} \end{bmatrix} \tilde{\boldsymbol{y}}_{e,1} = \begin{bmatrix} \boldsymbol{s}_1 \\ \boldsymbol{H}_{e,22}^{\perp}\boldsymbol{s}_2 \end{bmatrix} \tag{A.4}$$

where $\boldsymbol{H}_{e,22}^{\perp}$ denotes the orthogonal complement of $\boldsymbol{H}_{e,22}$, i.e. $\boldsymbol{H}_{e,22}^{\perp}\boldsymbol{H}_{e,22} = 0$. The decoding algorithm now consists of finding the largest value of t^* such that $\left[\boldsymbol{H}_{e,11}^T(\boldsymbol{H}_{e,22}^{\perp}\boldsymbol{H}_{e,21})^T\right]^T$ has full rank.

A.1.3 Decoding Complexity

The authors derive the average decoding complexity to be at most $K\sum_{d>0} d^3 2^{-n\alpha d}$, where K is some constant, n is the codeword length and α is the anytime exponent. The encoding complexity per time iteration increases linearly with time.

A.2 Spatially Coupled Anytime Codes

In [NARNL15, NARNL13] a class of spatially coupled codes is proposed and shown to have anytime properties asymptotically.

A.2.1 Code Construction

The codes are constructed by coupling a chain of L standard (d_v, d_c)-regular LDPC codes or protographs, where d_v and d_c are the degrees of variable nodes and check

nodes, respectively. The variable L is referred to as chain length and it is assumed to be large. Each of the d_v connections of a variable node at position i is possibly connected to a check node in the range i to $i + \gamma - 1$, where γ is the coupling length. The variable γ is chosen large in order to improve the reliability of each message with large delays. Fig. A.1 depicts a $(3, 6)$-regular LDPC protograph and a terminated spatially coupled LDPC code with $d_v = 3$ and $d_c = 6$ and uniform distribution of the connections between variable and check nodes. The way the

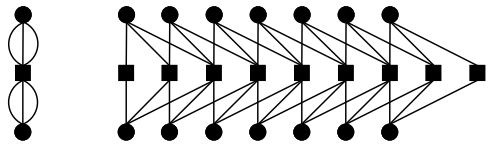


Figure A.1: (3,6)-regular LDPC protograph and a coupled chain of $L = 7$ protographs with $\gamma = 3$.

variable nodes are connected to the check nodes in [NARNL15, NARNL13] follows an exponential distribution, that is a variable node at position i and a check node at position j, where $j > i$ are connected with probability

$$P_r(d) = \frac{e^{-d\lambda}(1 - e^{-\lambda})}{1 - e^{-\gamma\lambda}}, \tag{A.5}$$

where λ is the exponential rate parameter and $d = j - i$ is the distance between the connected check node and variable node. For $\gamma = \infty$ this becomes

$$P_r(d) = e^{-d\lambda}(1 - e^{-\lambda}). \tag{A.6}$$

The code is referred to as a (d_v, d_c)-anytime SCLDPC code. The authors derive the anytime exponent to be $\alpha = \lambda d_v$. The anytime exponent is valid for $\epsilon \leq \bar{\epsilon}_b$ where $\bar{\epsilon}_b$ is given in [NARNL13]. Note that the value of $\bar{\epsilon}_b$ is significantly lower than the erasure rates used in the simulations. The choice of the parameters γ and λ is crucial here. The authors find that an infinite coupling length is necessary for the SCLDPC codes. And the parameter λ decides on whether the decay of the probability of error is fast or slow. For a large λ the connections of check node i are concentrated on nearby variable nodes whereas for a small λ the connections are more spread out.

A.2.2 Asymptotic Analysis

The asymptotic analysis is obtained through the following density evolution equations: Let $p_{i,t}^{\ell}$ denote the erasure probability of a message outgoing from a variable node at position i in the ℓ-th iteration at time t, then for $t \geq i$

$$p_{i,t}^{\ell} = \epsilon \left(1 - \sum_{j=0}^{t-i} P_r(j) \left[1 - \sum_{d=0}^{\infty} P_r(d) p_{i+j-d,t}^{\ell-1} \right]^{d_c - 1} \right)^{d_v - 1} \tag{A.7}$$

where

$$p_{i,t}^{0} = \begin{cases} 0 & \text{if} \quad i \leq 0 \\ \epsilon & \text{if} \quad 0 < i \leq t \\ 1 & \text{if} \quad i > t \end{cases} \tag{A.8}$$

and $P_r(d)$ is given in (A.6). The erasure probability $P_e(i,t)$ of variable node i is then given as

$$P_e(i,t) = \epsilon \left(1 - \sum_{j=0}^{t-i} P_r(j) \left[1 - \sum_{d=0}^{\infty} P_r(d) p_{i+j-d,t}^{\infty} \right]^{d_c-1} \right)^{d_v} . \tag{A.9}$$

Finite-Length Estimation

The authors in [NARNL15] use the results from [AMRU09] to develop an algorithm that estimates the finite-length behavior of the SCLDPC codes. The algorithm takes into account that the observed number of erasures in a block of k bits is not exactly equal to ϵk but instead the number of erasures is distributed according to a binomial distribution $\mathcal{B}(k, \epsilon)$. Conditioned on a channel realization, the algorithm first finds the conditional decoded erasure probability from density evolution and then the erasure probability is obtained by marginalizing over the channel realizations. The details are omitted here.

Bibliography

[AMRU09] A. Amraoui, A. Montanari, T. Richardson, and R. Urbanke. Finite-length scaling for iteratively decoded LDPC ensembles. *IEEE Trans. Inf. Theory*, 55(2):473–498, February 2009.

[BEV14] T. Breddermann, B. Eschbach, and P. Vary. On the design of hybrid automatic repeat request schemes with unreliable feedback. *IEEE Trans. Commun.*, 62(2):758–768, February 2014.

[BM97] V. Borkar and S. Mitter. *LQG Control with Communication Constraints. Communication, Computation, Control and Signal Processing: a tribute to Thomas Kailath*. Kluver, 1997.

[CFRU01] S-Y. Chung, Jr. Forney, G.D., T.J. Richardson, and R. Urbanke. On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit. *IEEE Commun. Letters*, 5(2):58–60, February 2001.

[CFZ10] G. Como, F. Fagnani, and S. Zampieri. Anytime reliable transmission of real-valued information through digital noisy channels. *SIAM J. Control and Opt.*, 48(6):3903–3924, March 2010.

[CIP+10] G. E. Corazza, A. R. Iyengar, M. Papaleo, P. H. Siegel, A. Vanelli-Coralli, and J. K. Wolf. Latency constrained protograph-based LDPC convolutional codes. In *Proc. Int. Symp. Turbo Codes Iterative Inf. Process.*, pages 6–10, Brest, France, September 2010.

[CLSZ13] A. Chiuso, N. Laurenti, L. Schenato, and A. Zanella. LQG cheap control over SNR-limited lossy channels with delay. In *IEEE Ann. Conf. on Dec. and Control*, pages 3988–3993, Florence, Italy, December 2013.

[DPT+02] C. Di, D. Proietti, I.E. Telatar, T.J. Richardson, and R.L. Urbanke. Finite-length analysis of low-density parity-check codes on the binary erasure channel. *IEEE Trans. Inf. Theory*, 48(6):1570–1579, June 2002.

[DRTS12] L. Dössel, L.K. Rasmussen, R. Thobaben, and M. Skoglund. Anytime reliability of systematic LDPC convolutional codes. In *IEEE Int. Conf. Commun.*, pages 2171–2175, Ottawa, ON, June 2012.

[EM01] N. Elia and S. K. Mitter. Stabilization of linear systems with limited information. *IEEE Trans. Autom. Control*, 46(9):1384–1400, September 2001.

[Gal63] R. G. Gallager. *Low-density parity-check codes*. PhD thesis, MIT Cambridge, 1963.

[Gar15] Inc. Gartner. *Press Release*, November 2015.

[GDH+09] V. Gupta, A.F. Dana, J.P. Hespanha, R.M. Murray, and B. Hassibi. Data transmission over networks for estimation and control. *IEEE Trans. Autom. Control*, 54(8):1807–1819, August 2009.

[GMS11] R. Gelles, A. Moitra, and A. Sahai. Efficient and explicit coding for interactive communication. In *IEEE Ann. Symp. of Comput. Sci.*, pages 768–777, Palm Springs, CA, October 2011.

[GRTS14] L. Grosjean, L.K. Rasmussen, R. Thobaben, and M. Skoglund. Systematic LDPC convolutional codes: Asymptotic and finite-length anytime properties. *IEEE Trans. Commun.*, 62(12):4165–4183, December 2014.

[GSHM05] V. Gupta, D. Spanos, B. Hassibi, and R.M. Murray. On LQG control across a stochastic packet-dropping link. In *Proc. American Control Conf.*, pages 360–365, Portland, OR, June 2005.

[ITQ11] Y. Ishido, K. Takaba, and D.E. Quevedo. Stability analysis of networked control systems subject to packet-dropouts and finite-level quantization. *Systems Control Letters*, 60(5):325 – 332, May 2011.

[IYB06] O.C. Imer, S. Yuksel, and T. Basar. Optimal control of dynamical systems over unreliable communication links. *Automatica*, 42(9):1429–1440, September 2006.

[JFZ99] A. Jimenez Feltström and K. S. Zigangirov. Time-varying periodic convolutional codes with low-density parity-check matrix. *IEEE Trans. Inf. Theory*, 45(6):2181–2191, September 1999.

[Kal60] R.E. Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME - Journal of Basic Eng.*, 82(D):35–45, March 1960.

[KRU11] S. Kudekar, T.J. Richardson, and R.L. Urbanke. Threshold saturation via spatial coupling: Why convolutional LDPC ensembles perform so well over the BEC. *IEEE Trans. Inf. Theory*, 57(2):803–834, February 2011.

[LC07] G. Liva and M. Chiani. Protograph LDPC codes design based on EXIT analysis. In *IEEE Global Telecom. Conf.*, pages 3250–3254, Washington D. C., USA, November 2007.

[LH02] G. Leen and D. Heffernan. Expanding automotive electronic systems. *Computer*, 35(1):88–93, January 2002.

[LMSS01] M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, and D.A. Spielman. Efficient erasure correcting codes. *IEEE Trans. Inf. Theory*, 47(2):569–584, February 2001.

[LSCZ10] M. Lentmaier, A. Sridharan, D.J. Costello, and K.S. Zigangirov. Iterative decoding threshold analysis for LDPC convolutional codes. *IEEE Trans. Inf. Theory*, 56(10):5274–5289, October 2010.

[LTF09] M. Lentmaier, M.B.S. Tavares, and G.P. Fettweis. Exact erasure channel density evolution for protograph-based generalized LDPC codes. pages 566 –570, Seoul, Korea, June/July 2009.

[MA07] Alexey S. Matveev and Andrey V. Avkin. *Estimation and Control over Communication Networks (Control Engineering)*. Birkhauser, 2007.

[MDE06] N.C. Martins, M.A. Dahleh, and N. Elia. Feedback stabilization of uncertain systems in the presence of a direct link. *IEEE Trans. Autom. Control*, 51(3):438–447, March 2006.

[MFDN09] P. Minero, M. Franceschetti, S. Dey, and G.N. Nair. Data rate theorem for stabilization over time-varying feedback channels. *IEEE Trans. Autom. Control*, 54(2):243–255, February 2009.

[MGS13] Y. Mo, E. Garone, and B. Sinopoli. LQG control with Markovian packet loss. In *Europ. Control Conf.*, pages 2380–2385, Zürich, Switzerland, July 2013.

[MWC+04] C. Meng, T. Wang, W. Chou, S. Luan, Y. Zhang, and Z. Tian. Remote surgery case: robot-assisted teleneurosurgery. In *IEEE Proc. Robotics Automation*, volume 1, pages 819–823, April 2004.

[NARNL13] M. Noor-A-Rahim, K. Nguyen, and G. Lechner. Anytime characteristics of spatially coupled codes. In *Proc. Annu. Allerton Conf. Commun., Control, Comput.*, pages 335–341, Allerton, IL, October 2013.

[NARNL14] M. Noor-A-Rahim, K. D. Nguyen, and G. Lechner. Anytime spatially coupled codes for relay channel. In *Austrl. Commun. Theory Workshop*, pages 39–44, Sydney, Australia, February 2014.

[NARNL15] M. Noor-A-Rahim, K.D. Nguyen, and G. Lechner. Anytime reliability of spatially coupled codes. *IEEE Trans. Commun.*, 63(4):1069–1080, April 2015.

[NE02] G.N. Nair and R.J. Evans. Mean square stabilisability of stochastic linear systems with data rate constraints. In *Proc. IEEE Conf. Decision Control*, volume 2, pages 1632–1637, Las Vegas, NV, December 2002.

[NE04] G.N. Nair and R.J. Evans. Stabilizability of stochastic linear systems with finite feedback data rates. *SIAM J. Control and Opt.*, 43(2):413–436, July 2004.

[Nil98] J. Nilsson. *Real-time Control Systems with Delays*. PhD thesis, Lund, Inst. Technology, 1998.

[ORS05] R. Ostrovsky, Y. Rabani, and L.J. Schulman. Error-correcting codes for automatic control. In *IEEE Symp. Found. Comput. Sci.*, pages 309–316, Pittsburgh, PA, October 2005.

[PFS+08] A.E. Pusane, A.J. Feltstrom, A. Sridharan, M. Lentmaier, K. Zigangirov, and D.J. Costello. Implementation aspects of LDPC convolutional codes. *IEEE Trans. Commun.*, 56(7):1060 –1069, July 2008.

[PIS+10] M. Papaleo, A. R. Iyengar, P. H. Siegel, J. K. Wolf, and G. E. Corazza. Windowed erasure decoding of LDPC convolutional codes. In *IEEE Inf. Theory Workshop*, pages 1–5, Cairo, Egypt, January 2010.

[PPV10] Y. Polyanskiy, H.V. Poor, and S. Verdu. Channel coding rate in the finite blocklength regime. *IEEE Trans. Inf. Theory*, 56(5):2307–2359, May 2010.

[Pus08] A.E. Pusane. *Analysis of LDPC convolutional codes derived from LDPC block codes*. PhD thesis, Notre Dame, IN, 2008.

[Ric03] T. Richardson. Error-floors of LDPC codes. 2003.

[RSU01] T.J. Richardson, M.A. Shokrollahi, and R.L. Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *IEEE Trans. Inf. Theory*, 47(2):619–637, February 2001.

[RU01] T.J. Richardson and R.L. Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Trans. Inf. Theory*, 47(2):599–618, February 2001.

[Sah01] Anant Sahai. *Anytime Information Theory*. PhD thesis, MIT, 2001.

[Sah04] S. Sahai. The necessity and sufficiency of anytime capacity for control over a noisy communication link. In *Proc. IEEE Conf. Decision Control*, volume 2, pages 1896–1901, Paradise Islands, The Bahamas, December 2004.

[SAYK10] B. Smith, M. Ardakani, W. Yu, and F. R. Kschischang. Design of irregular ldpc codes with optimized performance-complexity tradeoff. *IEEE Trans. Commun.*, 58(2):489–499, February 2010.

[Sch96] L. J. Schulman. Coding for interactive communication. *IEEE Trans. Inf. Theory*, 42(6):1745–1756, June 1996.

[SGQ10] E. Silva, G. Goodwin, and D. Quevedo. Control systems design subject to SNR constraints. *Automatica*, 46(2):428 – 436, February 2010.

[SH11a] R. T. Sukhavasi and B. Hassibi. Linear error correcting codes with anytime reliability. In *IEEE Int. Symp. Inf. Theory*, St. Petersburg, Russia, June 2011.

[SH11b] R.T. Sukhavasi and B. Hassibi. Anytime reliable codes for stabilizing plants over erasure channels. In *IEEE Conf. Decision Contr.*, pages 5254–5259, Orlando, FL, December 2011.

[SH11c] R.T. Sukhavasi and B. Hassibi. Error correcting codes for distributed control. *ArXiv e-prints*, 2011.

[Sha48] C.E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(4):623–656, October 1948.

[SJV04] T. Simsek, R. Jain, and P. Varaiya. Scalar estimation and control with noisy binary observations. *IEEE Trans. Autom. Control*, 49(9):1598–1603, September 2004.

[SM06] A. Sahai and S. Mitter. The necessity and sufficiency of anytime capacity for stabilization of a linear system over a noisy communication link - Part I: Scalar systems. *IEEE Trans. Inf. Theory*, 52(8):3369–3395, August 2006.

[Sri05] A. Sridharan. *Design and analysis of LDPC convolutional code*. PhD thesis, University of Notre Dame, IN, February 2005.

[SS01] P. Seiler and R. Sengupta. Analysis of communication losses in vehicle control problems. In *Proc. IEEE American Contr. Conf.*, volume 2, pages 1491–1496, Arlington, VA, June 2001.

[SS05] P. Seiler and R. Sengupta. An h-infinty approach to networked control. *IEEE Transactions on Automatic Control*, 50(3):356–364, March 2005.

[SSF+03] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M.I. Jordan, and S.S. Sastry. Kalman filtering with intermittent observations. In *IEEE Ann. Conf. on Dec. and Control*, volume 1, pages 701–708, Maui, HI, December 2003.

[SSF+05a] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, and S. Sastry. LQG control with missing observations and control packets. In *IFAC World Congress*, Prague, Czech Republic, July 2005.

[SSF+05b] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, and S.S. Sastry. An LQG optimal linear controller for control systems with packet losses. In *IEEE Europ. Contr. Conf.*, pages 458–463, Cambridge, UK, December 2005.

[SSF+05c] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, and S.S. Sastry. Optimal control with unreliable communication: the TCP case. In *Proc. American Contr. Conf.*, pages 3354–3359 vol. 5, Portland, OR, June 2005.

[SSF+06] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, and S.S. Sastry. Optimal linear LQG control over lossy networks without packet acknowledgment. In *Proc. IEEE Conf. Decision Control*, pages 392–397, San Diego, CA, December 2006.

[Ste94] R.F. Stengel. *Optimal Control and Estimation*. Dover, 1994.

[SV03] A. Sangiovanni-Vincentelli. Electronic-system design in the automobile industry. *IEEE Micro*, 23(3):8–18, May 2003.

[tBKA04] S. ten Brink, G. Kramer, and A. Ashikhmin. Design of low-density parity-check codes for modulation and detection. *IEEE Trans. Commun.*, 52(4):670–678, April 2004.

[Tho03] J. Thorpe. Low-density parity-check codes constructed from protographs. Technical report, JPL INP, 2003.

[TIH09] K. Tsumura, H. Ishii, and H. Hoshina. Tradeoffs between quantization and packet loss in networked control of linear systems. *Automatica*, 45(12):2963 – 2970, December 2009.

[TM04] S. Tatikonda and S. Mitter. Control under communication constraints. *IEEE Trans. Autom. Control*, 49(7):1056–1068, July 2004.

[TNDT13a] A. Tarable, A. Nordio, F. Dabbene, and R. Tempo. Anytime reliable LDPC convolutional codes for networked control over wireless channel. In *IEEE Int. Symp. Inf. Theory*, pages 2064–2068, Istanbul, Turkey, July 2013.

[TNDT13b] A. Tarable, A. Nordio, F. Dabbene, and R. Tempo. LDPC codes for networked control. *Technical Report*, 2013.

[TSM04] S. Tatikonda, A. Sahai, and S. Mitter. Stochastic linear control over a communication channel. *IEEE Trans. Autom. Control*, 49(9):1549–1561, September 2004.

[TSS+04] R.M. Tanner, D. Sridhara, A. Sridharan, T.E. Fuja, and Jr. Costello, D.J. LDPC block and convolutional codes based on circulant matrices. *IEEE Trans. Inf. Theory*, 50(12):2966 – 2984, December 2004.

[UR08] R. Urbanke and T. Richardson. *Modern Coding Theory*. Cambridge, 2008.

[WB99] W.S. Wong and R.W. Brockett. Systems with finite communication bandwidth constraints. ii. stabilization with limited information feedback. *IEEE Trans. Autom. Control*, 44(5):1049–1053, May 1999.

[YB04] S. Yuksel and T. Basar. Minimum rate coding for state estimation over noiseless channels. In *Proc. IEEE Conf. Decision Control*, volume 4, pages 4503–4508, Paradise Islands, The Bahamas, December 2004.

[Yuk09] S. Yuksel. A random time stochastic drift result and application to stochastic stabilization over noisy channels. In *Ann. Allerton Conf. Commun., Contr., Comput.*, pages 628–635, Monticello, IL, September/October 2009.

[Zam08] S. Zampieri. Trends in networked control systems. In *Proc. IFAC World Congress*, Seoul, Korea, July 2008.

[ZNARVN15a] Nan Zhang, M. Noor-A-Rahim, B. N. Vellambi, and K. D. Nguyen. Anytime properties of protograph-based repeat-accumulate codes. In *IEEE Inf. Theory Workshop*, pages 177–181, Jeju Island, Korea, October 2015.

[ZNARVN15b] Nan Zhang, M. Noor-A-Rahim, B.N. Vellambi, and K.D. Nguyen. Protograph-based anytime reliable channel coding design. In *IEEE Int. Conf. Comput.*, pages 4048–4053, London, UK, June 2015.