2011

Bit error rate locked loops using log-likelihood error correction decoders

Eric Rives

BIT ERROR RATE LOCKED LOOPS

USING

LOG-LIKELIHOOD ERROR CORRECTION DECODERS

by

ERIC RIVES

A DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in
The Department of Electrical and Computer Engineering
to
The School of Graduate Studies
of
The University of Alabama in Huntsville

HUNTSVILLE, ALABAMA

2011

In presenting this dissertation in partial fulfillment of the requirements for a doctoral degree from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department or the Dean of the School of Graduate Studies. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this dissertation.

_____ _____

(student signature) (date)

DISSERTATION APPROVAL FORM

Submitted by Eric Rives in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical Engineering and accepted on behalf of the Faculty of the School of Graduate Studies by the dissertation committee.

We, the undersigned members of the Graduate Faculty of The University of Alabama in Huntsville, certify that we have advised and/or supervised the candidate on the work described in this dissertation. We further certify that we have reviewed the dissertation manuscript and approve it in partial fulfillment of the requirements of the degree of Doctor of Philosophy in Electrical Engineering.

_____ Committee Chair

(date)

_____

_____

_____

_____

_____ Department Chair

_____ College Dean

_____ Graduate Dean

ABSTRACT
The School of Graduate Studies
The University of Alabama in Huntsville

Degree Doctor of Philosophy College/Dept Engineering/Electrical and Computer Engineering

Name of Candidate Eric Rives
Title Bit Error Rate Locked Loops Using Log-Likelihood Error Correction Decoders

In this dissertation a new application utilizing error correction code (ECC) decoders is studied, wherein a feedback control loop is attached to the decoder in order to tune the device until it produces a desired average bit error rate (BER) on its codeword estimates. The resulting structure is aptly named a bit error rate locked loop, abbreviated as BERLL. The specific design of the ECC decoder lying at the heart of the loop is not of foremost importance; only the input-to-output characteristics of the decoder and its associated impact on the control loop is considered relevant. The feedback controller takes as input the soft log-likelihood ratio (LLR) variables used by the decoder to determine codeword estimates, while delivering various configuration outputs back to the decoder to achieve the target BER. A trade-off between decoder error correction performance and multiple decoder parameters is thus levied in an effort to minimize the decoder system resource burden while meeting a specified data reliability objective. Direct benefits of operating the decoder in a reduced performance mode include decreased decoder power consumption and decreased decoder throughput latency, per codeword. The loop components of the BERLL are each examined in turn and their impact on overall loop performance is analyzed and discussed. Two independent methods of controlling the performance of the ECC decoder are examined with mathematical analysis and software simulation of the closed loop time domain and frequency domain responses.

Abstract Approval: Committee Chair _____

Department Chair _____

Graduate Dean _____

ACKNOWLEDGMENTS

I am indebted foremost to my family for their support and encouragement throughout my lengthy educational journey resulting in this dissertation.

~

I also thank my advisory committee for providing me with constructive feedback and support.

~

"...let us run with endurance the race that is set before us..."
(Heb. 12:1)

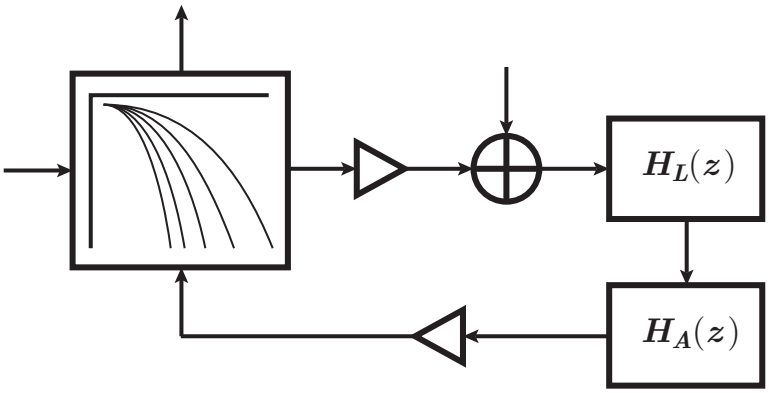


TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AWGN	Additive White Gaussian Noise
BER	Bit Error Rate (or Ratio)
BERLL	Bit Error Rate Locked Loop
BPA	Belief Propagation Algorithm
BPSK	Binary Phase Shift Keying
CCITT	Comité Consultatif International Téléphonique et Télégraphique
CE	Cross Entropy
CSI	Channel State Information
DC	Direct Current
ECC	Error Correction Code (or Control Code)
FER	Frame Error Rate
FIR	Finite Impulse Response
HDA	Hard Decision Aided
IEEE	Institute of Electrical and Electronics Engineers
IIR	Infinite Impulse Response
LDPC	Low-Density Parity-Check
LLR	Log-Likelihood Ratio
LPF	Low-Pass Filter
LUT	Look-up Table
MOR	Measurement of Reliability
MPA	Message Passing Algorithm
OFDM	Orthogonal Frequency Division Multiplexing
PDF	Probability Density Function
PLL	Phase Locked Loop
SCR	Sign Change Ratio
SES	Severely Errored Second
SNR	Signal-to-Noise Ratio
SPA	Sum-Product Algorithm
VCO	Voltage Controlled Oscillator
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network
XOR	Exclusive-Or

LIST OF SYMBOLS

$L(x)$	Log-likelihood ratio of the random variable x.
$\Pr(x)$	Probability of the random variable x.
P_e	Probability of bit being in error.
$\hat{P}_e$	Estimate of P_e.
$\overline{P}_e$	Average of $\hat{P}_e$.
P_c	Probability of bit being correct.
$\hat{P}_c$	Estimate of P_c.
P_{ML}	Maximum-likelihood probability of block error.
P_0	Target probability of bit error, or bit error rate.
ε_1	Bit error rate error candidate 1.
α	Scale factor for bit error rate error candidate 1.
ε_2	Bit error rate error candidate 2.
β	Scale factor for bit error rate error candidate 2.
n	Number of codeword symbols (error correcting code context), Time index (discrete time system context).
k	Number of message symbols.
m	Number of parity symbols.
R	Code rate of an error correcting code.
$\mathbf{C}$	Block error correcting code.
$\mathbf{G}$	Generator matrix for $\mathbf{C}$.
$\mathbf{g}$	Row vector of $\mathbf{G}$.
$\mathbf{H}$	Parity check matrix for $\mathbf{C}$.
$\mathbf{H}_{b,R}$	Parity check base matrix for code rate R.
$\mathbf{I}_{z\times z}$	$z\times z$ identity matrix.
$\mathbf{C}^{\perp}$	Dual code of $\mathbf{C}$.
$\mathbf{m}$	Message symbol vector.
$\mathbf{c}$	Codeword symbol vector.
$\mathcal{N}$	Normal distribution.
σ	Standard deviation.
$\phi(x)$	Gallager's phi-function.
v_i	Variable node (bit vertex) i.
c_j	Check node (parity vertex) j.
J_i	Set of indexes of all check nodes adjacent to v_i.
I_j	Set of indexes of all variable nodes adjacent to c_j.

L_c	Log-likelihood of received channel symbol.
L_0	Target log-likelihood ratio.
L_e	Extrinsic log-likelihood ratio.
L_s	Scaled decoder log-likelihood output.
E_b	Energy per bit (Joules).
N_0	Noise power spectral density (Watts/Hz).
θ	Phase of a periodic signal.
θ_0	Nominal phase of a periodic signal.
ε	Error signal.
ε_F	Filtered error signal.
a	BERLL IIR loop filter feedback coefficient.
b	BERLL IIR loop filter feedforward coefficient.
k_L	BERLL ECC decoder output LLR scale factor.
k_A	BERLL accumulator-to-decoder scale factor.
k_D	BERLL ECC decoder approximate linear scale factor.
c	BERLL lumped parameter product, $k_A k_D k_L$.
ω	Frequency in radians/second.
z	Discrete time system complex-valued variable.
$\mathcal{Z}$	Z-transform operator.
s	Complex valued frequency.
ω_n, ω_0	Natural frequency (radians/second).
T	Discrete time system sampling period (seconds).
ζ	Damping ratio (unitless).
s_i	ECC decoder configuration state variable i.
p_1	Transfer function denominator pole having positive-valued imaginary component.
p_2	Transfer function denominator pole having negative-valued imaginary component.
x_{od}	Real axis origination point in the complex plane for pole p_1 corresponding to maximally-overdamped response.
x_{cd}	Real axis intersection point in the complex plane for poles p_1 and p_2 corresponding to critically-damped response.
p	Complex exponential magnitude of pole p_1.
ω_d	Complex exponential angle of pole p_1 in radians/second.
	Natural resonant frequency in radians/second.
N	Number of decoder codewords per BERLL loop update.

Chapter 1

Introduction

Error correcting codes have played an increasingly important role in communications system design since their inception with Claude Shannon's "A Mathematical Theory of Communication" [1]. Essentially all modern communication systems require some form of error detection or error correction to combat bit errors caused by noisy channel conditions. Recent strides in the field of error correction coding have brought communication link performance quite close to reliability levels promised in Shannon's now-famous theoretical channel coding limit for certain channel types [2], [5], [9]. While by no means can Shannon's channel coding theorem be declared *solved* or *attained*, the progressive convergence of contemporary error correction code performance to that of Shannon's ideal, capacity-achieving code gives rise to a branch of error control coding applications that can adapt error correction strength to simultaneously optimize multiple system-level objectives.

A feature common among high-performance contemporary error correction decoders is the use of multiple decoding cycles per codeword. Two classes of codes utilizing this technique have emerged, and continue to be the prominent focus of recent research and applications: (1) Turbo codes first introduced by Berrou et al. [2], and (2) low-density parity-check (LDPC) codes first described by Gallager [3], [4]. For both code types, each decoding cycle, or iteration, a statistical confidence quantity for each bit in the codeword

called the log-likelihood ratio (LLR) is updated and ideally improved by the decoder. The accuracy of the codeword estimate output from the decoder increases as a function of the LLR magnitude value of the bits comprising the codeword, and is typically an increasing function of the number of decoder iterations. Furthermore, codeword estimate accuracy tends to exhibit a diminishing benefit for each additional iteration, up to a saturation point, above which conducting further iterations provides no measurable improvement to accuracy. At this point of saturation, all of the useful information available to the decoder—embedded in the structured redundancy of the codeword—has been utilized to the furthest extent possible by the decoder.

Each iteration the decoder operates it consumes some amount of available system power and some finite amount of processing time. It stands to reason that operating the decoder for the minimum number of iterations needed to meet a bit error rate specification will be a superior use of system resources as opposed to running the decoder for its maximum iteration limit. This same argument can be made not only in terms of the number of decoder iterations, but also for any decoder parameter that is available for dynamic configuration. Certainly, a lesson learned with newfound appreciation in the present age of mobile devices is that power is a limited resource. Using fewer decoding iterations can provide additional benefits, for example in systems that have the flexibility to use decoded data as soon as it is available, such as variable data rate systems.

Three distinct classes of ECC decoder adaptation become apparent:

1. Those that modify both encoder and decoder parameters.

2. Those that modify only encoder parameters.

3. Those that modify only decoder parameters.

Each adaptation method has unique advantages and disadvantages. One specific disadvantage that exists for method (1) in some implementations is the requirement to pass encoder/decoder configuration across the communication channel; this reduces the amount of bandwidth used for the payload data stream, and also introduces the need for an ECC configuration management scheme, which in turn consumes hardware, software, and power resources. An example of this method is the proposed IEEE 802.16e WiMAX wireless specification, the LDPC codes of which are used throughout this dissertation as a practical simulation basis. Another example technology of method (1) is the most recent physical layer update to the forebearer of IEEE 802.16e, the ubiquitous 802.11n WLAN technology. Within both IEEE 802.16e and 802.11n, the parity check matrix used for encoding and decoding error correction codewords is dynamically selected, which in turn requires message passing across the communication channel for both endpoints to remain synchronized to the present matrix in service.

In contrast, encoder-only and decoder-only adaptation methods operate and adapt to time-varying channel conditions autonomously, without requiring ECC configuration exchange across the channel; thus additional bandwidth, and to a degree some portion of additional ECC configuration management resources are not required to realize these methods. The ECC adaptation method chosen in this study belongs to the class of decoder-only methods. This allows a receiver-side decoder and supporting modules to evaluate the present state of the channel and choose a decoder configuration that achieves some locally-

optimal trade-off between multiple, competing design constraints, while also allowing the transmitter-side encoder to maintain a static configuration.

Within the class of decoder-side ECC adaptation methods, several subclasses can be enumerated, classified by the manner in which channel conditions are measured or estimated. A popular subclass consumes a small fraction of available data bandwidth to examine the degradation on known data patterns embedded in the payload, and adjusts both demodulation and decoding configurations to maximize the signal-to-noise ratio. An example of this first subclass is the pilot tone technique used in many orthogonal frequency division multiplexing (OFDM) systems. Primary drawbacks to OFDM are the sacrifice of payload bandwidth needed to carry the pilot tone signal and the additional signal processing required to utilize the pilot tone waveforms.

A second subclass of the decoder-only adaptive ECC methods are those that make signal-to-noise estimates using the received payload data non-intrusively. This subclass is typically regarded as using what is termed in the literature as channel state information (CSI). The broad-sense idea is that a transmitter and/or receiver examines the characteristics of the communication channel over time, gathering data and estimating such impairments as noise power, noise variance, fading, etc., which are then accounted for and, to the extent possible by the transmitter/receiver pair, compensated for by adjusting demodulation and error correction configurations. The advantage of this second subclass over the first subclass is the ability to retain all available channel bandwidth for the dedicated use of carrying payload data. However, like the first subclass, the second subclass requires a moderate amount of additional signal processing, separate from that necessary for basic signal demodulation and error correction decoding.

In light of the disadvantages of the two decoder-only ECC adaptation methods described thus far, which both require either data bandwidth reduction or at least additional channel measurement signal processing, a third potential subclass of methods makes use of decoder parameters and behavior alone to adjust decoder configuration and thus performance in order to achieve a suitable amount of error correction benefit as a function of dynamic channel conditions. The bit error rate locked loop concept presented in this dissertation belongs to this third subclass. The additional computational overhead needed to operate the loop is quite modest, on the same order of magnitude found in a traditional second order control loop, such as a phase locked loop (PLL).

The layout of the remaining chapters of the dissertation is as follows. Chapter 2 gives an overview of iterative decoding of block codes defined by sparse, or low-density, parity-check matrices. Within Chapter 2, the discussion begins with the well-established definition of the log-likelihood ratio, a fundamental quantity used during the decoding process, and shows how bit error rate can be calculated directly from this ratio. An overview of general block codes is given next, followed by a description of low-density parity-check codes and the iterative message passing belief propagation decoding algorithm, which are the class of correction codes and decoding method used as a basis for simulation and validation of the BERLL concept. Chapter 2 concludes with a summary of the IEEE WiMAX LDPC codes.

Chapter 3 brings the main subject of the dissertation, the bit error rate locked loop, into sharper focus. The purpose of the chapter is to describe in rather moderate detail each of the main loop components used to tune an error correction decoder to produce a required bit error rate on output message bit hard decision estimates. Obvious connections and analo-

gies between the BERLL and the omnipresent PLL are established to better understand the basic operation of the BERLL.

Chapter 4 contains a detailed discrete time linear analysis for the BERLL. The linear model of the loop is based on an approximately linear relationship observed in actual Monte Carlo simulation data between the decoder input parameter (e.g., iteration number) and the output LLR magnitude used to form the loop error signal. This allows the decoder to be replaced with a simple linear gain term for analysis purposes. The chapter lays out in detail the frequency, magnitude and time-domain impulse and step responses as functions of the loop coefficients and gain terms, as well as loop stability and pole-zero analysis. Expressions for natural undamped frequency, natural resonant frequency and damping ratio are also developed.

Chapters 5 and 6 present the results of loop simulations using the number of decoder iterations and number of nulled (discarded) parity bits to control decoder error correction strength, respectively. Underdamped and critically-damped loop response types are simulated to expand the discussion, with frequent comparison and contrast made with the linear analysis in Chapter 4.

Finally, Chapter 7 concludes the dissertation, and explores a few of the possible extensions to the present research and topics for further study.

Chapter 2

Iterative Decoding Fundamentals

This chapter gives an overview of the fundamental elements of iterative decoding. An appreciation of the basic operations involved with iterative decoding is necessary to gain a better understanding of the operation of the bit error rate locked loop. The phrase *iterative decoding* refers to an ECC decoder that is capable of receiving a single codeword and making multiple decoding passes that aim to improve the probability of properly estimating the correct transmitted codeword. One of the earliest descriptions of an iterative decoding technique was given by Gallager in [3] and [4] which was applied to his low-density parity-check codes, which are now some 50 years later finding increased application in modern communication systems, the proposed IEEE 802.16 WiMAX standard [20] being one instance. Another popular class of error correction codes that rely on iterative decoding to achieve high levels of error correction performance are the Turbo codes of Berrou et al.[2], also used in many contemporary communications standards as the error correction technology.

A comprehensive treatment of log-likelihood ratio iterative decoding was given by Hagenauer in [12] for both block codes (including parity-check codes) and convolutional codes, and is regarded as a cornerstone article on the subject. Interestingly, Gallager's LDPC research is not referenced by Hagenauer in [12], despite the former preceding the

latter by 34 years—evidence of the extent to which Gallager's work was essentially *forgotten*, but eventually rediscovered by others in piecemeal fashion.[1]

The chapter begins with a discussion of the log-likelihood ratio and how to estimate bit error rate from this ratio; the resulting equations are stated as definitions which will be relied on in following chapters. A brief survey of block code operation in general and LDPC codes in particular is presented next. Within the LDPC section, both the history of these fascinating codes and a moderately detailed outline of the decoding algorithm for the codes are given. The chapter concludes with an overview of the WiMAX LDPC codes, which are used as a contemporary LLR-based ECC technology to demonstrate BERLL operation.

2.1 The Log-Likelihood Ratio

Both LDPC and Turbo codes use a common statistical measure that is manipulated and ideally strengthened in magnitude each decoding cycle named the log-likelihood ratio (LLR), which is defined for binary-valued random variables as follows.

[1] In particular, it is rather ironic to see the (re)derivation of equation (13) in [12] for parity-check codes, which is to within a logarithmic transform identical to Gallager's equation (6) in [3]—the fundamental binary parity-check decoding equation.

Definition 1. *Let x be a binary valued random variable from the set $\{-1, +1\}$. The* log-likelihood ratio *of x is defined as*

$$L(x) = \log\left(\frac{\Pr(x = +1)}{\Pr(x = -1)}\right). \tag{2.1}$$

The log-likelihood ratio of x is also called the L-value of x. Note that the definition of $L(x)$ requires no additional statistical conditioning on other variables, and depends only on the probability distribution of the random variable x. A graph of $L(x)$ versus $\Pr(x = +1)$ is shown in Figure 2.1. The graph is seen to have vertical asymptotes of $-\infty$ at $\Pr(x = +1) = 0$ and $+\infty$ at $\Pr(x = +1) = 1$, and a value of zero for $\Pr(x = +1) = 0.5$. This can be put into words by saying the log-likelihood of $x = +1$ when $\Pr(x = +1) = 0$ is infinitely *unlikely*, the log-likelihood of $x = +1$ when $\Pr(x = +1) = 1$ is infinitely likely, and the log-likelihood of $x = +1$ or $x = -1$ when $\Pr(x = +1) = 0.5$ is equally likely. An advantage of working with log-likelihood ratios over straight probabilities is evident at the algorithm implementation stage, since products of probabilities are replaced with summations of LLRs, and thus resource-costly multiplication operations are replaced with simple additions.

2.2 Estimating Bit Error Rate from Log-Likelihood Ratio

Of central importance to BERLL operation is the need to determine an estimate of bit error rate from a log-likelihood ratio value. Definition 1 can be restated in terms of the probability of bit error, P_e, of a decoded bit. Suppose the true value of a bit x received and

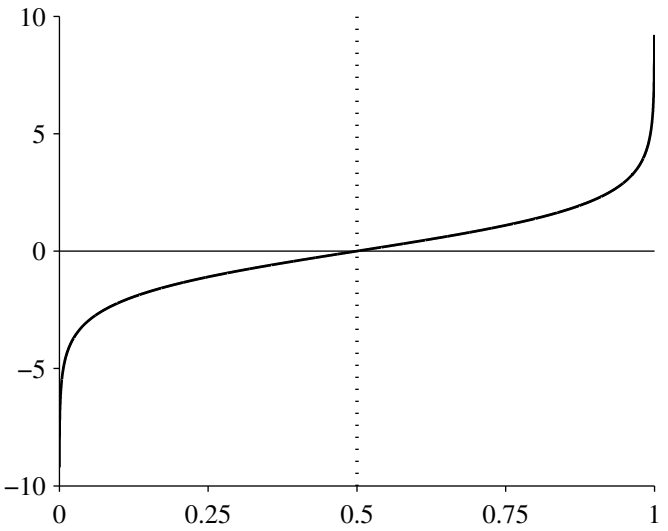


Figure 2.1: Log-likelihood Function.

decoded is $+1$, then the probability of x being face-value decoded correctly, without the aid of an error correction device, is $P_c = \Pr(x = +1) = 1 - P_e$, while the corresponding probability of decoding x incorrectly is simply P_e. The LLR of x is then expressed in terms of P_e as

$$L(x)|_{x=+1} = \log\left(\frac{1 - P_e}{P_e}\right). \tag{2.2}$$

Alternatively, suppose the true value of x is -1, then in this case

$$P_c = \Pr(x = -1) = 1 - \Pr(x = +1)\,,$$

while $P_e = \Pr(x = +1)$ which gives

$$L(x)|_{x=-1} = \log\left(\frac{P_e}{1 - P_e}\right) = -\log\left(\frac{1 - P_e}{P_e}\right) = -L(x)|_{x=+1}. \tag{2.3}$$

It is then clear that the magnitude of $L(x)$ is the same in both cases, and is often referred to as the *reliability* or *confidence value* of x in the log-likelihood ratio domain [18].

The foregoing discussion leads to the following two conversely related definitions.

Definition 2. *Let x be a binary-valued random variable from the set $\{-1, +1\}$ and P_e be the probability that x is in error after being subjected to an additive noise process. The* reliability *of x as a function of P_e is expressed as*

$$|L(x)| = \left| \log\left(\frac{1 - P_e}{P_e} \right) \right|. \tag{2.4}$$

Definition 3. *Let x be a binary-valued random variable from the set $\{-1, +1\}$ having log-likelihood ratio $L(x)$. The probability that x is in error after being subjected to an additive noise process expressed as a function of $L(x)$ is*

$$P_e = \frac{1}{1 + e^{|L(x)|}}. \tag{2.5}$$

Definition 2 is significant to a BERLL from the standpoint of converting an application-specific BER target, P_e, to a corresponding required LLR to meet the BER target. This conversion will play a key role in defining the error term in the closed loop circuit, which will compare the real-time LLR decoder output to an LLR value defined from a target P_e. The relationship between P_e and $L(x)$ is especially advantageous since it provides a means of using $L(x)$, a native variable already utilized for normal decoder operation, to produce an estimate of the decoder output BER. Otherwise, supplementary BER measurement devices would be required to assist a BERLL circuit; in this regard the LLR-processing decoder and

BERLL are self-sufficient. Note that for ordinary decoder operation, $0 < P_e < 0.5$, and thus the logarithm in Definition 2 is strictly positive. Definition 3 gives a formula whereby decoder error probability can be estimated from a functional transform of the decoder LLR output, which can then be monitored and compared to the target P_e throughout decoder operation.

2.3 Block Codes

This section briefly reviews block code theory, which will facilitate the discussion of LDPC codes in the next section, since LDPC codes are a special subclass of block codes. The basic premise behind error correcting codes follows from Shannon's noisy channel coding theorem, in that a sequence of uncoded message symbols can be associated with a codeword in such a way that the redundancy in the codeword (referred to as parity symbols), after transmission through a noisy channel, can be used to improve the probability of successfully decoding the original message at a receiver.

In traditional notation, a linear (n, k) code $\mathbf{C}$ transforms a message of length k symbols to a codeword of length n symbols. The code $\mathbf{C}$ thus adds $m = n - k$ additional symbols to the original message; Figure 2.2 shows an illustration of this process. These m additional symbols will allow a decoder to inverse transform, or *decode* a partially errored received codeword into the proper message with a lower probability of error than the uncoded message, where the codeword has been corrupted by one or more noise sources.

Associated with a linear code is a generator matrix $\mathbf{G}$ having dimension $k \times n$, which can be used to transform a $1 \times k$ message vector $\mathbf{m}$ into a $1 \times n$ codeword vector $\mathbf{c}$ by

matrix multiplication, $\mathbf{mG} = \mathbf{c}$. The rows of $\mathbf{G}$ can be expressed as n-tuple vectors, $\mathbf{g}_i$, $i = 1, 2, ..., k$. The vectors $\mathbf{g}_i$ are said to *span* the vector space associated with the code $\mathbf{C}$. When the k vectors $\mathbf{g}_i$ are linearly independent, the code $\mathbf{C}$ is k-dimensional, and is a proper vector subspace of a larger n-dimensional vector space. Thus there exists a set of $n - k$ linearly independent vectors that form the basis vector set of the vector space $\mathbf{C}^{\perp}$, the dual code of $\mathbf{C}$. From the properties of vector spaces and their duals, it follows that if $\mathbf{c}$ is a codeword from $\mathbf{C}$, and the $(n - k) \times n$ matrix $\mathbf{H}$ is the generator matrix for the dual code $\mathbf{C}^{\perp}$, then necessarily $\mathbf{cH}^{\mathsf{T}} = \mathbf{0}$. In general this last vector-matrix product resolving to the $1 \times (n - k)$ zero vector forms the basis for parity-checking $\mathbf{c}$ to test whether it is a valid codeword. The $n - k$ equations formed by multiplying $\mathbf{c}$ and $\mathbf{H}$ are correspondingly referred to as *parity-check* equations, and $\mathbf{H}$ is the *parity-check matrix* for $\mathbf{C}$. If the parity-check matrix $\mathbf{H}$ contains a considerably small number of nonzero elements relative to the total size of the matrix, then $\mathbf{H}$ is generally referred to as having *low density*, and $\mathbf{C}$ is referred to as a low-density parity-check (LDPC) code.

When $\mathbf{G}$ and $\mathbf{H}$ are binary and of full rank, the code rate R will be exactly

$$R = \frac{\log_2 2^k}{\log_2 2^n} = \frac{k}{n}. \tag{2.6}$$

However, for many parity-check codes where a general system of parity checks defines the code, the parity-check matrix $\mathbf{H}$ is not full rank; that is, $\mathrm{rank}(\mathbf{H}) < n - k$. Consequently the code rate of $\mathbf{C}$ will be less than k/n,

$$R = \frac{\log_2(\text{number of codewords in } \mathbf{C})}{n} < \frac{k}{n}. \tag{2.7}$$

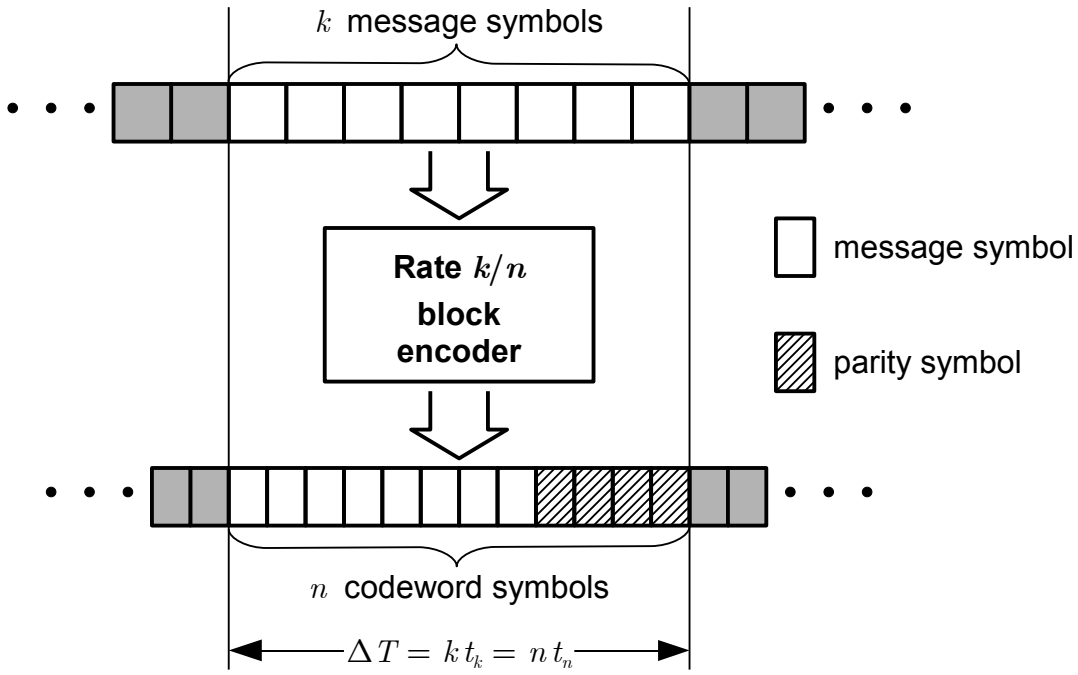


Figure 2.2: Encoder diagram for a rate k/n block code.

However, the *design rate* of the code is still considered to be k/n, primarily since the block encoder will continue to break the message stream up into k-bit blocks and append $n-k$ redundancy bits onto each block to create n-bit codewords.

The basic encoding procedure for a rate $R = k/n$ block code is shown pictorially in Figure 2.2. A k-bit long message is shown entering the encoder, which produces an n-bit long codeword destined for the next transmission module (e.g., channel modulator). The hatch-patterned bits in the codeword correspond to the $n-k$ parity bits associated with the message. At the bottom of Figure 2.2 is a timescale showing the amount of time occupied by k message bits, $\Delta T = kt_k = nt_n$. Since ΔT is a constant and $n > k$, then necessarily $t_n < t_k$, which has a direct impact on the signal-to-noise ratio (SNR) associated with a codeword bit, compared to a message bit. The SNR level in turn impacts the amount of

error correction, typically measured as bit error rate improvement, that the ECC decoder can provide.

2.4 Low-Density Parity-Check Codes

2.4.1 Historical Perspective

This section gives a brief overview of the origins, history and recent trends of LDPC codes, followed by a review of the sum-product decoding algorithm (SPA) for these codes. In 1962 Robert Gallager formally introduced LDPC codes, defined and characterized by the parity-check matrix of the code as opposed to the generator matrix [3], [4]. In particular, the parity-check matrices of Gallager's codes had a relatively small number or density of ones compared to the number of zeros.

For nearly two decades, the ideas presented in Gallager's thesis received little attention. In [7], Tanner emphasized the bipartite graphical structure of LDPC and other codes, and proposed two recursive decoding algorithms that traded decoding complexity for decoding accuracy. Viewing LDPC codes as graphical objects led to further analysis of the codes in recent research in terms of their graph theoretic properties, for example the girth of the graph corresponding to a code. In general, code performance was found to improve with increasing code girth. Additionally, research effort was expended to reduce the number of short cycles in the code graph, since short cycle reduction was also found to yield improved code performance.

The error correction potential of LDPC codes was not truly explored nor appreciated for another two decade time span after Tanner's research. In [5] and [6], MacKay and his colleagues pursued methods and algorithms whereby the parity-check matrix was found with constraint-guided search methods. The error correction performance of a candidate parity-check matrix produced by the search was then examined via simulation. Those matrices demonstrating better-than-average performance were noted and cataloged. While the LDPC codes generated using these search techniques showed excellent error-correcting ability relative to other error correcting code families such as Turbo and Reed Solomon codes, error analysis and implementation of these almost randomly-generated codes was no trivial task. In [8], Kou and others conducted bit error rate performance analysis on codes generated from finite geometry structures, a design method that was also described in Tanner's research. The error correction potential demonstrated by both the heuristic parity-check matrix search method and finite geometry method re-ignited LDPC code research.

Both Gallager's and Tanner's parity-check matrix designs corresponded to regular bipartite graphs. In their models, codeword bits were regarded as one set of vertices having the same degree, while parity-check equations were regarded as a separate set of vertices having a common degree. In [9], a creative approach to LDPC code design was introduced based upon *irregular* bipartite graphs, where some codeword bits participated in more parity checks than others. This particular construction technique, along with the belief propagation iterative decoding algorithm, produced an LDPC code that came within fractions of a decibel to Shannon's theoretical limit, placing LDPC codes on the same performance tier as Turbo codes. Along with demonstrating codes having remarkable error correcting power, this same team of researchers also introduced the use of normalized graph degree

sequence polynomials as a means of parameterizing the underlying code–one polynomial corresponding to the codeword bit vertices and the other polynomial corresponding to the parity-check vertices. Polynomial coefficients could then be listed that produced codes having good performance.

Within the last decade, reducing the sizable gap separating LDPC codes exhibiting good error correcting performance from LDPC codes having reasonable implementation cost has been a prominent research objective. A performance versus implementation design space *coordinate* has emerged where the parity-check matrix is constructed using cyclic shifts of a square identity submatrix interspersed with square zero submatrices of the same size. The results have produced families of structured LDPC codes that have feasible encoder and decoder implementations and exhibit good error correction strength, and are emerging as error correction options in contemporary wireless communication standards [20], [21].

2.4.2 Belief Propagation Algorithm

This section summarizes the fundamental equations used to realize the iterative belief propagation algorithm (BPA), also referred to as the message passing algorithm (MPA) or the sum-product algorithm (SPA). The material in this section stems from Gallager's original thesis, and so is not claimed as unique, but is valuable for gaining insight into the mechanics of a BPA decoder. First, the log-likelihood ratio defined previously in Definition 1 is applied to the binary input, continuous-output AWGN (BI-AWGN) channel to determine the channel LLR of a received codeword bit, which is relied on as a statistically independent estimate for the bit and used to initialize the algorithm. Next, alternate proofs

are provided for Gallager's development of the parity-check probability and log-likelihood ratio equations, motivated in part by a tutorial exposition similar to the present appearing as a chapter in [10]. Finally, the previous equations are grouped together to form the iterative log-likelihood ratio belief propagation algorithm which continues to be used as the normative basis for LDPC decoding.

The BI-AWGN channel is one of the fundamental communication channel models used to study modulation and coding. In this model, a binary-valued random variable x is transmitted through the channel and subjected to a zero-mean Gaussian noise process having standard deviation σ, often denoted as the normal distribution $\mathcal{N}(0, \sigma^2)$. The output variable of the channel y will be drawn from the continuum of values in $(-\infty, +\infty)$, having a conditional probability density function (PDF) that depends upon the value of $x \in \{-1, +1\}$ transmitted and the noise deviation σ. The conditional PDF of the received value y as a function of transmitted x and channel σ is termed the *a priori* distribution and is given by

$$f_{Y|X}(y|x) = \frac{\exp\left(-\frac{(y-x)^2}{2\sigma^2}\right)}{\sqrt{2\pi}\sigma}. \tag{2.8}$$

In the literature the function subscript of $Y|X$ is omitted when the context is clear. In a typical digital communications text, (2.8) is manipulated to determine expressions for probability of bit error versus energy-to-noise ratio; however, in the present vein, it is useful for determining the channel *a posteriori* LLR for transmitted bit x conditioned on

the received y, starting from the basic LLR equation in Definition 1 as follows:

$$\begin{aligned} L_c &= L(x|y) \\ &= \log\left(\frac{\Pr(x=+1|y)}{\Pr(x=-1|y)}\right) \\ &= \log\left(\frac{\Pr(y|x=+1)\Pr(x=+1)}{\Pr(y)} \cdot \frac{\Pr(y)}{\Pr(y|x=-1)\Pr(x=-1)}\right) \\ &= \log\left(\frac{\Pr(y|x=+1)}{\Pr(y|x=-1)} \cdot \frac{\Pr(x=+1)}{\Pr(x=-1)}\right) \\ &= L(y|x) + L(x)\,, \end{aligned} \tag{2.9}$$

which agrees with the result in [12]. In the development above for L_c, the subscript c denotes *channel*, and the log-likelihood ratio $L(y|x)$ is the *a priori* channel LLR. If the random variable x is presumed to take values from $\{-1, +1\}$ with equal probability, then $L(x) = 0$ and the *a priori* and *a posteriori* LLRs are equivalent, making the channel LLR simply equal to the *a priori* LLR:

$$L_c = L(y|x). \tag{2.10}$$

By using the *a priori* formulation in (2.10), the conditional point-density in (2.8) can be utilized to determine L_c:

$$\begin{aligned} L_c &= \log\left(\frac{f(y|x=+1)}{f(y|x=-1)}\right) \\ &= \log\left(\frac{\exp\left(-\frac{(y-1)^2}{2\sigma^2}\right)}{\exp\left(-\frac{(y+1)^2}{2\sigma^2}\right)}\right) \\ &= \frac{(y+1)^2}{2\sigma^2} - \frac{(y-1)^2}{2\sigma^2} \\ &= \frac{2y}{\sigma^2}\,, \end{aligned} \tag{2.11}$$

which agrees with the result stated without proof in [10]. Thus for the BI-AWGN channel model, the received, demodulated sample for each codeword bit is to within a noise variance scaling the channel LLR itself, which is in a sense a stroke of good fortune from an implementation viewpoint. The quantity in (2.11) is the initial value used to seed the BP algorithm at the receipt of each new codeword. It is noted that the $2/\sigma^2$ scale factor is important to realizing full decoder error correction potential, which means some reasonably accurate measurement of SNR must be made.

The operation of both the LDPC encoder and decoder hinges on computing simple multi-bit modulo-2 additions, also called an exclusive-or (XOR). Since decoder operation is probabilistic in nature, it is of interest next to develop expressions for the probability that an XOR of a set of binary-valued random variables is either zero (even parity) or one (odd parity). Let $x_1, x_2, \ldots, x_n$ be a set of n statistically independent binary-valued random

variables and p_i, $i = 1, 2, \ldots, n$ be the probability that x_i is 1. Gallager proved in [3] that

$$\Pr(x_1 \oplus x_2 \oplus \cdots \oplus x_n = 1) = \frac{1}{2} - \frac{1}{2}\prod_{i=1}^{n}(1 - 2p_i) \tag{2.12}$$

using an error polynomial method. For the purpose of exposition, the same result is alternatively verified here using mathematical induction, as suggested in [10]. Throughout the inductive proof, the *traditional* binary set $\{0, 1\}$ is chosen over the isomorphic set $\{-1, +1\}$, due to the usual association of the former with digital logic operations such as the XOR function, whereas the latter is typically used when dealing with real-valued signals. When the set $\{0, 1\}$ is regarded as sign bits of binary words, in most binary number representations 0 is mapped to $+1$ (positive values), while 1 is mapped to -1 (negative values).

As the basis step, $n = 2$ is chosen so that $\Pr(x_1 \oplus x_2 = 1)$ is sought for binary-valued, statistically independent random variables x_1 and x_2, where $p_1 = \Pr(x_1 = 1)$ and $p_2 = \Pr(x_2 = 1)$. Defining y as the binary-valued random variable

$$y = x_1 \oplus x_2\,, \tag{2.13}$$

we seek expressions for both $\Pr(y = 1)$ and $\Pr(y = 0)$ as functions of p_1 and p_2. Using the logical definition of the XOR function and the fact that x_1 and x_2 are statistically

independent,

$$\begin{aligned}
\Pr(y=1) &= \Pr(x_1 \oplus x_2 = 1) \\
&= \Pr\big(\big[(x_1=0)\cap(x_2=1)\big] \cup \big[(x_1=1)\cap(x_2=0)\big]\big) \\
&= \Pr(x_1=0)\cdot\Pr(x_2=1) + \Pr(x_1=1)\cdot\Pr(x_2=0) \\
&= (1-p_1)p_2 + p_1(1-p_2) \\
&= p_1 + p_2 - 2p_1p_2 \\
&= \frac{1}{2} - \frac{1}{2}(1 - 2p_1 - 2p_2 + 4p_1p_2) \\
&= \frac{1}{2} - \frac{1}{2}(1-2p_1)(1-2p_2) \\
&= \frac{1}{2} - \frac{1}{2}\prod_{i=1}^{2}(1-2p_i)\,, \qquad (2.14)
\end{aligned}$$

which is the 2-term form of (2.12). Solving for $\Pr(y=0)$ is accomplished simply by subtracting (2.14) from one, which gives

$$\Pr(y=0) = \Pr(x_1 \oplus x_2 = 0) = \frac{1}{2} + \frac{1}{2}\prod_{i=1}^{2}(1-2p_i). \qquad (2.15)$$

For the induction step, presume a set of $n-1$ statistically independent, binary-valued random variables $x_1, x_2, \ldots, x_{n-1}$ that are combined into an $(n-1)$-term XOR function,

$$z = x_1 \oplus x_2 \oplus \cdots \oplus x_{n-1}\,, \qquad (2.16)$$

where as before $p_i = \Pr(x_i = 1)$ for $i = 1, 2, \ldots, n-1$. Using the 2-term basis step result, it is assumed that the $(n-1)$-term form of (2.12) holds:

$$\Pr(z = 1) = \frac{1}{2} - \frac{1}{2}\prod_{i=1}^{n-1}(1 - 2p_i). \tag{2.17}$$

An additional statistically independent binary random variable x_n having $p_n = \Pr(x_n = 1)$ is incorporated into (2.17), and a new random variable y is defined as

$$y = z \oplus x_n = [x_1 \oplus x_2 \oplus \cdots \oplus x_{n-1}] \oplus x_n. \tag{2.18}$$

The n-term XOR probability in (2.18) is reduced to a 2-term XOR probability evaluation:

$$\begin{aligned}
&\Pr(x_1 \oplus x_2 \oplus \cdots \oplus x_n = 1) \\
&\quad = \Pr(y = 1) \\
&\quad = \Pr(z \oplus x_n = 1) \\
&\quad = \frac{1}{2} - \frac{1}{2}\big[1 - 2\Pr(z = 1)\big](1 - 2p_n) \\
&\quad = \frac{1}{2} - \frac{1}{2}\prod_{i=1}^{n-1}(1 - 2p_i)(1 - 2p_n) \\
&\quad = \frac{1}{2} - \frac{1}{2}\prod_{i=1}^{n}(1 - 2p_i)\,,
\end{aligned} \tag{2.19}$$

which completes the proof. From (2.19) it is also easy to compute the probability of the n-term XOR function being zero,

$$\Pr(x_1 \oplus x_2 \oplus \cdots \oplus x_n = 0) = \frac{1}{2} + \frac{1}{2}\prod_{i=1}^{n}(1 - 2p_i). \tag{2.20}$$

Having developed expressions for the probability of a multi-bit XOR being either zero or one, the LLR of the multi-bit XOR can be calculated. The resulting equation will be the LLR of a parity-check of an arbitrary number of binary-valued random variables, which will be used to evaluate extrinsic log-likelihood ratios in the LDPC decoding algorithm. The parity-check LLR takes advantage of the following equivalent expressions that convert p_i to and from $L_i = L(x_i)$, for a given random variable x_i, $i = 1, 2, \ldots, n$, which are easily verified:

$$1 - 2p_i = \frac{e^{L_i} - 1}{e^{L_i} + 1} = \tanh \frac{L_i}{2}. \tag{2.21}$$

Using the results of (2.19), (2.20) and (2.21), the parity-check LLR of the random variable $y = x_1 \oplus x_2 \oplus \cdots \oplus x_n$ can be written as

$$\begin{aligned} L(y) &= \log \left(\frac{\Pr(y = 0)}{\Pr(y = 1)} \right) \\ &= \log \left(\frac{\frac{1}{2} + \frac{1}{2} \prod_{i=1}^{n} (1 - 2p_i)}{\frac{1}{2} - \frac{1}{2} \prod_{i=1}^{n} (1 - 2p_i)} \right) \\ &= \log \left(\frac{1 + \prod_{i=1}^{n} \tanh \frac{L_i}{2}}{1 - \prod_{i=1}^{n} \tanh \frac{L_i}{2}} \right). \end{aligned} \tag{2.22}$$

Exponentiating both sides of (2.22) and rearranging the result to solve for the product of bit-wise hyperbolic tangent functions in terms of $L(y)$ gives

$$\prod_{i=1}^{n} \tanh \frac{L_i}{2} = \frac{e^{L(y)} - 1}{e^{L(y)} + 1} = \tanh \frac{L(y)}{2}, \tag{2.23}$$

which can be further separated into sign and magnitude portions since the tanh function has odd symmetry:

$$\prod_{i=1}^{n} \text{sign}(L_i) \cdot \prod_{i=1}^{n} \tanh \frac{|L_i|}{2} = \text{sign}(L(y)) \cdot \tanh \frac{|L(y)|}{2}. \tag{2.24}$$

The natural logarithm is next applied to the magnitude component of both sides of (2.24), followed by a sign inversion:

$$\sum_{i=1}^{n} -\log\Big(\tanh \frac{|L_i|}{2}\Big) = -\log\Big(\tanh \frac{|L(y)|}{2}\Big). \tag{2.25}$$

The function $\phi(x) = -\log(\tanh(x/2))$, often referred to as *Gallager's ϕ-function* in literature, is its own inverse for positive arguments, and is plotted in Figure 2.3. The positive argument inverse property of ϕ is utilized in (2.25) to express the magnitude of $L(y)$ as a function of L_i magnitudes,

$$|L(y)| = \phi\Big(\sum_{i=1}^{n} \phi(|L_i|)\Big). \tag{2.26}$$

Combining (2.26) with the product-of-signs component in (2.24) produces the desired expression for the LLR of an n-bit XOR in terms of the LLRs of individual bits:

$$L(y) = \text{sign}(L(y)) \cdot |L(y)| = \prod_{i=1}^{n} \text{sign}(L_i) \cdot \phi\Big(\sum_{i=1}^{n} \phi(|L_i|)\Big), \tag{2.27}$$

which serves as a proof of Gallager's result in [3].

In a generic parity check matrix, an individual codeword bit vertex (or variable node) will be adjacent to multiple parity check vertices (or check nodes). Likewise, an individual

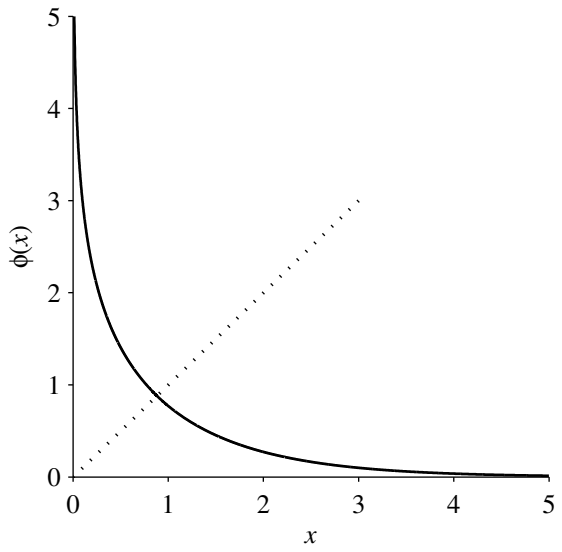


Figure 2.3: The function $\phi(x) = -\log(\tanh(x/2))$ for $x > 0$.

check vertex will be adjacent to multiple variable vertices; additionally each bit vertex is graphically connected with a single edge to the channel estimate LLR. Figure 2.4(a) illustrates this arrangement, and will be used to describe the BP algorithm. In the figure, bit nodes are represented as squares (bearing resemblance to a flip-flop device) while parity check nodes are represented as circles (bearing resemblance to a circular XOR operator symbol). As an example, Figure 2.4(b) shows the bipartite graph corresponding to the (7,4) Hamming code.

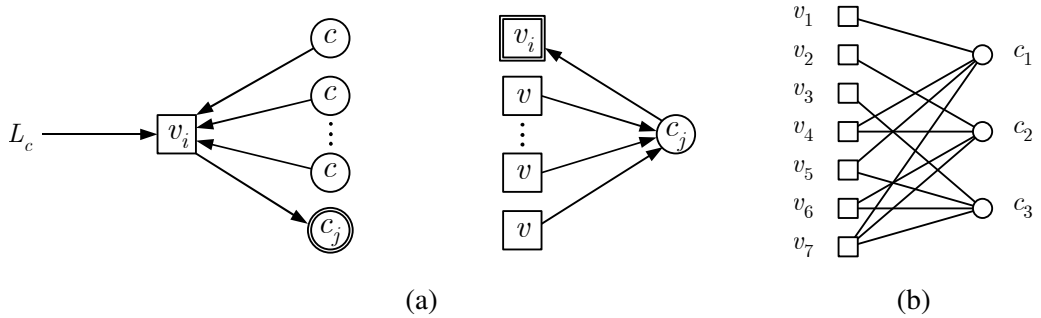


Figure 2.4: (a) Variable and check node message passing graph illustrations; (b) Bipartite (Tanner) graph for the (7,4) Hamming code.

A single decoder iteration for an LDPC decoder can be decomposed into a check node update followed by a variable node update. For the check node update, each check node computes an individualized extrinsic LLR *message* to send each adjacent variable node using a $(d-1)$-term parity check LLR formulation of (2.27), where d is the degree of check node c_j, $d = \deg c_j$. This procedure corresponds to the right-hand side of Figure 2.4(a) where check node c_j takes the individual variable node LLRs from all variable nodes other than v_i, applies a $(d-1)$-term computation according to (2.27), and passes the result back to v_i, which becomes the check node extrinsic LLR from check node j to variable node i. Thus a total of $\binom{d}{d-1} = d$ unique LLR messages must be computed for c_j alone.

For the variable node update, each variable node computes an individualized extrinsic LLR message to send to adjacent check nodes, as illustrated in the left-hand graph of Figure 2.4(a). As indicated in the figure, variable node v_i combines the LLR messages from check nodes other than c_j (sent to v_i during the previous half-iteration) with the channel LLR value for v_i and sends the resulting extrinsic variable node LLR back to c_j for use in the next iteration. The arithmetic required for the variable node update is a simple summation of LLR values. The LLR estimate of variable node v_i is also straightforward to compute, and is simply the sum of all extrinsic variable node LLR values passed to v_i plus the channel LLR.

In light of the preceding discussion, the iterative message passing algorithm is summarized as follows. Let variable node v_i and check node c_j be adjacent in the graph of a code. Define index sets J_i and I_j as

$J_i =$ set of indexes of all check nodes adjacent to v_i

I_j = set of indexes of all variable nodes adjacent to c_j.

Initialize each variable node extrinsic LLR to $L_e(v_i) = L_c(v_i)$ using (2.11).

For each iteration:

1. For $j \in J_i$, compute the check node extrinsic LLR

$$L_e(c_j) = \prod_{k \in I_j \backslash i} \text{sign}(L_e(v_k)) \cdot \phi\Big(\sum_{k \in I_j \backslash i} \phi(|L_e(v_k)|) \Big).$$

2. For $i \in I_j$, compute the variable node extrinsic LLR

$$L_e(v_i) = L_c(v_i) + \sum_{k \in J_i \backslash j} L_e(c_k).$$

3. Repeat steps 1 and 2 for the prescribed number of iterations for all pairs of adjacent variable and check nodes, after which compute the total LLR

$$L(v_i) = L_c(v_i) + \sum_{k \in J_i} L_e(c_k).$$

The binary, hard decision for v_i is determined by examining the sign of $L(v_i)$, discarding the magnitude. The message passing algorithm as presented has received various levels of optimization in the literature to make it more suitable for hardware and software implementation, typically at the expense of accuracy and thus decoder error correction potential. The simulation code written for this dissertation to decode LDPC codewords uses the algorithm in its original form above to examine BERLL operation. It is worth noting here that the BERLL uses the discarded magnitudes $|L(v_i)|$ from step 3 above as an input to the loop to tune decoder performance, described in detail in Chapters 3 and 4 that follow.

2.5 Summary of IEEE 802.16 WiMAX LDPC Codes

Throughout this dissertation, the IEEE 802.16 WiMAX LDPC codes will be used as the *focus* codes for which numerical results are produced, and for which practical BERLL operation will be demonstrated. The family of WiMAX codes has a highly-refined parity-check matrix construction intended to facilitate encoder and decoder implementation. A handful of code rates R and numerous block lengths n are available within the code set. Among the code rates, rate 1/2, 2/3 (2 variants, denoted 2/3A and 2/3B), 3/4 (2 variants, denoted 3/4A and 3/4B) and 5/6 codes are available. For each code rate, including variants, a unique *base matrix* of integer entries $P \in \{-1, 0, 1, 2, \ldots, 95\}$ is defined. The base matrix corresponding to a code rate is used along with an integer $z \in \{24, 28, \ldots, 96\}$ to define a specific (n, k) binary-valued parity-check matrix; this results in a total of $6{\times}19 = 114$ different codes available in the standard. Each of the base matrices has 24 columns and an appropriate number of rows based upon the code rate. When expanding a base matrix into its equivalent binary form, each of the index values in the base matrix is replaced with either a $z{\times}z$ zero matrix or cyclic shifts (row or column permutations) of the $z{\times}z$ identity matrix, $\mathbf{I}_{z\times z}$. Zero matrix substitution occurs for -1 index entries, while cyclic shifts of $\mathbf{I}_{z\times z}$ occur for nonnegative entries. For nonnegative indexes the number of shifts applied to $\mathbf{I}_{z\times z}$ is $P_{ij} \mod z$, where P_{ij} is the index at row i, column j in the base matrix.

The lowest-performance WiMAX LDPC code uses the rate 5/6 base matrix, shown below, and has $z = 24$ which gives a (576, 480) code:

$$\mathbf{H}_{b,5/6} = \begin{pmatrix} 1 & 25 & 55 & -1 & 47 & 4 & -1 & 91 & 84 & 8 & 86 & 52 & 82 & 33 & 5 & 0 & 36 & 20 & 4 & 77 & 80 & 0 & -1 & -1 \\ -1 & 6 & -1 & 36 & 40 & 47 & 12 & 79 & 47 & -1 & 41 & 21 & 12 & 71 & 14 & 72 & 0 & 44 & 49 & 0 & 0 & 0 & 0 & -1 \\ 51 & 81 & 83 & 4 & 67 & -1 & 21 & -1 & 31 & 24 & 91 & 61 & 81 & 9 & 86 & 78 & 60 & 88 & 67 & 15 & -1 & -1 & 0 & 0 \\ 68 & -1 & 50 & 15 & -1 & 36 & 13 & 10 & 11 & 20 & 53 & 90 & 29 & 92 & 57 & 30 & 84 & 92 & 11 & 66 & 80 & -1 & -1 & 0 \end{pmatrix}.$$

To appreciate the size and low-density (sparseness) of the expanded 96×576 binary parity-check matrix for this code, a bitmap pictorial representation is shown in Figure 2.5, where 1-entries are represented as small pixels, and 0-entries are left blank; the 24×24 cyclic shift submatrices are easily spotted in the figure. It is also worth noting that no whitespace was included around pixels in Figure 2.5.

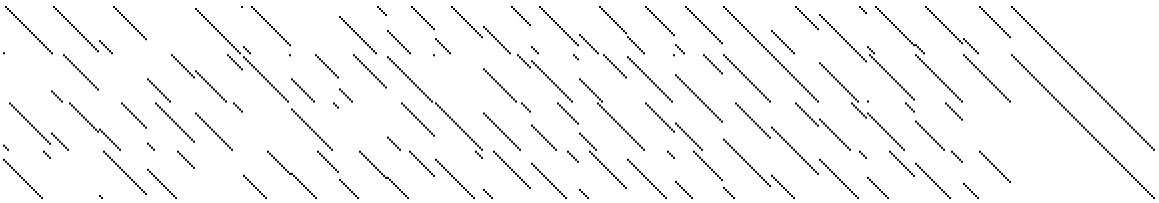


Figure 2.5: Bitmap representation of the rate 5/6, (576, 480)
WiMAX LDPC parity-check matrix (1 pixel = 1 matrix entry).

At the other end of the performance spectrum in the WiMAX code family is the highest-performance code, which uses the rate 1/2 base matrix, shown below, and has $z = 96$ which

gives a (2304, 1152) code:

$$\mathbf{H}_{b,1/2} = \begin{pmatrix}
-1 & 94 & 73 & -1 & -1 & -1 & -1 & -1 & 55 & 83 & -1 & -1 & 7 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & 27 & -1 & -1 & -1 & 22 & 79 & 9 & -1 & -1 & -1 & 12 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 24 & 22 & 81 & -1 & 33 & -1 & -1 & -1 & 0 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
61 & -1 & 47 & -1 & -1 & -1 & -1 & -1 & 65 & 25 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & 39 & -1 & -1 & -1 & 84 & -1 & -1 & 41 & 72 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & 46 & 40 & -1 & 82 & -1 & -1 & -1 & 79 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & 95 & 53 & -1 & -1 & -1 & -1 & -1 & 14 & 18 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\
-1 & 11 & 73 & -1 & -1 & -1 & 2 & -1 & -1 & 47 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 \\
12 & -1 & -1 & -1 & 83 & 24 & -1 & 43 & -1 & -1 & -1 & 51 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & 94 & -1 & 59 & -1 & -1 & 70 & 72 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 \\
-1 & -1 & 7 & 65 & -1 & -1 & -1 & -1 & 39 & 49 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 \\
43 & -1 & -1 & -1 & -1 & 66 & -1 & 41 & -1 & -1 & -1 & 26 & 7 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0
\end{pmatrix}.$$

A pictorial representation for the expanded 1152×2304 parity-check matrix is shown in Figure 2.6; it is noted that in the figure each 1-entry has been replicated three times in adjacent pixels for clarity, thus the actual density of the matrix is less than the visual density in the figure.

A feature worth noting in the parity-check matrices, visible as a staircase of zero indexes in $\mathbf{H}_{b,5/6}$ and $\mathbf{H}_{b,1/2}$ and as long major-diagonals in the bitmaps, is an upper-triangular submatrix on the right-hand side of the parity-check matrices. The column labeling for this submatrix corresponds to the message bit positions (thus these LDPC codes are systematic). The diagonally parallel pair of zero-shifted identity matrices allows for a low-complexity encoding process, in which message bits appear as lone, successive pairs when traversing the parity-check equations in order from top-to-bottom or bottom-to-top. This structure allows for a successive variable substitution algorithm to solve for unknown parity bits

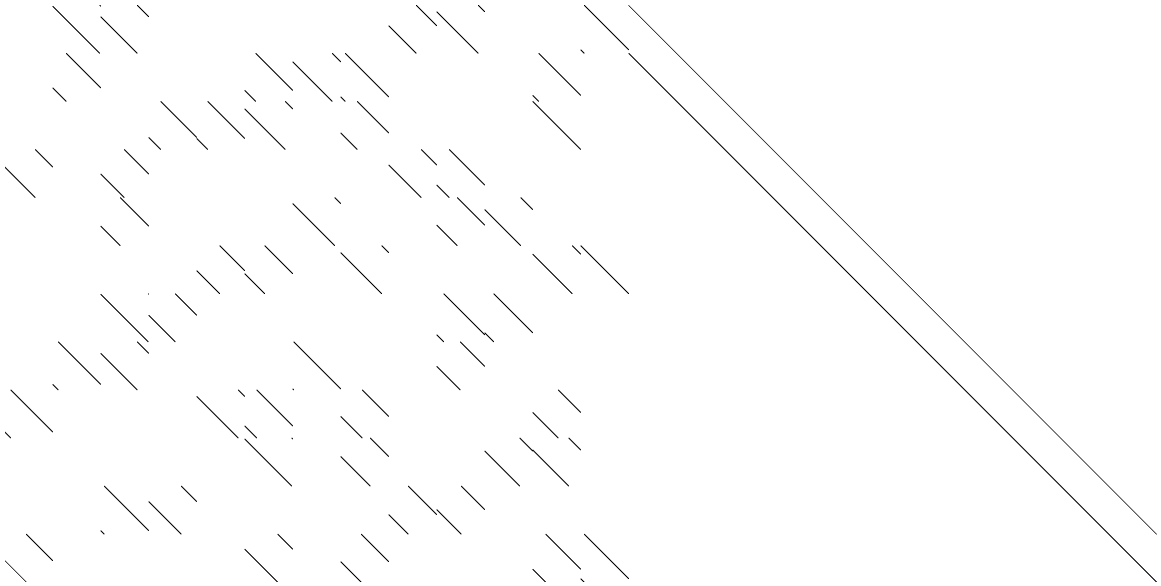


Figure 2.6: Bitmap representation of the rate 1/2, (2304, 1152)
WiMAX LDPC parity-check matrix (2×2 pixels = 1 matrix entry).

during encoding, similar in function to the triangular matrix decomposition method used to simplify the solution of a system of linear equations in matrix algebra.

Chapter 3

Bit Error Rate Locked Loops

3.1 Introduction

The bit error rate locked loop, or BERLL, from a black-box perspective must be able to measure the rate at which bit errors occur in a communications receiver and modify one or more modules used in the communication system to meet a bit error rate requirement. The maximum bit error rate allowed is typically application-specific, and for many communication systems is determined by a standards committee based upon the type of data being transferred and the associated quality level deemed acceptable. Communication systems handling digitized voice will ordinarily specify a less-stringent bit error rate requirement, since the human ear-brain combination is itself a very error-tolerant device, and can heuristically correct errors; additionally, the data bandwidth needed for voice communication is relatively small, on the order of tens-of-kHz. For example, CCITT recommendation G.821 [22] defines a severely errored second (SES) condition as an error rate of 10^{-3} over a one second time duration, while an error rate of 10^{-6} is considered suitable for general-purpose voice transmission [23]. In contrast, digital communication systems transporting high-speed data streams between two computers normally specify a much lower (higher-

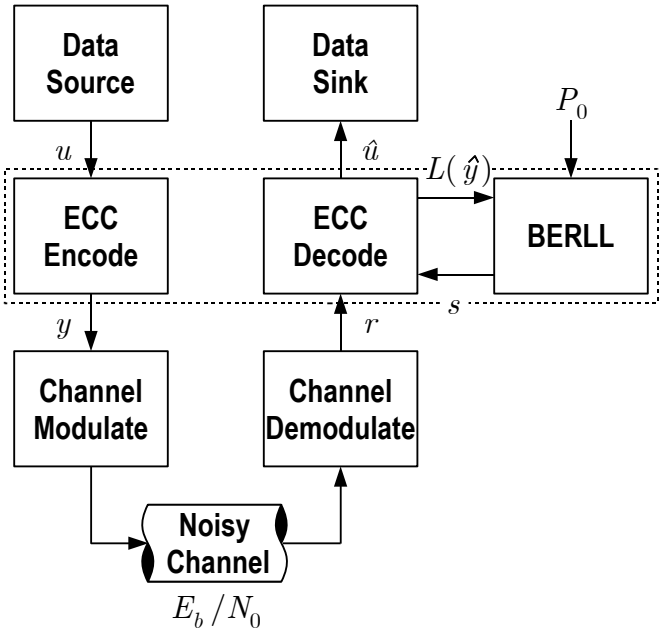


Figure 3.1: Error correction system with bit error rate locked loop feedback controller attachment.

stringent) error rate than voice communications since a bit error incurred by a data file during transport could render the file unusable at the receiving computer.

Figure 3.1 shows the basic components of a simple communications system and the position of the ECC encoder, decoder and BERLL highlighted with a dashed box within the system. In the figure, u is the uncoded data stream produced by the data source, while $\hat{u}$ is the error-prone decoded data stream estimate of u delivered to the data sink. The codeword corresponding to u is y, which is modulated and transmitted with E_b Joules per bit through a noisy channel having noise power spectral density N_0 Watts/Hz. At the receiving end of the channel, the signal is first demodulated and then presented (with possible errors) for decoding at the ECC decoder as r. As the iterative decoder develops estimates of y and u from r, (namely $\hat{y}$ and $\hat{u}$ respectively), the LLR of each codeword bit in $\hat{y}$ gradually strengthens in magnitude. The BERLL processes the decoder LLR outputs $L(\hat{y})$ for each

codeword, and produces a decoder parameter state, denoted as s, to configure the decoder for the next codeword in order to meet the target BER $P_e = P_0$. The signal processing required to convert decoder output LLR values to a BER estimate is examined in the error detector section of this chapter.

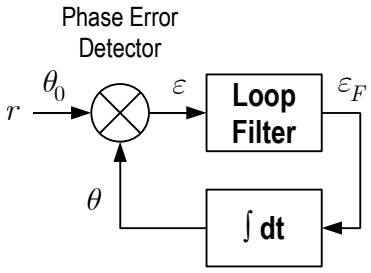


Figure 3.2: Traditional phase locked loop.

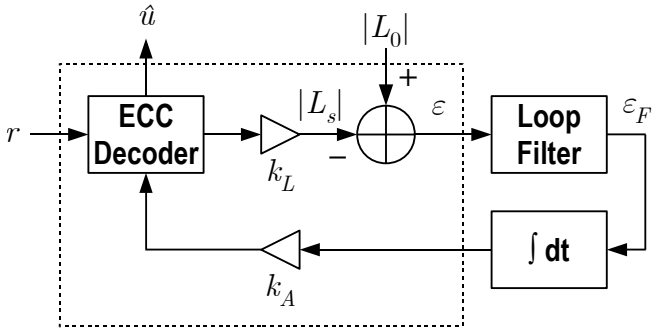


Figure 3.3: Proposed bit error rate locked loop.

From a component perspective, the BERLL has several elements in common with the venerable phase locked loop (PLL). Due to the frequent use of PLLs in communication systems, it was considered the natural model from which to draw when constructing a BERLL. To further establish the analogy between the PLL and the BERLL and to gain a more intuitive feel for how a BERLL operates, Figures 3.2 and 3.3 show both structures juxtaposed with common components highlighted. The remaining subsections of this chapter explore each of the subcomponents of the BERLL, and relate them to the PLL as a normative reference. Special attention will be given to the component that generates the error signal, ε, in Figure 3.3 since it is perhaps the truly novel component in the loop, all other components being somewhat standard and conceptually off-the-shelf in a manner of speaking. Before taking a tour of the loop elements, a brief discussion regarding the dynamic operation of the

decoder is presented since this is the underlying mechanism whereby a statically configured decoder is turned into an adjustable device.

3.2 Dynamic Decoder Operation

The error correction capability of an ECC decoder given a set of transmission channel conditions is well-characterized by a plot of the probability of error present on decoded message bits output from the decoder, P_e, as a function of channel bit energy-to-noise ratio, E_b/N_0. The curvature of a semi-logarithmic plot of P_e versus E_b/N_0 resembles a waterfall, and as a result is referred to as an ECC *waterfall curve*. For an ECC that can have various parameters adjusted, a unique set of waterfall plots will exist that characterize the performance of the decoder as a function of all possible parameter states. The term *subcode* is used to refer to a decoder configured in a specific parameter state. Higher performance subcodes exhibit waterfall curves having steeper descent than those of lower performance subcodes beyond some minimum E_b/N_0 ratio. The set of all possible waterfall curves over the range of E_b/N_0 levels incurred by the decoder is used to define the decoder *operating range*, while the pairing of a specific channel E_b/N_0 with a required P_e is used to define the decoder *operating point*. The operating range and operating point of a decoder are discussed in the next two subsections.

3.2.1 Decoder Operating Range

The concept of what is termed the decoder *operating range* is now described. An ECC decoder will have a minimum and maximum energy-to-noise ratio, E_b/N_0, over which it

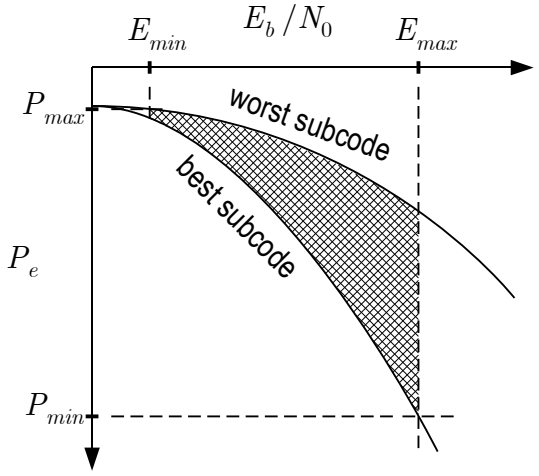


Figure 3.4: Bit error rate locked loop decoder operating range illustration.

will operate; the minimum E_b/N_0 will correspond to some value at which the receiver is just able to make reasonably accurate decisions on the received data (as an example), while the maximum E_b/N_0 will be at most the value at which the receiver saturates and begins to clip the signal-plus-noise resultant (typically occurring as front end analog overload in RF systems). The ECC decoder will exhibit large probability of error at low E_b/N_0 and small probability of error at high E_b/N_0. Thus for low E_b/N_0, a properly-designed BERLL will select a high-performing subcode, while at high E_b/N_0 the BERLL will select a low-performing subcode. This range of operation is depicted graphically using the decoder waterfall plot, in Figure 3.4.

The shaded area is domain-bounded by the minimum and maximum E_b/N_0 ratios over which the decoder will operate, while range-bounded by the probability of errors admitted by the weakest and strongest subcodes at these two E_b/N_0 endpoints. The bounded area graphically represents the full operating range possible for the decoder; in essence all decoder dynamic subcode switching and resulting performance will be constrained to fall within this operating range. Between the best and worst subcode waterfall curves are

some number of intra-performance subcode waterfalls. As an example, if the number of decoder iterations is used to dynamically adjust decoder performance, then there will be a discrete, finite number of subcodes from which to choose; the best code will correspond to the maximum number of iterations that can be practically allowed, while the worst code will correspond to one decoder iteration.

3.2.2 Decoder Operating Point

When a fixed P_e target is to be met, a horizontal slice through the set of subcode waterfall curves is traversed by the BERLL as the E_b/N_0 channel level varies. At any particular instant in time, the present E_b/N_0 level paired with the P_e objective becomes the decoder *operating point*. Figure 3.5 illustrates this concept, where the operating point is positioned at coordinate (E_0, P_0). The intersection of the vertical $E_b/N_0 = E_0$ dashed line with each subcode is shown as a hollow circle in Figure 3.5, and represents the probability of bit error that each subcode will admit for $E_b/N_0 = E_0$. For each decoder subcode waterfall curve, the difference between the required BER P_0 and attainable BER is represented graphically as vertical arrows originating at the $P_e = P_0$ dashed horizontal line and terminating at solid horizontal lines intersecting each subcode BER waterfall curve at $E_b/N_0 = E_0$; under-performing subcodes relative to P_0 have vertical arrows pointing upward, while over-performing subcodes have vertical arrows pointing downward. The BER locked loop should be designed to select the subcode having an error rate that meets or is just smaller than the target BER for a given E_b/N_0; in other words, when the loop

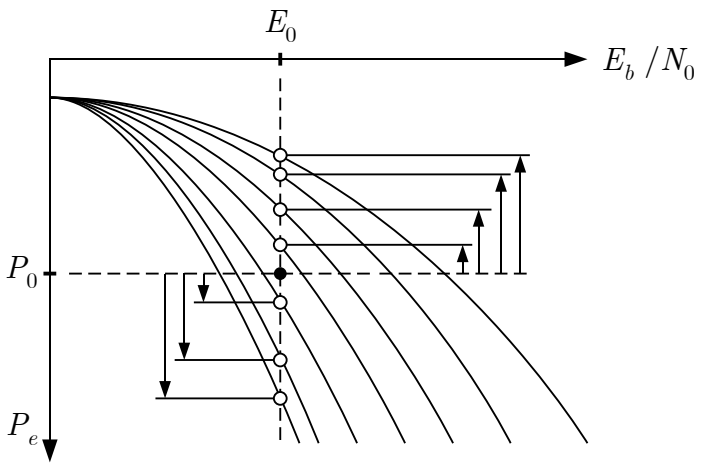


Figure 3.5: Bit error rate locked loop decoder operating point illustration.

reaches steady state, an under-performing subcode should never be selected so long as an over-performing subcode option is available for a given E_b/N_0 ratio.

3.3 Error Detector

The PLL and BERLL must both be able to make a comparison between a nominal signal, or reference, and an internal signal generated by the loop. In general terms, the difference between the nominal signal and generated signal is calculated by a device called an error detector. For the PLL, the error signal of interest is the difference between the phase of a reference signal, θ_0, and the phase θ of a loop-generated signal, as shown previously in Figure 3.2. The error detector in this case measures phase error, and is symbolized by a generic multiplier symbol in the figure. For the traditional PLL, the error term ε is thus ideally defined as $\varepsilon = \theta - \theta_0$. (Throughout this section, the terms *error comparator* and *error detector* will be used interchangeably.)

For the BERLL, defining the error term requires a bit more effort compared to the PLL, but is certainly tractable, and in fact is also the numerical difference between a nominal reference and loop-generated signal, as will be shown. The BERLL error detector must be able to compare a target BER, P_0, and the BER present in decoder output codeword bits, P_e. The numerical difference between P_e and P_0 would seem to be an intuitive definition for the BERLL error term ε, but is not well-suited for practical implementation. This can be explained in loose terms by observing that P_e can span several orders of magnitude over the range of E_b/N_0 levels the decoder will operate, which will in turn create a very large dynamic range on an arithmetic difference. Also worth considering is the large degree of nonlinearity when taking differences of exponentially-varying quantities, which is the case with decoder P_e values versus changing E_b/N_0 levels. It is also of foremost interest in this dissertation to utilize the fundamental arithmetic quantity used in modern iterative decoders—the log-likelihood ratio—to help construct a BERLL error signal. The remaining error detector discussion begins by examining the well-established concept of decoder stopping criteria, and develops two candidate BER error signals well-suited for use in the closed loop BERLL. The arithmetic required to generate the BER error signals becomes the BER error detector loop component, analogous to the phase error detector in a traditional PLL.

3.3.1 Decoder Stopping Criteria

A decoder stopping criteria is simply any method whereby the state of an error correction decoder, typically an iterative decoder, is evaluated for the purpose of suspending

further codeword decoding iterations to produce a hard-decision codeword estimate. As such, the established decoder stopping techniques are conceptually similar to the BER error signal, especially when the number of decoder iterations is used to modulate decoder performance. The similarity is established in that both the presently pursued BER error signal and stopping criteria permit a reduction in the number of decoder iterations needed to meet a target BER.

Several approaches to decoder stopping criterion have been studied. One of the first methods introduced, which continues to be a normative reference on the subject, is the cross entropy (CE) measure of Hagenauer [12]. In [13] the sign-change-ratio (SCR) and hard-decision-aided (HDA) techniques are described. The SCR method examines the number of sign bit changes in the extrinsic LLRs across decoded codeword bits from one iteration to the next, while the HDA method simply compares the hard decision (sign) bits of the output LLRs from one iteration to the next and stops decoding when no changes are detected. In [14] the magnitudes of Turbo decoder output LLRs are individually compared against upper and lower thresholds, with decoder iterations continuing so long as all magnitudes fall within the threshold limits; the corresponding method is labeled measurement of reliability (MOR). While the CE, SCR, HDA and MOR techniques are suitable as measures for determining when to halt decoding and produce hard decisions for a data sink, the methods do not have an explicit relationship to decoder BER as a function of decoder output LLR values, which is the desired formulation for the BERLL.

In [15] the frame error rate (FER) for Turbo codes is estimated via table look-up for a specific decoder architecture using the average number of decoder frames as an input variable. The *BER-estim* technique in [16] moves toward the present objective, and gives

a method to determine a Turbo decoder stopping rule using output LLRs. The arithmetic in [16] bears strong resemblance to that in [18] (and is in fact equivalent, which will be proved), where in the latter the LLR of codeword bits is used to speed up BER simulations, but not specifically as a decoder stopping criteria. (The relationship between LLR values and BER estimates in [18] was stated previously in Definition 3.) In the next subsection the ideas of [16] are complemented and extended using the slightly more-simplified formulation of BER as a function of decoder LLR values proposed in [18]. The result is not simply a decoder stopping criteria alternative; after incorporating a target BER specification and a simple log-ratio transformation, a BER comparator signal emerges, possessing its own sign and magnitude indicating both direction and extent of decoder BER and target BER difference. It is important at this stage in the discussion to note that the conventional DSC methods, while intended to reduce decoder iterations, do so with the intent of having the decoder continue to provide best-effort codeword estimate accuracy (minimum BER). The notion of further reducing decoder iterations at the *expense* of system BER to better optimize other system resources is not a stated objective in DSC literature.

3.3.2 BER Estimation from Stopping Criteria

Definition 3 can be used to convert a BER value for a codeword bit using the decoder LLR value for that bit. It is of interest to establish equivalence between an intermediate decoder stopping criteria expression found in [16] and Definition 3 as a starting point toward the desired BER error signal needed for loop operation. Let u_k be the kth transmitted bit in a codeword and $\hat{u}_k$ be the corresponding ECC decoder estimate of u_k. In equation (6)

of [16], an estimate $\hat{P}_c$ for the probability that $\hat{u}_k$ is correct is formulated from the basic log-likelihood ratio definition:

$$\hat{P}_c \triangleq P(\hat{u}_k = u_k) = \frac{e^{\text{sign}(\hat{u}_k)L(\hat{u}_k)}}{1 + e^{L(\hat{u}_k)}}\,, \tag{3.1}$$

where $L(\hat{u}_k)$ is the decoder output LLR of $\hat{u}_k$ and sign$(\hat{u}_k)$ assumes values from $\{0, 1\}$. Using this formulation, it follows that an estimate for the probability that $\hat{u}_k$ is in error, $\hat{P}_e$, is

$$\hat{P}_e = 1 - \hat{P}_c = \frac{1 + e^{L(\hat{u}_k)} - e^{\text{sign}(\hat{u}_k)L(\hat{u}_k)}}{1 + e^{L(\hat{u}_k)}}. \tag{3.2}$$

For $L(\hat{u}_k) \geq 0$, (3.2) can be expressed using the absolute value of $L(\hat{u}_k)$ as

$$\hat{P}_e|_{L(\hat{u}_k)\geq 0} = \frac{1 + e^{|L(\hat{u}_k)|} - e^{|L(\hat{u}_k)|}}{1 + e^{|L(\hat{u}_k)|}} = \frac{1}{1 + e^{|L(\hat{u}_k)|}}\,, \tag{3.3}$$

while for $L(\hat{u}_k) < 0$,

$$\hat{P}_e|_{L(\hat{u}_k)<0} = \frac{1 + e^{-|L(\hat{u}_k)|} - 1}{1 + e^{-|L(\hat{u}_k)|}} = \frac{e^{-|L(\hat{u}_k)|}}{1 + e^{-|L(\hat{u}_k)|}}. \tag{3.4}$$

Thus for both cases,

$$\hat{P}_e = \frac{1}{1 + e^{|L(\hat{u}_k)|}}\,, \tag{3.5}$$

which matches the result in [18] and Definition 3, and reconfirms that the probability of error for $\hat{u}_k$ can be estimated simply from the magnitude of the decoder output LLR of $\hat{u}_k$, without needing to consider its sign. From the formulation in (3.5), a BER comparator can be constructed and is established next.

3.3.3 BER Error Comparator

In [16] the natural logarithm of (3.1) is compared with $\log(1 - P_{\text{ML}})$, where P_{ML} is the maximum likelihood block error rate. The block error rate is chosen based on the conclusion that this value will exceed the bit error rate, P_e, and thus $1 - P_{\text{ML}}$ will be smaller than $1 - P_e$. The objective is to define a more conservative stopping criteria of

$$\log \hat{P}_c > \log (1 - P_{\text{ML}}). \tag{3.6}$$

Although not explicitly pointed out in [16], (3.6) can be viewed as an inverted BER *comparator* of sorts, with the inversion applied since the comparison is made between probability of correctness and not probability of error. A similar approach is proposed in this dissertation, but instead the actual estimated BER using (3.5) is compared with a target BER.

Consider a communication system with a BER specification of P_0 that uses an iterative error correction decoder. If $P_0/\hat{P}_e > 1$, then within the extent that $\hat{P}_e$ is an accurate estimate of the true decoder output BER, the decoder is producing codeword estimates that are *more* accurate than necessary. As a result, it may be possible to use fewer decoder iterations while still meeting the BER specification P_0. By similar reasoning, if instead $P_0/\hat{P}_e < 1$, the decoder is producing codeword estimates that are *less* accurate than required, and additional decoder iterations are needed to meet the BER specification P_0. As an ECC decoder is swept across a range of input energy-to-noise ratios, E_b/N_0, the decoder output BER will likely swing through several successive powers of 10. Subsequently, the ratio $P_0/\hat{P}_e$ for a fixed P_0 will vary from small fractions to potentially several powers of 10, presenting a

wide dynamic range on the ratio as a function of E_b/N_0. To narrow the dynamic range of an exponentially-varying quantity, the logarithm is useful. With these ideas in mind, a BER error comparator is proposed and takes the form

$$\varepsilon_1 \triangleq \log\left(\alpha \frac{P_0}{\overline{P}_e}\right), \tag{3.7}$$

where $\overline{P}_e$ is an average BER estimate of $\hat{P}_e$ values taken over the message bits of one or more codewords using the LLR-based function in (3.5), and α is a parameter introduced to compensate for proportional-based inaccuracies in $\overline{P}_e$. While it would be preferable to formulate a multiple, concurrent bit version of (3.5) and use this instead of $\overline{P}_e$, simulation results presented in the next section show that $\overline{P}_e$ tracks the Monte Carlo BER average reasonably well, where the Monte Carlo average is calculated as the number of errored bits divided by the total bit count.

In addition to compressing the dynamic range of the ratio $P_0/\overline{P}_e$, (3.7) also possesses four valuable characteristics:

1. Linearization of the exponent variation between P_0 and $\overline{P}_e$.
2. A sign indicating whether $\overline{P}_e$ is too small or too large relative to a specified P_0.
3. A zero-crossing point when $\alpha P_0 = \overline{P}_e$ (ideally unique).
4. Monotonicity as a function of decoder input E_b/N_0 (so long as decoder BER is also monotonic with E_b/N_0).

These properties suggest that (3.7) is well-suited as a BER *error signal*. Properties 2–4 can be envisioned as the channel E_b/N_0 ratio acting on an ECC decoder is swept across a

range of values. For a static decoder configuration the BER error signal will swing from negative values for low E_b/N_0 where $\overline{P}_e > \alpha P_0$, pass through zero as E_b/N_0 is increased to the point where $\overline{P}_e = \alpha P_0$, and finally become positive with further increases in E_b/N_0 where $\overline{P}_e < \alpha P_0$.

The target BER P_0 has an associated target LLR magnitude, $|L_0|$, which is expressed as a function of P_0 by applying the log-likelihood ratio definition:

$$|L_0| = \left| \log\left(\frac{1-P_0}{P_0}\right) \right| , \tag{3.8}$$

which is simply an application of Definition 2. The target LLR value $|L_0|$ can be regarded as the LLR magnitude that an ideal decoder would produce for a codeword bit estimate having bit error probability P_0. Due to decoder nonidealities, such as finite code length and statistical dependencies between constituent parallel decoders (Turbo codes) or finite-girth Tanner graphs (LDPC codes), the actual decoder LLR output magnitude $|L(\hat{u}_k)|$ will not equal the target LLR $|L_0|$ when the decoder is producing a BER of P_0 on output bit estimates. Noting this disparity, a second candidate error expression is proposed,

$$\varepsilon_2 \triangleq \beta\overline{|L|} - |L_0| , \tag{3.9}$$

where $\overline{|L|} = \overline{|L(\hat{u}_k)|}$ is an average LLR magnitude taken over the message bits of one or more codewords, and the parameter β is introduced to compensate for decoder nonidealities. The computational simplicity of the second candidate error term in (3.9) is noted, where all that is required to form the error signal is a simple subtraction operation between

the scaled average decoder output LLR magnitude and the target LLR. The second error signal candidate ε_2 will be shown to also possess the four properties desired for general error signals listed previously.

Noting the form of the second error signal candidate in (3.9) as a difference of two LLR values, it is worthwhile to show that this is numerically related to taking the difference of two probability of error exponents, proven as follows.

Theorem 1. *The difference of two log-likelihood ratios is directly proportional to the difference of exponents of their respective probabilities of error, for small probabilities.*

Proof. Assume two probabilities of error (or BERs) $P_1, P_2 \ll 1/2$, where $P_1 = 10^{-a_1}$ and $P_2 = 10^{-a_2}$. Define $L_1 \triangleq \log\left(\frac{1-P_1}{P_1}\right)$ and $L_2 \triangleq \log\left(\frac{1-P_2}{P_2}\right)$, then

$$\begin{aligned}
L_1 - L_2 &= \log\left(\frac{1-P_1}{P_1}\right) - \log\left(\frac{1-P_2}{P_2}\right) \\
&= \log\left(1/P_1\right) - \log\left(1/P_2\right) + \log\left(\frac{1-P_1}{1-P_2}\right) && (3.10) \\
&\approx \log\left(1/P_1\right) - \log\left(1/P_2\right) \qquad \text{since } P_1, P_2 \ll 1/2 \\
&= \log\left(P_2/P_1\right) && (3.11) \\
&= \log\left(\frac{10^{-a_2}}{10^{-a_1}}\right) \\
&= a_1 \log(10) - a_2 \log(10) \\
&= k\,(a_1 - a_2)\,,
\end{aligned}$$

where $k = \log(10) = \ln(10) \approx 2.3$. □

In the preceding proof, the constraint $P_1, P_2 \ll 1/2$ ensures that L_1 and L_2 are strictly positive. The logarithm of the ratio of probabilities expression in (3.11) bears strong resem-

blance to the first error signal candidate expression (3.7), and demonstrates that ε_1 and ε_2 are fundamentally similar in that both signals approximate the difference of exponents for two probabilities of error. Figure 3.6 plots the residual error term (the third term) in (3.10), normalized by the approximation result, as a function of probability of error exponent difference $a_2 - a_1$. In Figure 3.6, a_1 is fixed at -3 which gives $P_1 = 10^{-3}$, while a_2 is varied from -5 to -1, so that P_2 takes on values from 10^{-5} to 10^{-1}. The graph verifies that the direct variation approximation proposed in Theorem 1 is quite accurate throughout the range of probabilities, and grows rapidly as the variable error probability approaches 1/2.

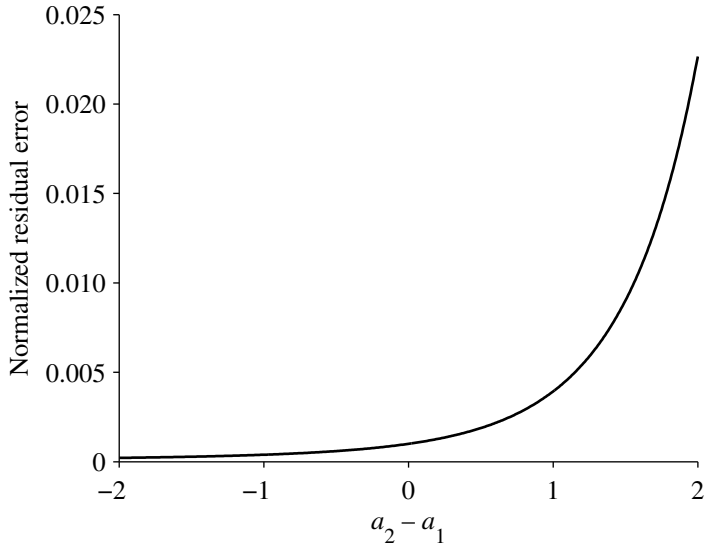


Figure 3.6: Normalized residual error in the LLR difference approximation of Theorem 1 as a function of the difference in exponents of probabilities.

For a given decoder architecture, suitable values for α and β are needed to ensure that (3.7) and (3.9) possess the desired characteristics associated with an error signal. In general, as the number of decoder iterations changes, so too will the optimal values of α and β that result in exact zero-crossings of ε_1 and ε_2 when $\overline{P}_e = \alpha P_0$ or $\beta\overline{|L|} = |L_0|$, respectively. Thus when maximum accuracy is more important than implementation effi-

ciency, a look-up table (LUT) of α and β parameters as functions of iteration number can be stored, with specific values inserted into (3.7) and (3.9) to evaluate the BER error signal for the present decoder configuration. The LUT approach is the typical method used for the class of decoder stopping criterion in [15] and [16]. The specific values of α and β to populate the tables with can be determined via Monte Carlo simulations, where for each iteration value the E_b/N_0 level is adjusted to the point where P_e, the Monte Carlo BER estimate, equals P_0. At this E_b/N_0 value, ε_1 and ε_2 are set to zero which, along with the Monte Carlo estimates for $\overline{P}_e$ and $\overline{|L|}$, allows α and β to be solved via (3.7) and (3.9), respectively. Simulation results demonstrating the table look-up method and verifying the suitability of ε_1 and ε_2 as reliable BER error signals are examined in the next section.

3.3.4 BER Error Comparator Simulation Results

Figure 3.7 shows plots of (a) P_e and (b) $\overline{|L(\hat{u}_k)|}$ versus E_b/N_0 for the rate 1/2 (576, 288) IEEE 802.16 WiMAX LDPC code for 1, 5 and 25 decoder iterations, simulated over an additive white Gaussian noise (AWGN) channel using BPSK modulation. To demonstrate the method of determining LUT values for α and β, a target $P_0 = 10^{-4}$ was applied to the family of P_e versus E_b/N_0 waterfall curves in Figure 3.7, which can be visualized as a horizontal line at $P_e = 10^{-4}$ intersecting each waterfall curve, where each intersection defines the zero-crossing point for a particular decoder iteration value. Figure 3.8 shows plots of the resulting per-iteration α and β parameters that produce ideal zero-crossing comparator error signals for ε_1 and ε_2, respectively.

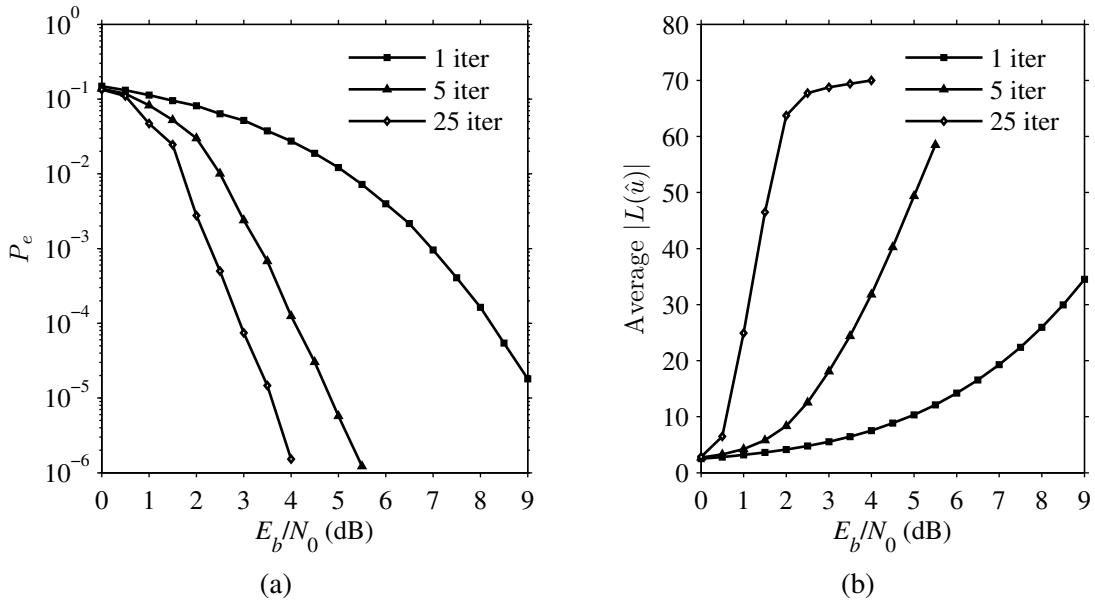


Figure 3.7: Graph of (a) probability of error and (b) average log-likelihood magnitude versus energy-to-noise ratio for the rate 1/2 (576, 288) IEEE 802.16 WiMAX LDPC code for 1, 5 and 25 iterations, using BPSK modulation over an AWGN channel.

The parameters α and β in Figure 3.8 show a decaying exponential characteristic as a function of decoder iteration, a result of the decaying improvement of either $\hat{P}_e$ or LLR as functions of the same. Thus if α- or β-table pruning is required, the first pruning step would be to collapse the table above a specific iteration number—for example 15 iterations based upon the simulation data—and replace the table entries beyond this iteration number with a single value, for example $\alpha \approx 0.4$ and $\beta \approx 0.13$. A more aggressive simplification is applied in the linear analysis and simulation chapters that follow, where a single value is used for β across the entire range of decoder iterations. This single value simplification is studied and compared in detail with both the CE and HDA methods in [17].

Graphs of BER error comparator signals (a) ε_1 and (b) ε_2 as functions of the channel E_b/N_0 ratio are shown in Figure 3.9 for a decoder configured to perform 5 iterations and meet a target BER of $P_0 = 10^{-4}$. The zero-crossings of both error signals correspond to

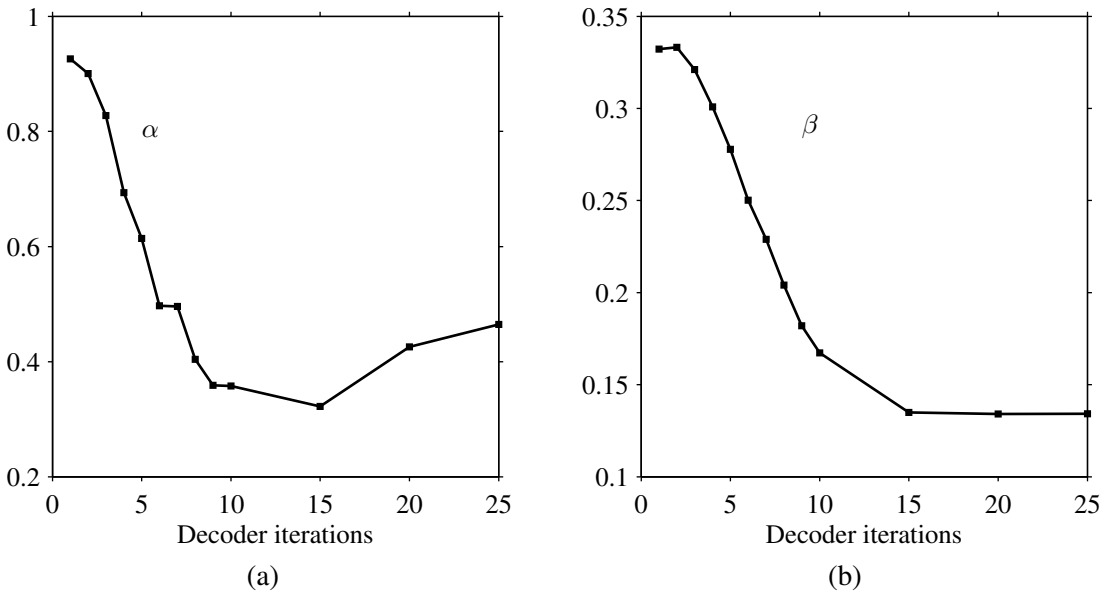


Figure 3.8: Parameters (a) α and (b) β that give ideal zero-crossings for the error comparators associated with the performance plots in Figure 3.7, as a function of decoder iteration number.

the point on the 5-iteration subcode waterfall performance curve in Figure 3.7(a) where the Monte Carlo BER equals 10^{-4}; the ideal crossing point giving zero error is shown as a crosshair on each plot. Comparing the two comparator graphs, both possess the desired characteristics listed in the previous section (specifically sign, magnitude and monotonicity), and thus the signals can be utilized as reliable BER error signals. The LLR-based error signal ε_2 is seen to have better trending and range about the zero-crossing point than ε_1, especially for negative values. For digital implementations, a vector size of 6 bits will easily fit the extent of error signal variation for the whole number portion of both ε_1 and ε_2 over the range of E_b/N_0 values shown, with a possible upper and lower clipping applied to prevent word overflow. It is an intuitive exercise to picture either error signal being fed into a sign-inverter, the output scaled and filtered, and then finally input into the decoder as an auxiliary control to tune the decoder in real-time to achieve a target BER.

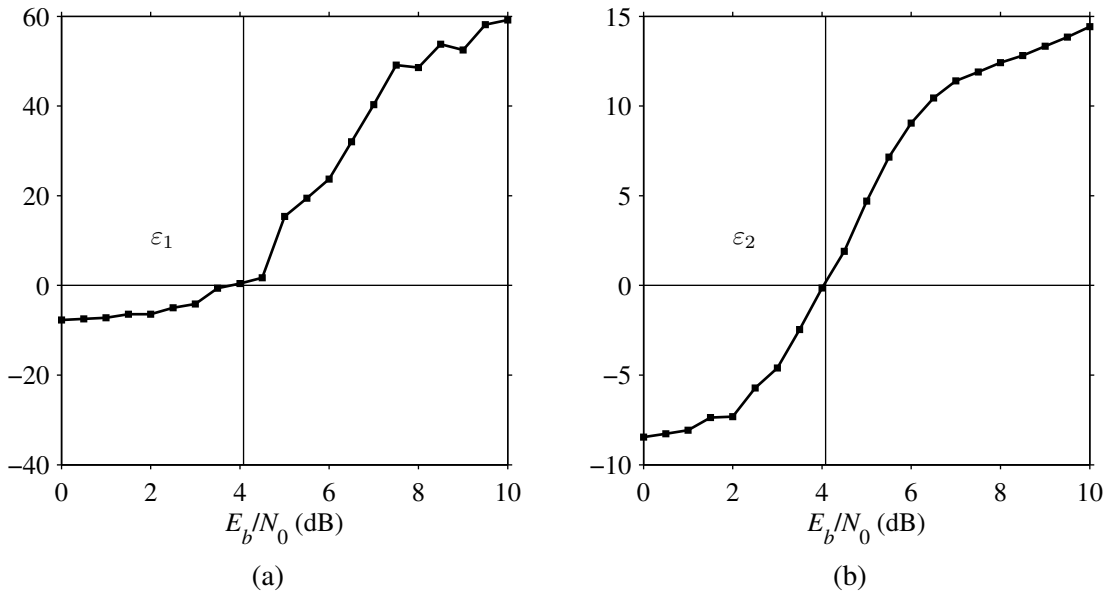


Figure 3.9: BER error signals (a) ε_1 and (b) ε_2 as functions of the channel E_b/N_0 ratio for 5 decoder iterations and a target BER of 10^{-4}.

3.4 Log-Likelihood Ratio Decoder Transfer Curve

Coupled to the input of the BER error detector in the previous section is the decoder itself. To fully characterize the closed loop circuit, knowledge of the input-to-output relationship of the decoder is required. From a conventional data path point of view, the input to the decoder is simply the received codeword from the channel, while the output is the error-corrected codeword sent to the data sink, shown previously in Figure 3.1. However from a loop point of view, the decoder input is one of the parameters available for modifying decoder error correction strength, while the decoder output of interest is an LLR value. This input parameter to output LLR relationship, in tandem with the error detector definition in (3.9), is the bulk replacement of the phase error detector in the standard PLL, as it relates to the BERLL.

The shape of the LLR output is primarily a function of the specific parameter being varied, and the E_b/N_0 level input to the decoder from the channel; in this dissertation the curve is termed the decoder LLR *transfer curve*. A unique LLR transfer curve will exist for each parameter available for modifying decoder performance. For a specific parameter, varying the E_b/N_0 level will generate a family of LLR transfer curves.

This dissertation examines two separate decoder configuration parameters, varied independently to actuate the decoder. While the specific LLR transfer curves and their impact on closed loop operation are examined in subsequent chapters, their general characteristics are shown in Figure 3.10 as a function of a generic decoder parameter s_i. Several features of the LLR transfer curve are worth noting. The first feature is *monotonicity*. For proper closed loop operation, it is important that the decoder LLR output remain monotonic as a function of parameter variation. This will allow a zero-crossing to exist on a plot of detector error (ε_2) versus s_i, which the loop will need to find a steady state operating point. (It is noted that the decoder transfer curve monotonicity is related to, but distinct from ε_1 and ε_2 BER error signal monotonicity versus channel E_b/N_0.)

The second feature illustrated in Figure 3.10 is that decoder output LLR increases more rapidly for larger values of channel E_b/N_0. Within this feature, a primarily linear relationship between output LLR and parameter variation is seen for lower E_b/N_0; this will be used to model the linear equivalent circuit for the BERLL in subsequent chapters, where the ECC decoder is replaced with a constant gain term, k_D. Unfortunately for large E_b/N_0 values, the LLR versus parameter curve deviates from the linear approximation; in this case a multi-segment, piecewise linear approximation can be used to model the transfer curve.

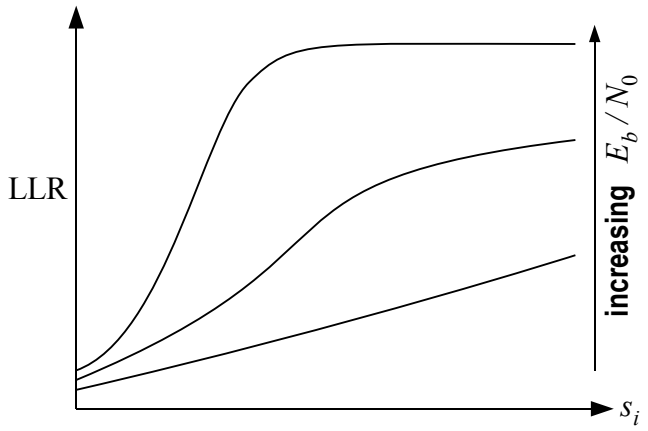


Figure 3.10: Log-likelihood ratio decoder transfer curve illustration.

A third characteristic of decoder LLR output as a function of increasing E_b/N_0 and parameter value is a saturation effect, where the LLR curve levels-off with further increases to parameter value. LLR saturation will lead to an effect similar to signal clipping when the parameter setting has reached the limit of its impact on decoder error correction. Finally, it is noted that depending upon the specific parameter used to adjust the decoder, the LLR transfer curve can be either increasing (as shown in Figure 3.10) or decreasing as a function of the parameter.

3.5 Low-Pass Loop Filter

While the error detectors for a PLL and a BERLL are noticeably distinct, both loops will have very similar loop filter structures, typically of a low-pass nature. Since ECC technology is usually employed in discrete time systems, a discrete time, or digital implementation of the loop filter will be assumed and used for closed loop time and frequency domain analyses. The typical order of the low-pass filter (LPF) is one, but higher orders

are occasionally used for a PLL. Due to the presence of the integrator (accumulator in the discrete version), the order of the loop will always be one plus the LPF order.

Digital filters come in finite impulse response (FIR, non-feedback) and infinite impulse response (IIR, feedback) topologies. A properly designed IIR LPF will typically require fewer delay taps than an FIR LPF to achieve a given attenuation level versus input frequency; for this reason a first-order, single tap IIR filter will be used for the loop filter in the BERLL, shown in Figure 3.11. Two IIR filter coefficients are provided in the design, and will be tuned in tandem with the other gains in the loop to meet a particular time-domain response and/or frequency domain filtering requirement. When this single-tap IIR filter is used as a stand-alone filter, it exhibits an excellent trade-off between attenuation of unwanted signals and hardware resources required to implement the filter, and is often used as a simple but effective noise attenuator [27].

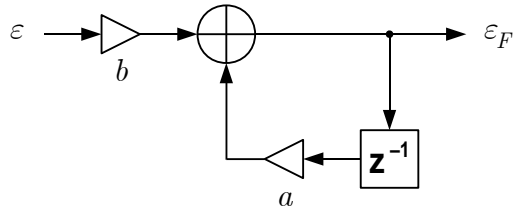


Figure 3.11: Digital infinite impulse response low-pass loop filter used for BERLL error signal filtering.

The magnitude and phase responses of the single tap IIR are shown in Figure 3.12 for (a, b) coefficient values of (0.5, 0.5) and (0.95, 0.05), where for both coefficient pairs the sum of coefficients is one, which is necessary for unity gain at $\omega = 0$ rad/s. The trade-off between the two coefficient pairs is evident from the response plots; the pair having larger a coefficient is seen to have good high frequency attenuation but poor phase response at the lower end of the Nyquist interval. Conversely, the coefficient pair having smaller a

coefficient exhibits poor high frequency attenuation but the best phase response across the Nyquist band. The phase responses of both coefficient pairs confirm traditional stability for all frequencies since the phase remains above -180° throughout the band.

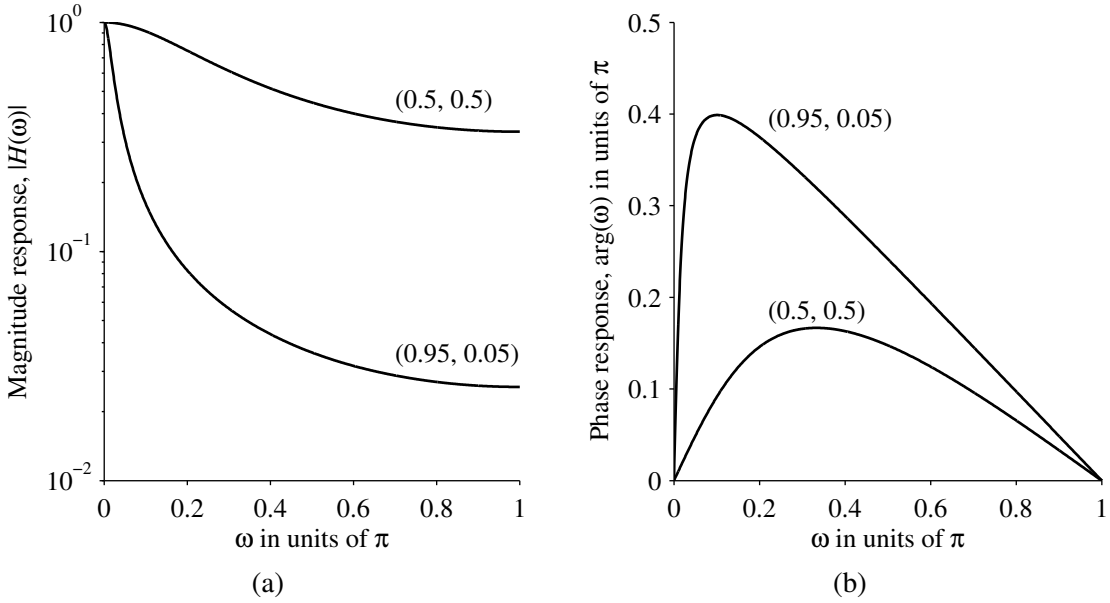


Figure 3.12: Graphs of (a) magnitude and (b) phase responses for the low-pass infinite impulse response loop filter proposed for use with the BERLL for filter (a, b) coefficient pairs of (0.5, 0.5) and (0.95, 0.05).

3.6 Accumulator and Translator

Both the PLL and BERLL require an integration function to cumulatively hold the filtered error signal variations that keep the input and local phases equal in the PLL, or the target LLR and scaled decoder output LLR equal in the BERLL. For analog PLL circuits, the integrator is commonly realized using a voltage-controlled oscillator (VCO) circuit. For digital PLL circuits and the BERLL, the integrator is realized using a simple accumulator, shown in block diagram form in Figure 3.13. Note that an accumulator is a specific instance

of the more general IIR filter in Figure 3.11, with $a = b = 1$, and is marginally stable from a discrete time stability viewpoint.

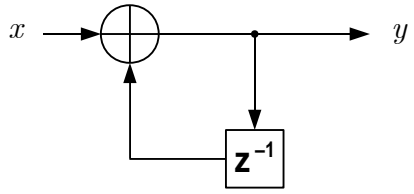


Figure 3.13: Digital accumulator equivalent of an analog integrator.

The final signal processing required in a BERLL, and in certain PLLs, is a scale factor that translates an accumulated error value into units appropriate for the error detector in the case of the PLL, or the ECC decoder in the case of the BERLL. For the BERLL, the value of the translation scale factor, also referred to as the post-accumulator gain, will change depending upon which decoder parameter is being used to tune its error correction strength. The scale factor is embodied in the fixed gain block, k_A, in Figure 3.3.

Chapter 4

Discrete Time Linear

Closed Loop Analysis

Having examined the individual components of the bit error rate locked loop in detail in the preceding chapter, this chapter derives the discrete time, linear closed loop expressions for the circuit. The analysis takes advantage of the constant gain approximation for the ECC decoder relating decoder output LLR as a function of decoder input parameter, which is valid for low-to-moderate channel E_b/N_0 levels. Both discrete time domain and frequency domain analyses are conducted. The analytical results will be valuable when predicting actual circuit operation with specific decoder feedback configurations and for prescribing specific loop performance, taken up in subsequent chapters.

Referring back to the archetypal PLL diagram in Figure 3.2, the primary transfer function of interest relates input phase $\theta_0(z) = \mathcal{Z}\left[\theta_0(n)\right]$ to output phase $\theta(z) = \mathcal{Z}\left[\theta(n)\right]$, where n is the discrete time index and $\mathcal{Z}$ denotes the z-transform of the discrete time signal:

$$\theta(z) = \mathcal{Z}\left[\theta(n)\right] \triangleq \sum_{n=-\infty}^{\infty} \theta(n) z^{-n}. \tag{4.1}$$

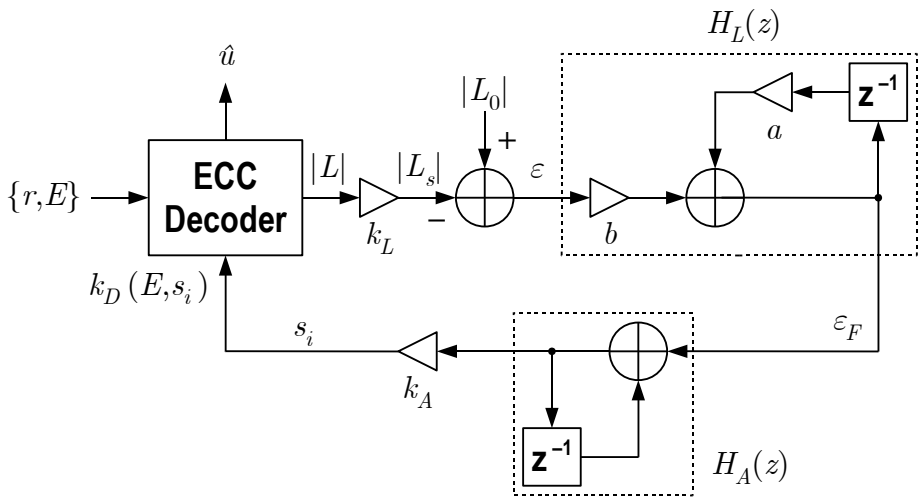


Figure 4.1: Fully annotated bit error rate locked loop circuit.

The transfer function of the PLL relating input phase to output phase is then defined as

$$H(z) = \frac{\theta(z)}{\theta_0(z)} \qquad \text{(PLL transfer function)}\,, \tag{4.2}$$

which is also referred to as the *system* transfer function in PLL texts [24].

Examining the BERLL diagram in Figure 3.3, redrawn in Figure 4.1 with all modules expanded and annotated with signal variables, the analogous transfer function of interest to the PLL phase signals relates input LLR magnitude $|L_0|$ to scaled decoder output LLR magnitude $|L_s|$. Letting $L_0(n)$ and $L_s(n)$ represent the discrete time LLR magnitude signals $|L_0|$ and $|L_s|$, where the absolute value operation is suppressed for convenience, the BERLL transfer function is defined as

$$H(z) \triangleq \frac{L_s(z)}{L_0(z)}, \qquad \text{(BERLL transfer function)} \tag{4.3}$$

where $L_s(z)$ and $L_0(z)$ are the z-transforms of $L_s(n)$ and $L_0(n)$, respectively. It is noted here that in Figure 4.1, r is the noise-corrupted received codeword, $\hat{u}$ is the decoder message estimate, and E is shorthand for the channel E_b/N_0 ratio; the decoder is modeled as a gain term k_D dependent upon the specific decoder configuration parameter s_i and the E_b/N_0 ratio E, and is regarded as a constant for a fixed s_i and for moderate E_b/N_0 levels. Although the closed loop analysis is not dependent upon r or $\hat{u}$, these signals are included in Figure 4.1 to indicate their positions in the system from a data path input/output view.

The analysis will proceed as follows: First the transfer function defined in (4.3) will be found as a function of the loop gains k_A, k_D, k_L and loop filter coefficients a and b. With the transfer function in hand, the frequency and magnitude responses of the closed loop is derived. Next, pole-zero analysis is conducted using the denominator of the transfer function, and is examined in light of varying the loop filter coefficient a across the continuum of values in $(0, 1)$. Approximate expressions for natural frequency, natural resonant frequency, and damping ratio using both linear and bilinear approximations to the complex digital frequency variable, z, are developed next. Finally, the transfer function is used to determine the discrete, closed loop impulse and step responses for underdamped, critically-damped and overdamped cases. The step responses are particularly important when determining time-domain characteristics such as rise time, and are also of interest after system start-up and when a sudden abrupt rise or drop in the channel E_b/N_0 ratio occurs, such as with a signal fading event.

4.1 Transfer Function

The analysis begins by expressing the error signal z-transform, $\varepsilon(z)$, as a function of $L_0(z)$ and $L_s(z)$,

$$\varepsilon(z) = L_0(z) - L_s(z). \tag{4.4}$$

It is noted that in (4.4), negative feedback is implicitly accounted for by subtracting the scaled decoder output LLR from the target LLR. The output of the loop filter in the z-domain is

$$\varepsilon_F(z) = H_L(z)\,\varepsilon(z)\,, \tag{4.5}$$

while the z-transform of the decoder parameter input is expressed in terms of the loop filter output as

$$s_i(z) = k_A\,H_A(z)\,\varepsilon_F(z), \tag{4.6}$$

where the subscript i on $s_i(z)$ is used as an index into a list of available decoder parameters (number of decoder iterations, number of nulled parity bits, etc.). In general the signal $s_i(n)$, the inverse z-transform of $s_i(z)$, is a discrete-valued signal in both time and amplitude (typically an integer), but is approximated for the purpose of loop analysis as having continuous-valued amplitude; this assumption is validated when the set of integers from which $s_i(n)$ can be selected contains a large number of contiguous values.

Using the linear model for the decoder LLR output as a function of the input parameter s_i, the z-transform $L_s(z)$ is written as

$$L_s(z) = k_D\,k_L\,s_i(z). \tag{4.7}$$

Substituting (4.5) into (4.6), and the intermediate result into (4.7) gives $L_s(z)$ as a function of the unfiltered error signal $\varepsilon(z)$:

$$L_s(z) = k_A\, k_D\, k_L\, H_A(z)\, H_L(z)\, \varepsilon(z). \tag{4.8}$$

A final substitution of (4.8) into (4.4) where $\varepsilon(z)$ is expressed as a function of $L_s(z)$ gives

$$L_s(z) = k_A\, k_D\, k_L\, H_A(z)\, H_L(z)\, [L_0(z) - L_s(z)]\,. \tag{4.9}$$

The BERLL transfer function expressed in terms of the loop gains is thus

$$H(z) \triangleq \frac{L_s(z)}{L_0(z)} = \frac{k_A\, k_D\, k_L\, H_A(z)\, H_L(z)}{1 + k_A\, k_D\, k_L\, H_A(z)\, H_L(z)}. \tag{4.10}$$

Defining the constant $c = k_A k_D k_L$, and substituting the z-transforms for the IIR loop filter and accumulator into (4.10),

$$H(z) = \frac{bc}{bc + (1 - z^{-1})\,(1 - az^{-1})} = \frac{\dfrac{bc}{bc+1}}{1 - \dfrac{a+1}{bc+1}\, z^{-1} + \dfrac{a}{bc+1}\, z^{-2}}. \tag{4.11}$$

The denominator $D(z)$ of (4.11) defines the characteristic polynomial of the loop and determines the location of the poles for the closed loop response:

$$D(z) = z^2 - \frac{a+1}{bc+1}\, z + \frac{a}{bc+1}. \tag{4.12}$$

The corresponding roots of the characteristic polynomial are

$$p_{1,2} = \frac{a+1}{2\left(bc+1\right)} \pm \frac{\sqrt{\left(1-a\right)^2 - 4abc}}{2\left(bc+1\right)}, \tag{4.13}$$

where p_1 corresponds to the root having positive radical and p_2 to the root having negative radical. The location of the poles in the complex plane as a function of the loop gains and the corresponding impact on the transient response is explored in the sequel.

4.2 Frequency and Magnitude Responses

The frequency response of the closed loop is determined by substituting $z = e^{j\omega}$ in (4.11):

$$H(\omega) \triangleq H(z = e^{j\omega}) = \frac{bc}{bc + \left(1 - e^{-j\omega}\right)\left(1 - ae^{-j\omega}\right)} \tag{4.14}$$

which after expanding the complex exponential terms becomes

$$H(\omega) = \frac{bc}{bc + 1 - (a+1)\cos\omega + a\cos 2\omega + j\left[(a+1)\sin\omega - a\sin 2\omega\right]}. \tag{4.15}$$

The squared magnitude response of the loop follows directly from (4.15):

$$\left|H(\omega)\right|^2 = \frac{\left(bc\right)^2}{a^2 + \left(a+1\right)^2 + \left(bc+1\right)^2 - 2\left(a+1\right)\left(a+bc+1\right)\cos\omega + 2a\left(bc+1\right)\cos 2\omega}. \tag{4.16}$$

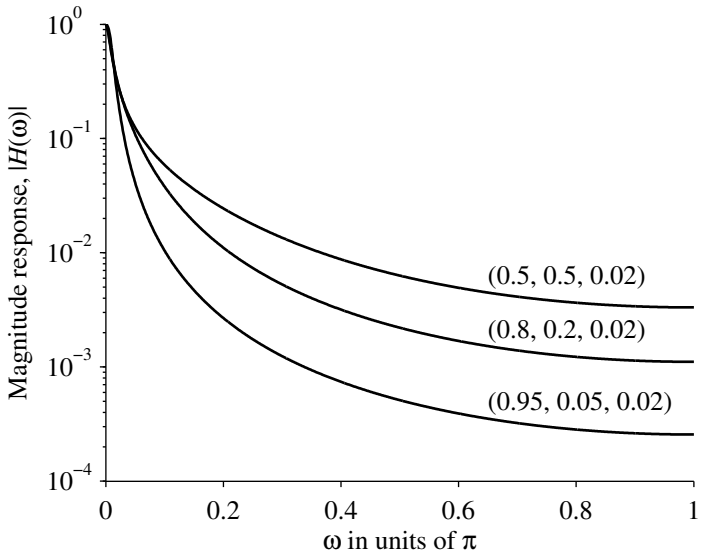


Figure 4.2: Bit error rate locked loop magnitude response.

Substituting $\omega = 0$ rad/sec into (4.16) or (4.14), the magnitude response can be shown to have unity gain for all a, b and c; it is noted that the closed loop unity gain result is independent from the loop filter DC unity gain condition $a + b = 1$.

The magnitude response of the closed BERLL is shown in Figure 4.2 for (a, b, c) ordered triples of $(0.5, 0.5, 0.02)$ (lowest-attenuation), $(0.8, 0.2, 0.02)$ (mid-attenuation) and $(0.95, 0.05, 0.02)$ (highest-attenuation). As with the PLL system transfer function, the BERLL attenuates high frequency terms and passes low-frequency signal variation. This provides a steady, consistent bit error rate tracking by the decoder and loop as the average channel conditions change over time. It is noted that the BERLL magnitude response closely resembles the basic notch filter at DC of the loop filter, shown previously in Figure 3.12(a).

4.3 Pole-Zero Analysis

It is of interest to study the location and behavior of the transfer function poles for the BERLL, particularly as a function of the loop gains a, b and $c = k_A k_D k_L$. The sign of the discriminant shared by the two poles in (4.13),

$$(1-a)^2 - 4abc \underset{>}{\overset{<}{\equiv}} 0 \tag{4.17}$$

determines whether the poles are complex conjugates (giving an underdamped response), real and identical (giving a critically-damped response) or real and distinct (giving an overdamped response). Adding the DC unity gain loop filter coefficient constraint of $a + b = 1$ to the discriminant cases in (4.17) results in expressions for a and b as functions of the lumped gain c that identify the three transient response types:

$$\begin{aligned}
\text{Underdamped:} \quad & a > \frac{1}{1+4c} \quad & b < \frac{4c}{1+4c} \\
\text{Critically-damped:} \quad & a = \frac{1}{1+4c} \quad & b = \frac{4c}{1+4c} \\
\text{Overdamped:} \quad & a < \frac{1}{1+4c} \quad & b > \frac{4c}{1+4c} \, .
\end{aligned} \tag{4.18}$$

Giving expressions for a and b in terms of c is practical from the standpoint that for a given E_b/N_0 ratio, decoder architecture, and feedback parameter type, c is essentially predetermined. In this context a and b are adjusted accordingly to achieve a particular loop performance objective, typically in terms of noise attenuation figure or time domain responsiveness.

The pole-zero plot for the BERLL is shown in Figure 4.3; the two zeros at the origin are represented with a single white $\times$ on top of the pole at the origin. As depicted in the figure, the two poles lie exclusively in the right-half of the z-plane for all $(a, b, c) \in (0, 1)$, where $a + b = 1$. For a loop filter coefficient pair $(a, b) = (0^+, 1^-)$ and a fixed value of $c \in (0, 1)$, the pole p_1 initiates at the complex coordinate $(x_{\text{od}}, 0)$ on the real-axis, where

$$x_{\text{od}} = \frac{1}{c+1}\,, \tag{4.19}$$

while the pole p_2 initiates at coordinate $(0, 0)$. Note that $(a, b) = (0^+, 1^-)$ is an impractical lower-limit for a and upper-limit for b, since the strict domains do not actually include the endpoints 0 and 1. These initial real-axis coordinates also correspond to the pole locations that produce the most overdamped response possible. Increasing the coefficient a above zero toward $1/(1 + 4c)$ moves the response toward the critically-damped type, which coincides with complex coordinate $(x_{\text{cd}}, 0)$ in Figure (4.3), where

$$x_{\text{cd}} = \frac{1}{2c+1}. \tag{4.20}$$

Further increasing a above $1/(1 + 4c)$ changes the transient response type to underdamped, at which point the p_1 root begins a circular arc-like trajectory in the first, positive-imaginary quadrant of the z-plane, while the p_2 root begins a real-axis mirror-image trajectory relative to p_1, in the fourth, negative-imaginary quadrant.

Increasing a within the sub-domain between $1/(1 + 4c)$ and 1 gives increasingly underdamped transient responses. Finally, at a coefficient pair of $(a, b) = (1^-, 0^+)$, the max-

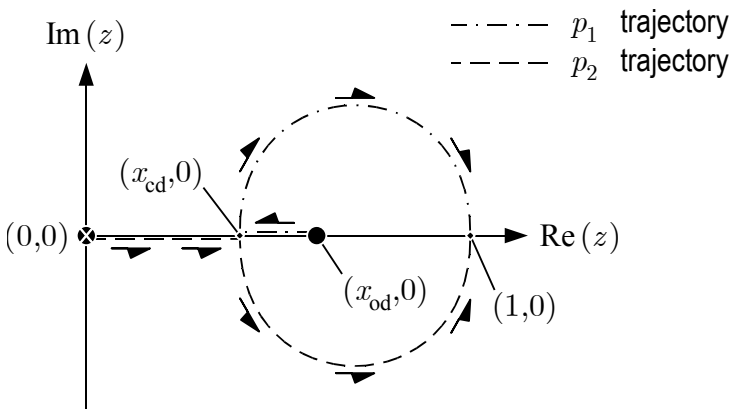


Figure 4.3: Closed loop pole trajectories in the complex plane for $a \in (0,1)$, $a+b=1$ and c constant.

imally underdamped (marginally stable) response is achieved, and the poles p_1 and p_2 re-converge, in the limit, to a value of 1:

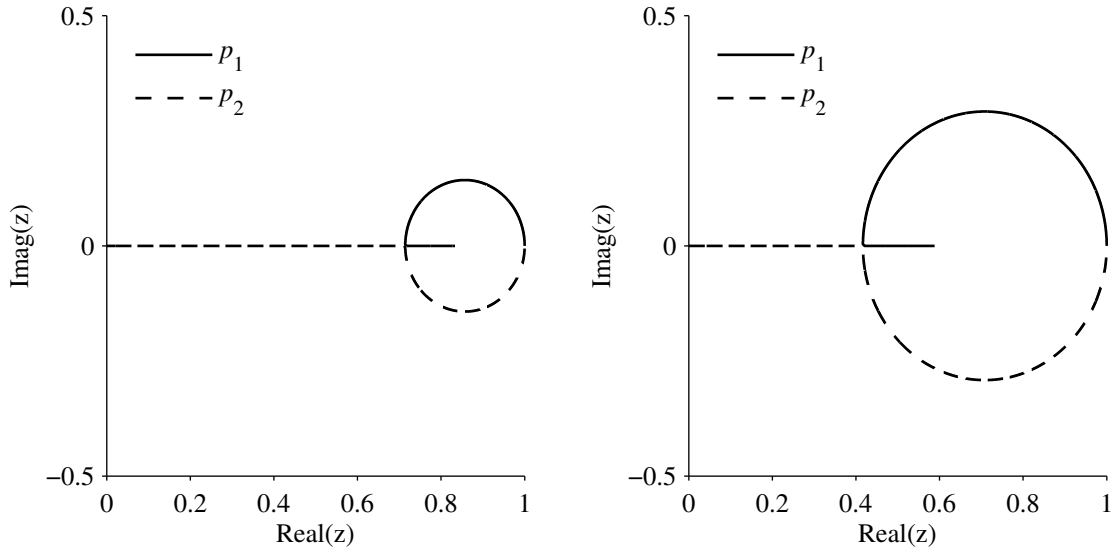
$$\lim_{a \to 1} p_{1,2} = 1. \tag{4.21}$$

Figure 4.4 shows two pole trajectory plots for specific values of the lumped parameter c. Figure 4.4(a) was generated for $c = 0.2$, while Figure 4.4(b) has $c = 0.7$. The off-axis trajectory arc area (associated with underdamped response behavior) is observed to increase as c increases, while both the initial launching point of pole p_1 on the real axis when $a = 0$ and the real axis merging point where $p_1 = p_2$ (associated with critically-damped response behavior) are confirmed to shift toward the origin as c increases.

The closed loop transfer function also has a pair of zeros at $z = 0$, which can be verified by multiplying both numerator and denominator in (4.11) by z^2,

$$H(z) = \frac{\dfrac{bc}{bc+1}\, z^2}{z^2 - \dfrac{a+1}{bc+1}\, z + \dfrac{a}{bc+1}} \tag{4.22}$$

and substituting $z = 0$ visually.



(a) $c = 0.2$ (b) $c = 0.7$

Figure 4.4: Pole trajectory comparison plots for (a) $c = 0.2$ and (b) $c = 0.7$.

4.4 Natural Frequency, Natural Resonant Frequency and Damping Ratio

Continuous-time second order system transfer functions can be expressed in terms of their natural frequency ω_n and damping ratio ζ. Additionally, the natural resonant frequency ω_d of a damped system is related to ω_n and ζ by

$$\omega_d = \omega_n \sqrt{1 - \zeta^2}. \tag{4.23}$$

A noted advantage of using ω_n, ω_d and ζ as parameters in the frequency-domain transfer function equations is the relative ease in which all three can be quickly determined from and linked to the time-domain characteristics of the impulse and step responses of the closed loop[1]. For the BERLL, it is of interest to express these three parameters in terms of the loop gains a, b and $c = k_A k_D k_L$.

The characteristic polynomial for a general continuous-time second order system is written in canonical form as

$$\lambda(s) = s^2 + 2\,\zeta\omega_n s + {\omega_n}^2\,, \tag{4.24}$$

where $s = \sigma + j\omega$ is complex frequency. To transform the discrete time transfer function in either (4.11) or (4.22) to a continuous-time form, the complex digital frequency is expressed in terms of the complex continuous frequency as $z = e^{sT}$, where T is the loop sampling time interval in seconds. Additionally, it is generally assumed that $|z| \approx 1$ which is valid for $T \ll 1$.

The bilinear transform method approximates z as

$$z = e^{sT} = \frac{e^{sT/2}}{e^{-sT/2}} \approx \frac{1 + sT/2}{1 - sT/2}. \tag{4.25}$$

Substitution of (4.25) into the first transfer function in (4.11), with the result arranged to produce a denominator polynomial in terms of s identical in form to that in (4.24), gives

[1]Some systems and control theory texts use the alternate symbol ω_0 for ω_n; in either case both symbols represent the radian frequency at which an undamped, or resistance-free (lossless) second order system will oscillate when released from rest having some pre-existing, nonzero amount of internally stored energy, with no external forces applied.

the bilinear transform approximation to the continuous-time transfer function:

$$H(z=e^{sT}) \approx \frac{\dfrac{4bc(1+sT/2)^2}{bc+2(1+a)}\dfrac{1}{T^2}}{s^2+\dfrac{4(bc+1-a)}{bc+2(1+a)}\dfrac{1}{T}\,s+\dfrac{4bc}{bc+2(1+a)}\dfrac{1}{T^2}}. \tag{4.26}$$

Comparing the denominator of (4.26) with the canonical form of (4.24), it is readily determined that

$$\omega_n \approx \frac{2}{T}\sqrt{\frac{bc}{bc+2(1+a)}} \qquad \text{(bilinear approx.)} \tag{4.27}$$

$$\zeta \approx \frac{bc+1-a}{\sqrt{bc\big[bc+2(1+a)\big]}} \qquad \text{(bilinear approx.)}\,, \tag{4.28}$$

and from 4.23,

$$\omega_d \approx \frac{2}{T}\frac{\sqrt{4abc-(1-a)^2}}{bc+2(1+a)} \qquad \text{(bilinear approx.)}. \tag{4.29}$$

An underdamped transient response will complete a damped cycle every $2\pi/\omega_d$ seconds. The approximate number of discrete time sampling instants of size T seconds/sample contained in one damped cycle is thus

$$N \approx \frac{2\pi}{\omega_d T} = \pi\frac{bc+2(1+a)}{\sqrt{4abc-(1-a)^2}} \qquad \text{(bilinear approx.)}, \tag{4.30}$$

where the result is rounded to the nearest integer.

A simpler alternative to the bilinear transform approximation is to replace z^{-1} with the first two terms of its Taylor's series as $z^{-1} \approx 1 - sT$, which will be referred to as the

linear approximation in the present discussion. The linear approximation is substituted into the first z-domain transfer function in (4.11), with the result again arranged to produce a denominator polynomial in terms of s identical in form to that in (4.24). The linear approximation to the continuous-time transfer function becomes

$$H(z = e^{sT}) \approx \frac{bc}{bc + sT(1 - a + asT)} = \frac{\dfrac{bc}{aT^2}}{s^2 + \dfrac{1-a}{aT}\, s + \dfrac{bc}{aT^2}}. \tag{4.31}$$

As with the bilinear transform method, the denominator of (4.31) is compared with (4.24), from which it is determined that

$$\omega_n \approx \frac{1}{T}\sqrt{\frac{bc}{a}} \qquad \text{(linear approx.)} \tag{4.32}$$

$$\zeta \approx \frac{1-a}{2\sqrt{abc}} \qquad \text{(linear approx.)} \tag{4.33}$$

$$\omega_d \approx \frac{1}{T}\frac{\sqrt{4abc - (1-a)^2}}{2a} \qquad \text{(linear approx.)} \tag{4.34}$$

$$N \approx \frac{4\pi a}{\sqrt{4abc - (1-a)^2}} \qquad \text{(linear approx.).} \tag{4.35}$$

The simplistic form of the expressions in (4.32)–(4.35) relative to their bilinear counterparts in (4.27)–(4.30) is apparent.

Figure 4.5 shows plots of ζ and ω_d for both the bilinear and linear approximations as functions of the loop filter integral path gain a, for a fixed value of $c = 0.05$ and an imposed loop filter constraint $a + b = 1$. The linear and bilinear approximations for ζ remain relatively equal throughout the range of a values, while the two approximations for ω_d are seen to initially diverge as a increases above 0.84 and then eventually reconverge as

a approaches 1. Additionally, Figure 4.5 shows plots of the exact expression (4.50) for ζ and (4.43) for ω_d developed in the underdamped transient analysis in the next section. The graphs of the exact expressions and bilinear approximations for ζ and ω_d are visually indistinguishable in the figure. For the value of c selected, the bilinear approximation has less than a 0.3% error for ω_d and less than a 0.04% error for ζ, while the linear approximation has a maximum error of 10% for ω_d and less than a 5% error for ζ.

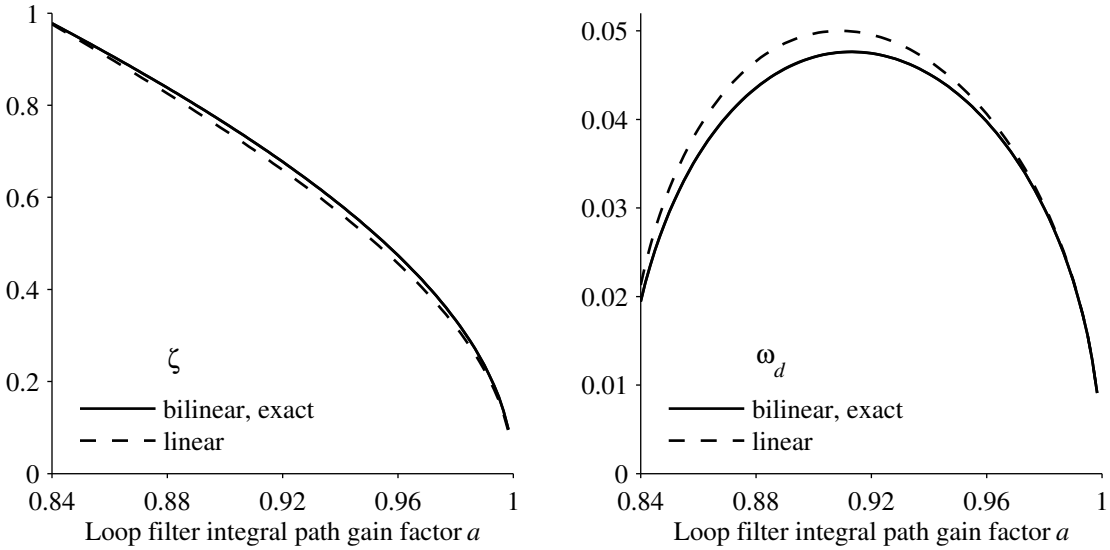


Figure 4.5: Comparison of exact expressions for (a) damping ratio ζ and (b) natural resonant frequency ω_d with bilinear and linear approximations as functions of loop gain factor a for $c = 0.05$.

4.5 Underdamped Transient Analysis

The closed loop response exhibits underdamped characteristics when

$$(1-a)^2 < 4abc\,, \tag{4.36}$$

which combined with the loop filter constraint $a + b = 1$ gives limits on the loop filter gains as listed in (4.18) of

$$a > \frac{1}{1+4c} \qquad\qquad b < \frac{4c}{1+4c}. \tag{4.37}$$

For this case, the poles p_1 and p_2 are a complex conjugate pair

$$p_{1,2} = \frac{a+1}{2(bc+1)} \pm j\frac{\sqrt{4abc-(1-a)^2}}{2(bc+1)}. \tag{4.38}$$

4.5.1 Stability

To ensure a stable, causal transient response, both poles must lie inside the complex unit circle. For p_1 and p_2 given in (4.38) this requires that

$$(a+1)^2 + 4abc - (1-a)^2 < 4(bc+1)^2 \tag{4.39}$$

for all $a, b, c \in (0, 1)$. After reducing (4.39) the stability requirement simplifies to

$$1 + bc > a\,, \tag{4.40}$$

which is satisfied for all $a, b, c \in (0, 1)$. Thus the underdamped transient response is unconditionally stable for all loop gain combinations of interest.

4.5.2 Impulse and Step Responses

For the underdamped case, the transfer function in (4.11) can be rewritten in factored denominator form as

$$H(z) = \frac{\dfrac{bc}{bc+1}}{(1 - pe^{j\omega_d} z^{-1})(1 - pe^{-j\omega_d} z^{-1})}, \tag{4.41}$$

where p and ω_d are the complex exponential magnitude and phase corresponding to p_1 and p_2, which can be reduced to

$$p \triangleq |p_{1,2}| = \sqrt{\frac{a}{bc+1}} \tag{4.42}$$

$$\omega_d \triangleq \tan^{-1}\left(\frac{\text{Im}(p_1)}{\text{Re}(p_1)}\right) = \tan^{-1}\left(\frac{\sqrt{4abc - (1-a)^2}}{a+1}\right). \tag{4.43}$$

The complex exponential angle ω_d (corresponding to the natural resonant frequency, normalized by the sampling time T) and magnitude p are shown graphically in the complex plane in Figure 4.6, which also shows the pole trajectory outlines from Figure 4.3 in the underdamped region, discussed previously.

A standard damped sinusoid z-transform pair similar to the transfer function for the underdamped case in (4.41) is

$$p^n \sin(n\omega_d)\, \text{u}(n) \quad \overset{\mathcal{Z}}{\longleftrightarrow} \quad \frac{(p \sin \omega_d)\, z^{-1}}{(1 - pe^{j\omega_d} z^{-1})(1 - pe^{-j\omega_d} z^{-1})}, \tag{4.44}$$

where $p < 1$ and ω_d is fixed, and $\text{u}(n)$ is the unit-step function. Applying the time domain shift of $n+1$ needed to compensate for the z^{-1} numerator factor in (4.44) and then nor-

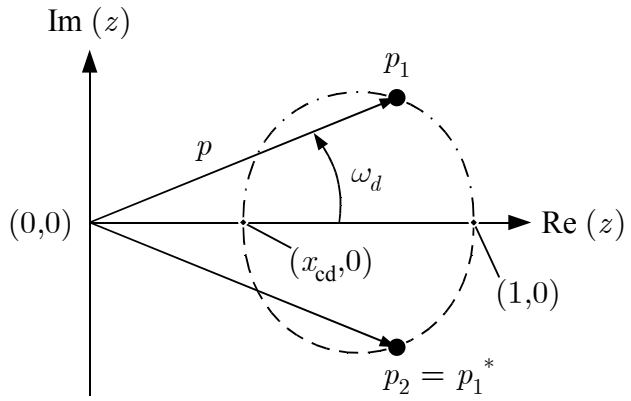


Figure 4.6: Complex exponential angle ω_d and magnitude p geometrical view in the complex plane.

malizing the remaining z-transform numerator terms to those in (4.41) gives the impulse response for the underdamped case:

$$h(n) = \frac{bc}{bc+1} \, \frac{\sin([n+1]\omega_d)}{\sin \omega_d} \, p^n \, \mathrm{u}(n). \tag{4.45}$$

Figure 4.7 shows the underdamped impulse response plotted for loop gain (a, b, c) triples of $(0.93, 0.07, 0.02)$, $(0.96, 0.04, 0.02)$ and $(0.99, 0.01, 0.02)$. The response produced by the triple having the smallest a coefficient is very close to the critically-damped response case, while the response for the triple having the largest a coefficient exhibits the most excessive overshoot (ringing) of the three selected responses. In the figure, the impulse responses are shown as smooth continuous curves for visual clarity; however the actual responses are sequences of discrete, discontinuous samples.

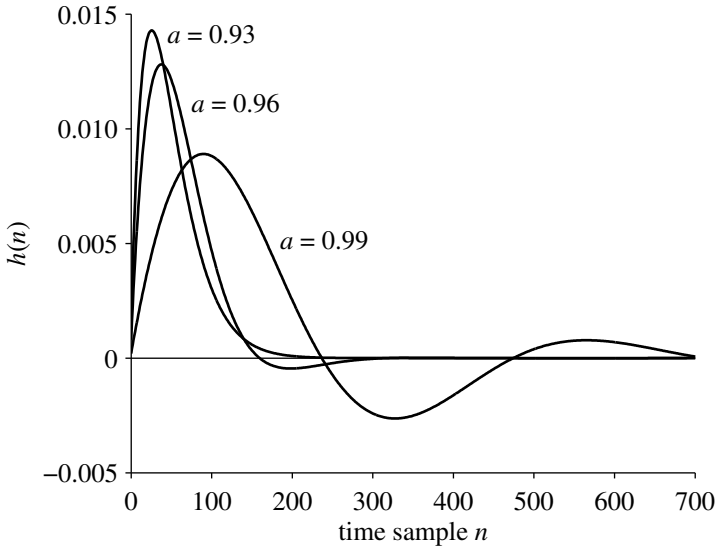


Figure 4.7: Underdamped closed loop impulse response.

The discrete step response $y(n)$ for a given causal impulse response $h(n)$ is defined in the time domain as

$$y(n) = \mathrm{u}(n) * h(n) = \sum_{k=-\infty}^{\infty} \mathrm{u}(n-k)h(k) = \sum_{k=0}^{n} h(k)\,, \tag{4.46}$$

where $*$ denotes discrete convolution. Applying (4.46) to (4.45) gives the step response for the underdamped case:

$$y(n) = \frac{bc}{bc+1}\,\frac{1}{1-2p\cos\omega_d + p^2}\left[1 - p^{n+1}\,\frac{\sin([n+2]\omega_d)}{\sin\omega_d} + p^{n+2}\,\frac{\sin([n+1]\omega_d)}{\sin\omega_d}\right]\mathrm{u}(n). \tag{4.47}$$

Figure 4.8 shows the step responses corresponding to each impulse response in Figure 4.7, where again the responses are shown as continuous functions for clarity.

An expression for the damping ratio ζ can be developed using the sampling rate-normalized natural resonant frequency ω_d in (4.43) and the impulse response expression

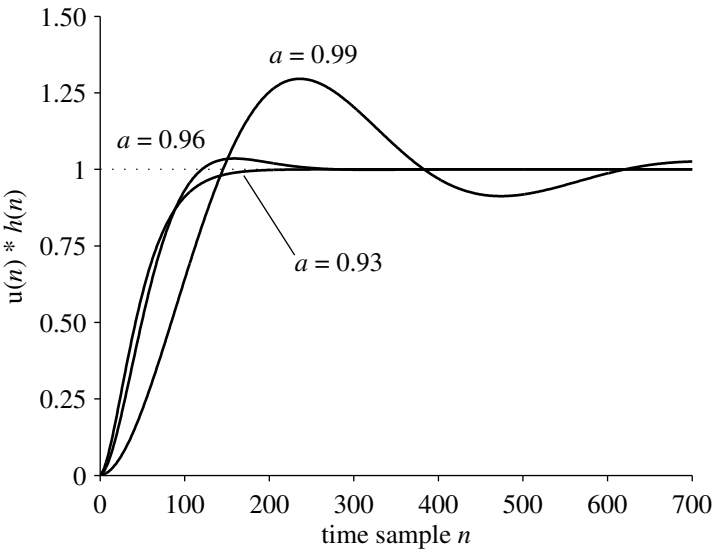


Figure 4.8: Underdamped closed loop step response.

(4.45). From harmonic analysis, the damping ratio present in an underdamped transient response $h(n)$ having natural resonant frequency ω_d can be determined from the ratio of successive damped sinusoid peaks $h(n_1)$ and $h(n_2)$ at discrete time samples n_1 and $n_2 = n_1 + 2\pi/\omega_d$ via the relation

$$\zeta = \frac{\ln \dfrac{h(n_1)}{h(n_2)}}{\sqrt{(2\pi)^2 + \ln^2 \dfrac{h(n_1)}{h(n_2)}}}. \tag{4.48}$$

In terms of the complex pole magnitude p defined in (4.42) and ω_d, (4.48) becomes

$$\zeta = \frac{\ln \dfrac{1}{p}}{\sqrt{{\omega_d}^2 + \ln^2 \dfrac{1}{p}}}. \tag{4.49}$$

For the sake of completion, (4.49) can be expanded in terms of the loop gains a, b and $c = k_A k_D k_L$, which produces

$$\zeta = \frac{\ln \sqrt{\frac{bc+1}{a}}}{\sqrt{\left[\tan^{-1}\left(\frac{\sqrt{4abc-(1-a)^2}}{a+1}\right)\right]^2 + \ln^2 \sqrt{\frac{bc+1}{a}}}}. \tag{4.50}$$

Fortunately, the complexity of (4.50) can be avoided with almost no loss in accuracy using the bilinear approximation for ζ in (4.28), or with only slight inaccuracy using the linear approximation in (4.33) based upon the previous analysis surrounding Figure 4.5(a).

4.6 Critically-Damped Transient Analysis

The closed loop response exhibits critically-damped characteristics when

$$(1-a)^2 = 4abc\,, \tag{4.51}$$

which after combining with the loop filter DC unity gain constraint $a + b = 1$ gives precise values for a and b of

$$a = \frac{1}{1+4c} \qquad\qquad b = \frac{4c}{1+4c}\,, \tag{4.52}$$

as listed previously in (4.18). For this case the poles are real and identical:

$$p_1 = p_2 = \frac{a+1}{2(bc+1)}. \tag{4.53}$$

From an implementation standpoint, the critically-damped case represents a somewhat infinitesimal, nonexistent boundary between underdamped and overdamped response types, in light of the coefficient $c = k_A k_D k_L$ being system-*defined* more so than system-*selectable*. The resulting precision required on loop filter gains a and b, selected *a posteriori* to c, in general will not be sufficiently fine-grained to meet the requirement in (4.51) with equality.

4.6.1 Stability

For the critically-damped case, causal stability requires that

$$a + 1 < 2(bc + 1)\,, \tag{4.54}$$

which is satisfied for all $(a, b, c) \in (0, 1)$ since it is clear that $a + 1 < 2 < 2 + 2bc$. Thus the critically-damped response is unconditionally stable.

4.6.2 Impulse and Step Responses

For the critically-damped response, the transfer function reduces to

$$H(z) = \frac{\dfrac{bc}{bc+1}}{(1 - pz^{-1})^2}\,, \tag{4.55}$$

where $p = p_1$ in (4.53). The associated impulse response can be quickly determined by first dividing the transform pair in (4.44) by $\sin\omega_d$ and evaluating the result in the limit as ω_d goes to zero:

$$\lim_{\omega_d \to 0} \frac{p^n \sin(n\omega_d)}{\sin\omega_d} = np^n \lim_{\omega_d \to 0} \frac{\cos(n\omega_d)}{\cos\omega_d} = np^n\,, \tag{4.56}$$

where l'Hôpital's rule was used in the first equality. This gives a modified z-transform pair of

$$np^n\,\mathrm{u}(n) \quad \overset{\mathcal{Z}}{\longleftrightarrow} \quad \frac{pz^{-1}}{(1-pz^{-1})^2}. \tag{4.57}$$

Compensating for the z^{-1} factor in the z-transform numerator of (4.57) with a time domain index shift of $n+1$, and dividing both sides by p gives the critically-damped impulse response:

$$h(n) = \frac{bc}{bc+1}\,(n+1)p^n\,\mathrm{u}(n). \tag{4.58}$$

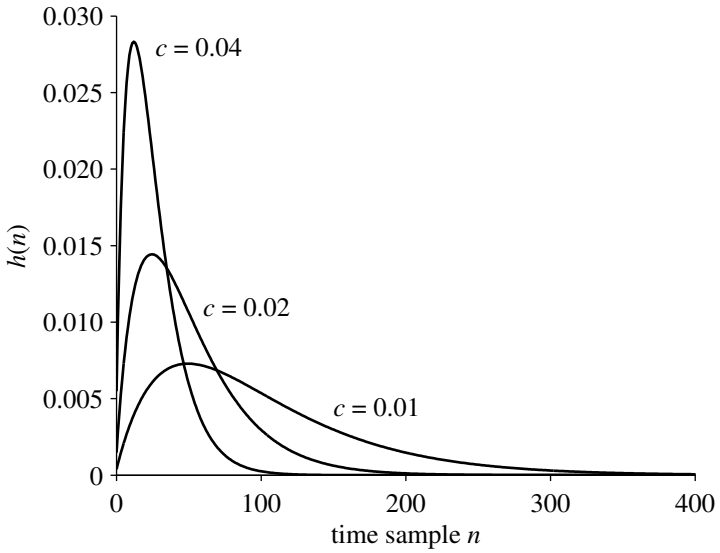


Figure 4.9: Critically-damped closed loop impulse response.

Figure 4.9 shows critically-damped impulse responses plotted for loop gain (a, b, c) triples of $(\frac{100}{104}, \frac{4}{104}, 0.01)$, $(\frac{100}{108}, \frac{8}{108}, 0.02)$ and $(\frac{100}{116}, \frac{16}{116}, 0.04)$, where the value of c is varied to produce each response, and the values of a and b are determined using (4.52). The impulse response peak (and subsequently the response speed, or rise time) is directly proportional to c, thus the response having highest peak in the figure belongs to the coefficient set having $c = 0.04$. As with the underdamped response plots, the critically-damped im-

pulse response plots are graphed as continuous curves for clarity, however the true nature of the responses are discrete in time.

The critically-damped step response is determined by taking the discrete convolution of the unit-step function $\mathrm{u}(n)$ and the impulse response in (4.58). Alternatively, the underdamped step response in (4.47) can be evaluated in the limit as ω_d goes to zero as was done to determine the critically-damped impulse response; using either method gives a critically-damped step response of

$$y(n) = \frac{bc}{bc+1} \frac{1-(n+2)p^{n+1}+(n+1)p^{n+2}}{(1-p)^2} \mathrm{u}(n)\,, \tag{4.59}$$

where p is a function of a, b and c given in (4.53).

Figure 4.10 shows the step responses corresponding to the critically-damped impulse responses in Figure 4.9. As noted in the impulse response plots discussion, higher values of c correspond to faster step responses given the characteristics of the closed BERLL. It is again noted that the step response curves appear to be continuous, but are actually piecewise linear threadings through discrete points in time.

4.7 Overdamped Transient Analysis

The final response type to consider is the overdamped case, which occurs for values of $a, b, c \in (0,1)$ such that

$$(a-1)^2 > 4abc. \tag{4.60}$$

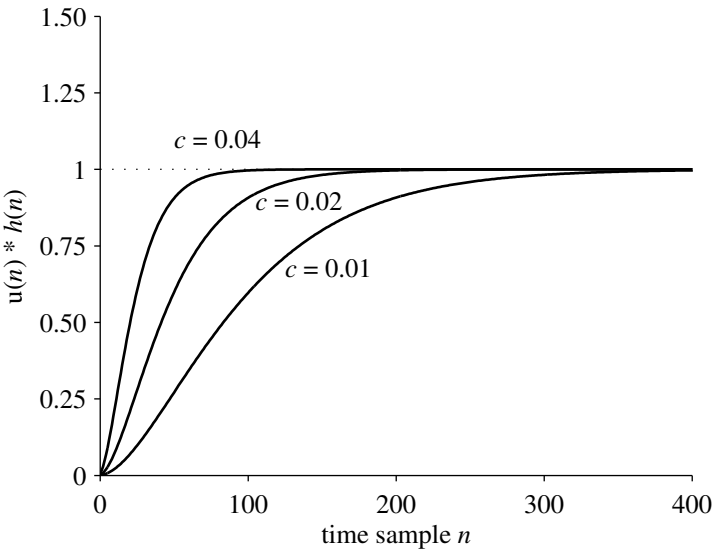


Figure 4.10: Critically-damped closed loop step response.

Coupling (4.60) with the DC unity gain loop filter constraint $a + b = 1$ gives the limitations on a and b in terms of c that produce an overdamped closed loop response of

$$a < \frac{1}{1+4c} \qquad\qquad b > \frac{4c}{1+4c}\,, \tag{4.61}$$

which is repeated here, first discussed in (4.18). For the overdamped response, the poles p_1 and p_2 are real and distinct:

$$\begin{aligned} p_1 &= \frac{a+1}{2(bc+1)} + \frac{\sqrt{(a-1)^2 - 4abc}}{2(bc+1)} \\ p_2 &= \frac{a+1}{2(bc+1)} - \frac{\sqrt{(a-1)^2 - 4abc}}{2(bc+1)}. \end{aligned} \tag{4.62}$$

4.7.1 Stability

It is clear from (4.62) that $p_1 > p_2$ for $a, b, c \in (0, 1)$ such that $(a-1)^2 > 4abc$. As such it is sufficient to examine the size of p_1, and if less than one conclude that both p_1 and

p_2 are stable roots for a causal response. Examining p_1, causal stability requires that

$$a + 1 + \sqrt{(a-1)^2 - 4abc} < 2(bc+1)\,,$$

which simplifies to

$$bc(bc+1) > 0\,,$$

which holds for all b and c of interest. Thus p_1 and p_2 are strictly less than one, and as a result the overdamped response is unconditionally stable. Note that as with the underdamped and critically-damped response types, stability for the overdamped response does not require the additional DC unity gain restriction on a and b, $a + b = 1$.

4.7.2 Impulse and Step Responses

The transfer function for the overdamped case becomes

$$H(z) = \frac{bc}{bc+1}\,\frac{1}{(1-p_1 z^{-1})(1-p_2 z^{-1})}\,, \tag{4.63}$$

which can be rewritten using the method of partial fractions as

$$H(z) = \frac{bc}{bc+1}\,\frac{1}{p_1 - p_2}\left(\frac{p_1}{1-p_1 z^{-1}} - \frac{p_2}{1-p_2 z^{-1}}\right). \tag{4.64}$$

The inverse z-transform corresponding to (4.64) is the overdamped impulse response:

$$h(n) = \frac{bc}{bc+1} \frac{{p_1}^{n+1} - {p_2}^{n+1}}{p_1 - p_2} \, \mathrm{u}(n) \,, \tag{4.65}$$

which is the difference of two discrete, decaying exponentials.

Figure 4.11 shows the overdamped impulse responses for (a, b, c) coefficient triples of $(0.1, 0.9, 0.02)$ (response having highest peak value) and $(0.9, 0.1, 0.02)$ (response having lowest peak value). Despite the large variation in a, the two overdamped responses show little appreciable difference in overall shape. Thus for a fixed value of $c = k_A k_D k_L$, the overdamped response of the BERLL cannot be manipulated by changes to a or b with much effect.

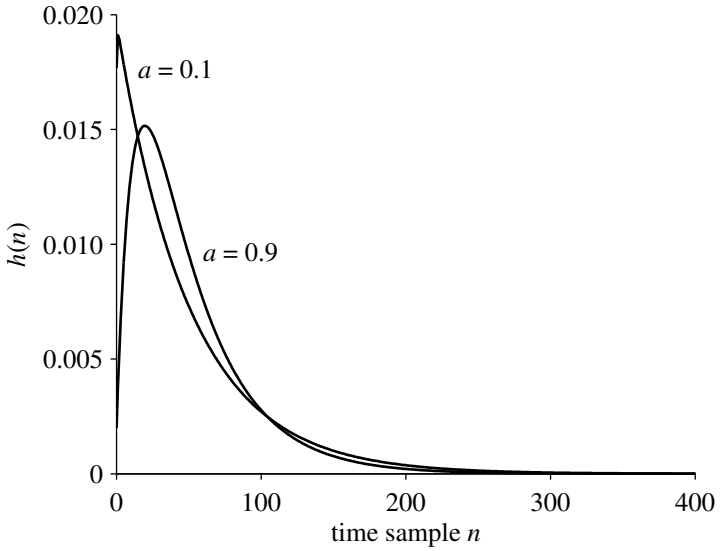


Figure 4.11: Overdamped closed loop impulse response.

Convolving the unit step function with (4.65) gives the overdamped step response expression:

$$y(n) = \frac{bc}{bc+1} \frac{1}{p_1 - p_2} \left[p_1 \frac{1 - p_1^{n+1}}{1 - p_1} - p_2 \frac{1 - p_2^{n+1}}{1 - p_2} \right] \mathrm{u}(n) \,, \tag{4.66}$$

where p_1 and p_2 are given in (4.62). Figure 4.12 shows the step responses for each impulse response in Figure 4.11. The step response having $a = 0.1$ hits the 50% rise time point first, but is quickly overtaken by the response having $a = 0.9$.

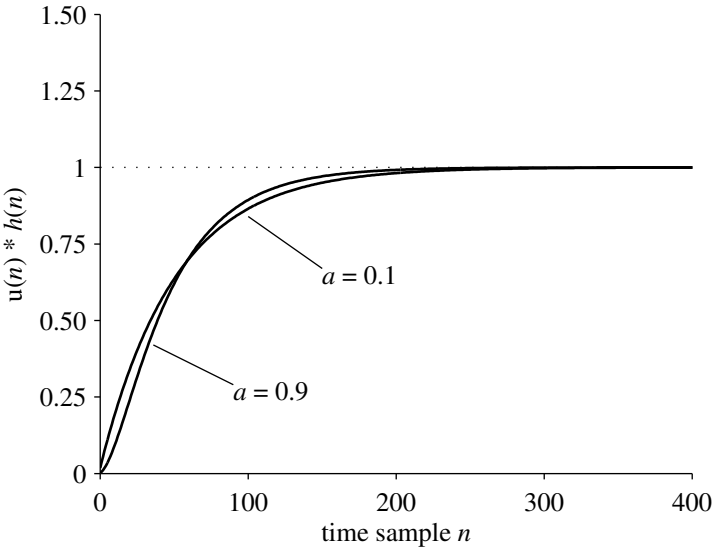


Figure 4.12: Overdamped closed loop step response.

4.8 Summary

Having developed and examined the linear discrete time model for the BERLL in this chapter, the next two chapters examine actual loop behavior in the presence of noise via Monte Carlo simulation of a WiMAX-class log-likelihood ratio LDPC decoder, using two different configuration parameters for the decoder. The response types studied in the present chapter will be immediately recognizable in the simulation results.

Tables 4.1–4.4 contain the prominent equations of interest for each transient response type developed in this chapter.

Table 4.1: Linear analysis equation summary I:
loop filter coefficient limits, poles, complex exponential terms.

Type	a, b Limits	Poles	p, ω_d
Underdamped	$a > \frac{1}{1+4c}$ $b < \frac{4c}{1+4c}$	$p_1 = \frac{a+1}{2(bc+1)} + j\frac{\sqrt{4abc-(1-a)^2}}{2(bc+1)}$ $p_2 = \frac{a+1}{2(bc+1)} - j\frac{\sqrt{4abc-(1-a)^2}}{2(bc+1)}$	$p = \sqrt{\frac{a}{bc+1}}$ $\omega_d = \tan^{-1}\left(\frac{\sqrt{4abc-(1-a)^2}}{a+1}\right)$
Critically-damped	$a = \frac{1}{1+4c}$ $b = \frac{4c}{1+4c}$	$p_1 = p_2 = \frac{a+1}{2(bc+1)}$	$p = \frac{a+1}{2(bc+1)}$ $\omega_d = 0$
Overdamped	$a < \frac{1}{1+4c}$ $b > \frac{4c}{1+4c}$	$p_1 = \frac{a+1}{2(bc+1)} + \frac{\sqrt{(1-a)^2-4abc}}{2(bc+1)}$ $p_2 = \frac{a+1}{2(bc+1)} - \frac{\sqrt{(1-a)^2-4abc}}{2(bc+1)}$	N/A

Table 4.2: Linear analysis equation summary II:
transfer functions and impulse responses.

Type	Transfer Function $H(z)$	Impulse Response $h(n)$
Underdamped	$\frac{\frac{bc}{bc+1}}{(1-pe^{j\omega_d}z^{-1})(1-pe^{-j\omega_d}z^{-1})}$	$\frac{bc}{bc+1}\frac{\sin([n+1]\omega_d)}{\sin\omega_d}\,p^n\,\mathrm{u}(n)$
Critically-damped	$\frac{\frac{bc}{bc+1}}{(1-pz^{-1})^2}$	$\frac{bc}{bc+1}\,(n+1)\,p^n\,\mathrm{u}(n)$
Overdamped	$\frac{\frac{bc}{bc+1}}{(1-p_1z^{-1})(1-p_2z^{-1})}$	$\frac{bc}{bc+1}\,\frac{{p_1}^{n+1}-{p_2}^{n+1}}{p_1-p_2}\,\mathrm{u}(n)$

Table 4.3: Linear analysis equation summary III: step responses.

Type	Step Response $y(n)$
Underdamped	$\frac{bc}{bc+1}\ \frac{1}{1-2p\cos\omega_d+p^2}\left[1-p^{n+1}\ \frac{\sin([n+2]\omega_d)}{\sin\omega_d}+p^{n+2}\ \frac{\sin([n+1]\omega_d)}{\sin\omega_d}\right]\mathrm{u}(n)$
Critically-damped	$\frac{bc}{bc+1}\ \frac{1-(n+2)p^{n+1}+(n+1)p^{n+2}}{(1-p)^2}\,\mathrm{u}(n)$
Overdamped	$\frac{bc}{bc+1}\ \frac{1}{p_1-p_2}\left[p_1\ \frac{1-{p_1}^{n+1}}{1-p_1}-p_2\ \frac{1-{p_2}^{n+1}}{1-p_2}\right]\mathrm{u}(n)$

Table 4.4: Linear analysis equation summary IV:
natural resonant frequency and damping ratio approximations.

Approximation Method	Natural Resonant Frequency ω_d	Damping Ratio ζ
Exact	$\frac{1}{T}\tan^{-1}\left(\frac{\sqrt{4abc-(1-a)^2}}{a+1}\right)$	$\frac{\ln\sqrt{\frac{bc+1}{a}}}{\sqrt{\left[\tan^{-1}\left(\frac{\sqrt{4abc-(1-a)^2}}{a+1}\right)\right]^2+\ln^2\sqrt{\frac{bc+1}{a}}}}$
Linear	$\frac{1}{T}\frac{\sqrt{4abc-(1-a)^2}}{2a}$	$\frac{1-a}{2\sqrt{abc}}$
Bilinear	$\frac{2}{T}\frac{\sqrt{4abc-(1-a)^2}}{bc+2(1+a)}$	$\frac{bc+1-a}{\sqrt{bc\left[bc+2(1+a)\right]}}$

Chapter 5

Method I : Feedback Using Variable Number of Decoder Iterations

To adapt the performance of an iterative ECC decoder, perhaps the most natural and easily-accessible parameter to adjust is the number of decoder iterations performed on each received codeword before producing an estimate. This method is ideal from the standpoint that no modifications are required to the underlying structure of the ECC or to the mechanics of the decoder. Figure 5.1 shows the P_e versus E_b/N_0 waterfall performance plot for 1, 5 and 25 decoder iterations for the rate 1/2 (576, 288) WiMAX LDPC code using BPSK modulation over the AWGN channel. For this method, parameter variation begins at 1 iteration and ends at the decoder iteration number at which further increases in the parameter produce no noticeable error correction improvement, typically between 25 and 30 iterations for this particular WiMAX code.

5.1 Decoder LLR Transfer Curve

As discussed in Chapter 3, each specific decoder parameter has an LLR transfer curve relating input parameter value to decoder output LLR magnitude. Furthermore, the analysis

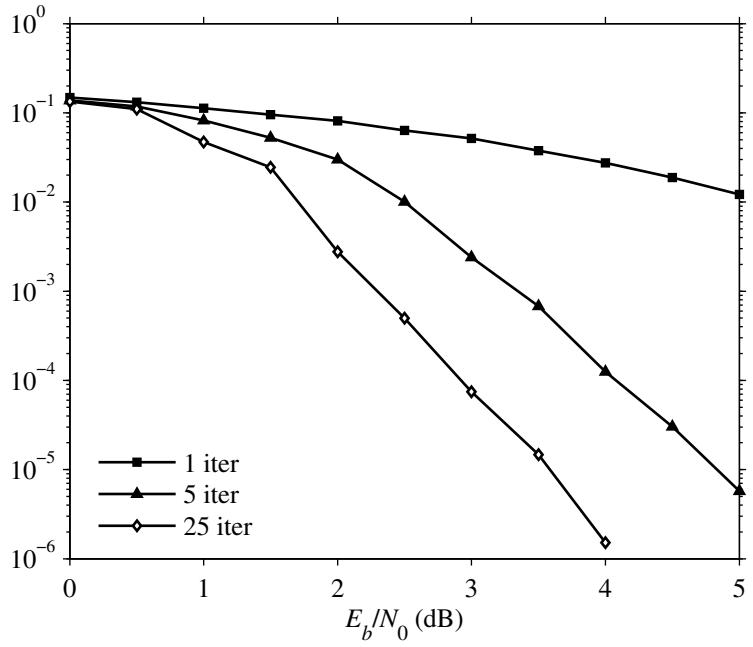


Figure 5.1: Probability of error versus energy-to-noise ratio for the rate 1/2 (576, 288) WiMAX LDPC code for 1, 5 and 25 decoder iterations.

in Chapter 4 presumed a linear relationship between these two variables, embodied as the decoder loop gain k_D. A set of LLR transfer curves for method I is shown in Figure 5.2 over a range of E_b/N_0 levels from 0 dB to 4 dB in 1 dB increments. The transfer curves show a moderately-linear input/output relationship for E_b/N_0 values less than 2 dB, beyond which the linear simplification becomes less and less accurate. The 3 dB and 4 dB transfer curves reach saturation quickly, between 10 and 15 iterations. In contrast, the 0 dB and 1 dB transfer curves suggest a saturation point beyond the cap of 25 decoder iterations shown in the figure, while the 2 dB transfer curve appears to be at the edge of saturation at 25 decoder iterations. A more complete linear model would also include the E_b/N_0-dependent LLR magnitude bias present at 1 iteration.

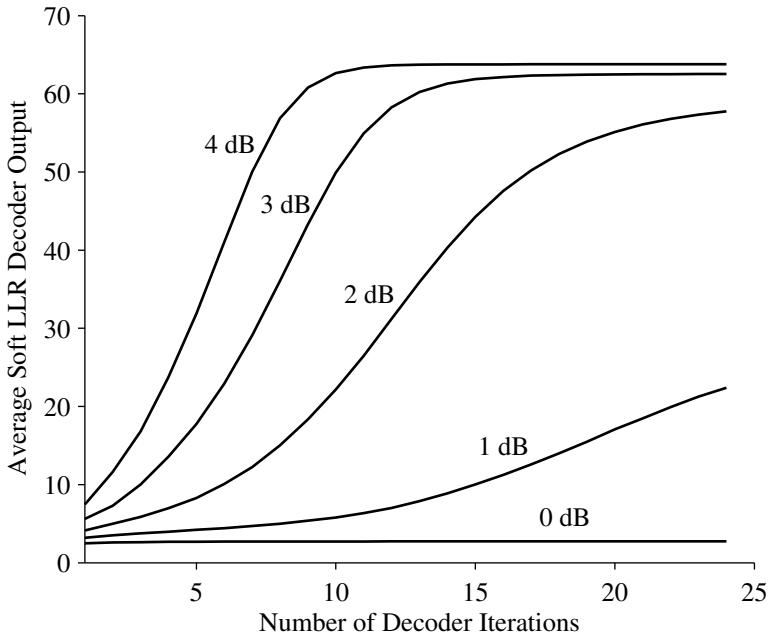


Figure 5.2: Decoder log-likelihood ratio versus number of decoder iterations.

5.2 Simulation Results

Monte Carlo simulations were performed to evaluate the performance of feedback method I. The complete closed loop BERLL circuit including LDPC decoder shown previously in Figure 4.1 was modeled in the C programming language. The BERLL was released for operation at time index 0, and given a target BER of $P_0 = 10^{-3}$ with a channel energy-to-noise ratio of $E_b/N_0 = 3$ dB. The loop filter was configured with two different sets of (a, b) coefficients to produce underdamped and critically-damped transient responses. The underdamped response corresponded to a coefficient pair of $(a, b) = (0.9, 0.1)$, while the critically-damped response corresponded to a coefficient pair of $(a, b) = (0.7, 0.3)$. The decoder output LLR gain factor was set at $k_L = 0.15$, while the post-accumulator gain factor was set at $k_A = 1.0$. The decoder gain factor k_D, modeled as a constant for the linear analysis in Chapter 4, is not a modifiable parameter for simulation since decoder intrinsic

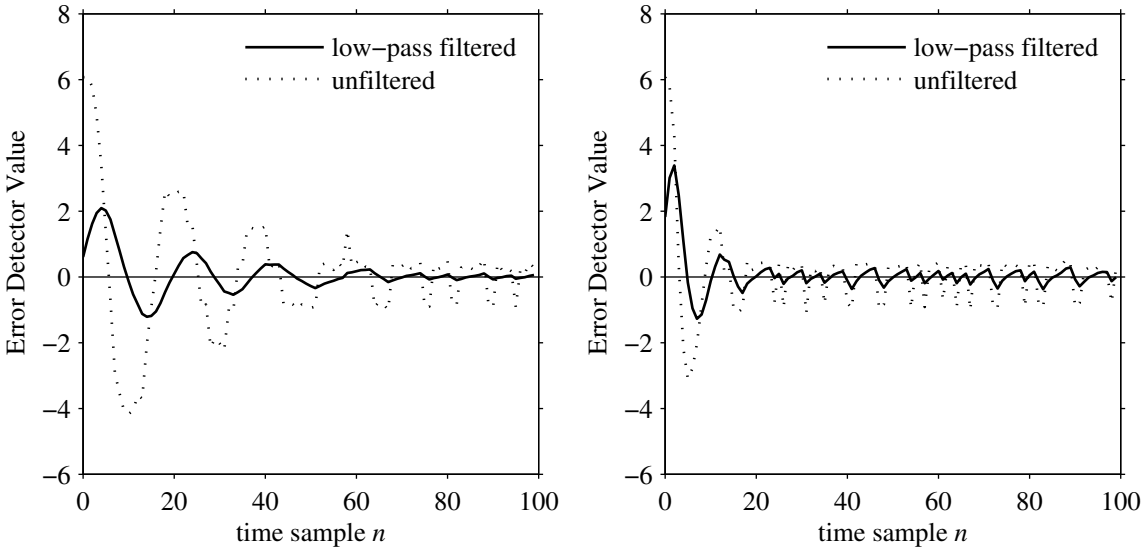
behavior determines this effective, nonlinear value as a function of parameter and E_b/N_0. Each BERLL update processed $N = 200$ decoder codewords, which along with 288 message bits per codeword resulted in a total of 57,600 LLR magnitudes being averaged to produce an LLR magnitude estimate for the BER error detector. This averaging procedure acts as an over-sampled, first-order filter of sorts to the aggregate loop and gives a noise suppression benefit. Additionally, the time index sample in the waveform plots that follow in this chapter should be multiplied by a factor of $1/N = 1/200$ when converting to a physical time, in seconds. Table 5.1 summarizes the simulation parameters for feedback method I.

Table 5.1: Feedback method I simulation parameters.

Parameter	Value
P_0	1×10^{-3}
E_b/N_0	3 dB
$\|L_0\|$	6.907 (derived)
(a, b)	$\begin{cases} (0.9, 0.1), \text{underdamped} \\ (0.7, 0.3), \text{critically-damped} \end{cases}$
k_A	1.0
k_L	0.15
N	200 codewords

Releasing the system at time index 0 with a fixed P_0 level is equivalent to injecting a step input on the $|L_0|$ signal. After a suitable number of loop iterations, the signal $|L_s|$ will converge to $|L_0|$ for a properly designed circuit. Equivalently, the error signal ε and filtered error signal ε_F will converge to an average value of zero. Figure 5.3 plots the unfiltered and filtered error signals for (a, b) loop coefficient pair values of $(0.9, 0.1)$ (left) and $(0.7, 0.3)$ (right); in the figure, sample markers are omitted to allow better visibility

of the two overlaid error waveforms. The general predictions of the linear analysis in Chapter 4 appear to hold, where the coefficient pair with smaller a value gives a (nearly) critically-damped response, while the coefficient pair having larger a value produces an underdamped response.

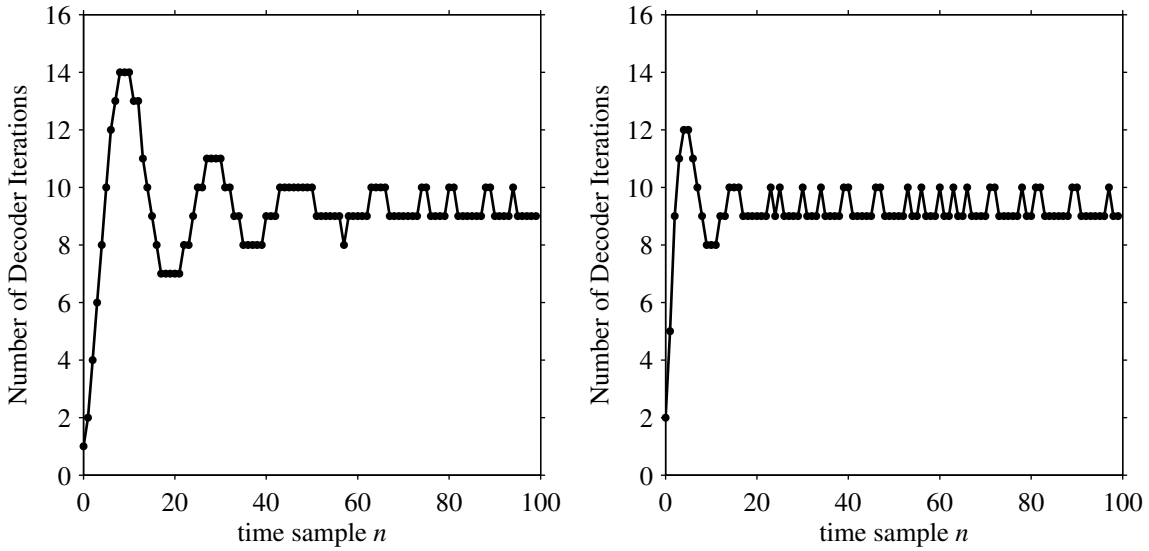


(a) Underdamped (b) Critically-damped

Figure 5.3: Method I BER error signals versus loop sample index for (a) underdamped and (b) critically-damped loop gain values.

For the specific simulation parameters chosen in Figure 5.3, a less-damped response is seen than what is expected based on the linear analysis; this is attributed in part to the E_b/N_0 value of 3 dB causing the decoder LLR versus input parameter curve to be outside its pseudo-linear region, where LLR saturation is reached earlier in the input parameter range compared to the linear case. The beneficial effects of the low-pass loop filter are easily seen when comparing the unfiltered and filtered error signals; notably the attenuation and signal smoothing (high-frequency intermediate sample-to-sample signal jumps are essentially missing in the underdamped filtered error output). A phase delay of about 5 samples, peak-

to-peak, for the underdamped response and 3 samples for the critically-damped response can also be observed between filter input and filter output waveforms.

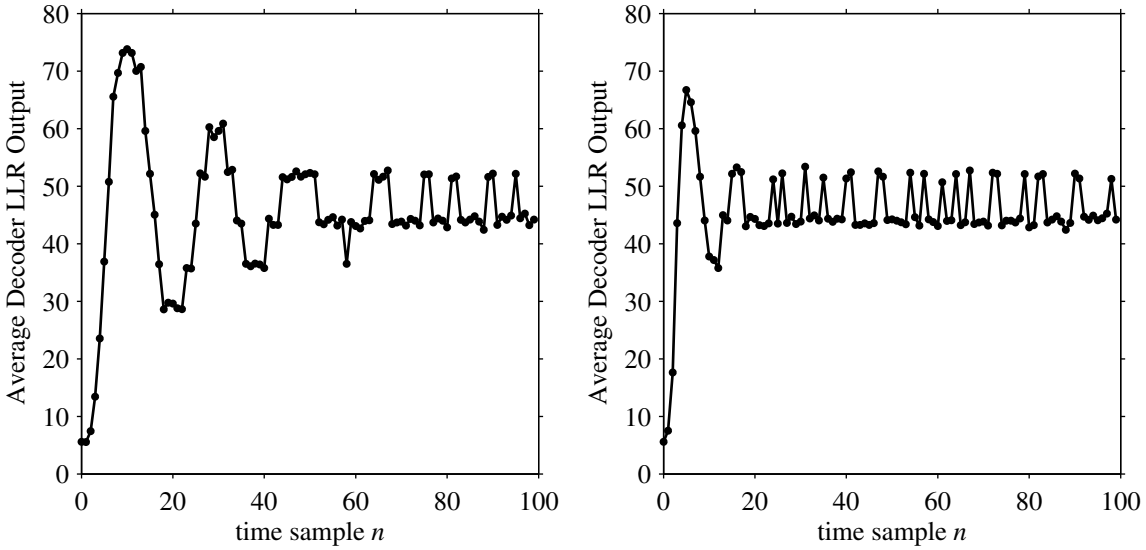


(a) Underdamped (b) Critically-damped

Figure 5.4: Method I number of decoder iterations versus loop sample index for (a) underdamped and (b) critically-damped loop gain values.

Figure 5.4 compares the discrete-valued decoder iteration number—the input to the decoder for method I—between the underdamped and critically-damped cases. After about 55 samples for the underdamped case and 15 samples for the critically-damped case, the loop settles around a value of 9 or 10 decoder iterations required to achieve the target BER of 10^{-3} at an E_b/N_0 of 3 dB. The quantization toggling between 9 and 10 iterations is evident, as the granularity available to the loop is not fine enough to achieve zero error for an integer number of decoder iterations. In essence, the loop will place the correct *duty cycle* on the iterations parameter between 9 and 10 so that over time, an effective fractional number of iterations can be computed that would render zero average detector error. Applying this reasoning to Figure 5.4(b), once the loop reaches steady state, the ratio of 9-iteration to 10-iteration samples is $64/24$, which suggests an *ideal* fractional iteration

value of $\frac{64}{64+24}\times 9 + \frac{64}{64+24}\times 10 \approx 9.27$. To a limited extent the layered, sub-decoder iteration techniques described in [28] and [29] can be viewed as a class of methods that employ fractional iteration values, and as such are worth consideration to combat quantization effects. In any event, quantization error on the iteration parameter cannot be completely avoided since it is an inherently discrete number.

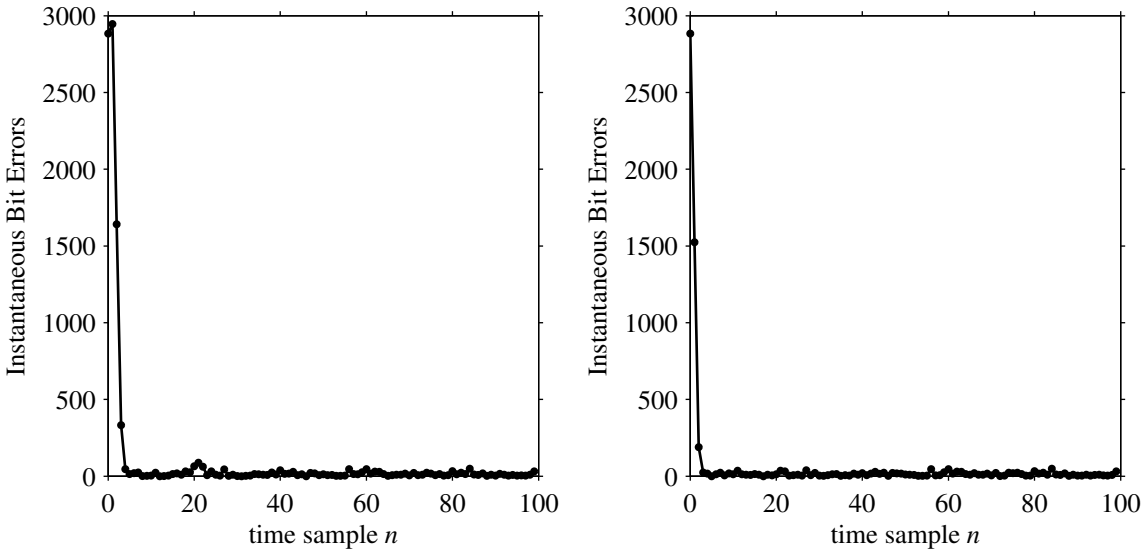


(a) Underdamped (b) Critically-damped

Figure 5.5: Method I decoder output log-likelihood ratio magnitude versus loop sample index for (a) underdamped and (b) critically-damped loop gain values.

Figure 5.5 shows the decoder output LLR magnitude for the underdamped and critically-damped cases. The same general trends are seen with LLR magnitude as were observed with number of decoder iterations. The quantization limitation on iteration number is immediately seen in the LLR waveforms, which bounce between roughly 45 and 55 due to the loop not being able to tune iteration number to the precise fractional value required for zero instantaneous error. It can be seen that additional LLR integer resolution is certainly available and could be utilized if the granularity on iteration number could be increased. Note

that the LLR magnitude waveforms in the plots will be scaled by k_L prior to comparison with the ideal, target LLR $|L_0|$ which becomes the loop error signal, ε.

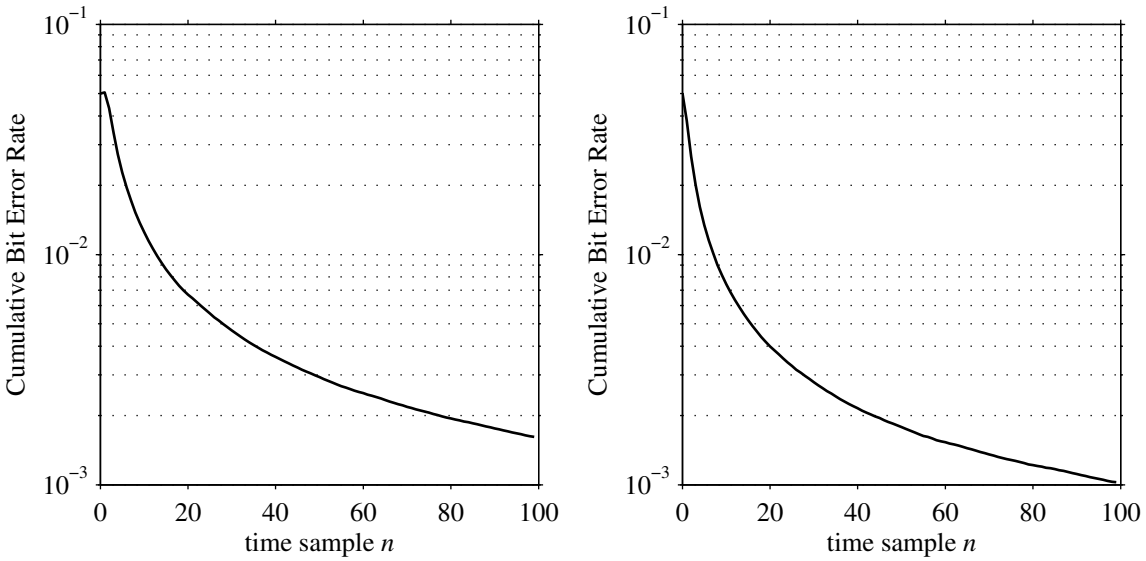


(a) Underdamped (b) Critically-damped

Figure 5.6: Method I instantaneous bit error count versus loop sample index for (a) underdamped and (b) critically-damped loop gain values.

Figure 5.6 shows the instantaneous number of bit errors for the underdamped and critically-damped cases. These values are available by simply comparing the hard-decision decoder output with the known message bit sequence; in an actual implementation this knowledge is of course unavailable, in-band. Both loop configurations act to quickly tune the decoder to mitigate bit error count. The underdamped configuration is seen to take several more samples to match the bit error suppression of the critically-damped case. The subtle bit error count *bump* for the underdamped case just beyond the time index of 20 is coincident with the accumulated dip that occurs in the iteration number parameter (as well as LLR magnitude) during the first sinusoidal trough between 15 and 25 samples.

Figure 5.7 is the final plot set for the feedback method I discussion, and is of particular interest since it confirms the loop is heading toward steady state conditions that provide



(a) Underdamped (b) Critically-damped

Figure 5.7: Method I cumulative bit error rate versus loop sample index for (a) underdamped and (b) critically-damped loop gain values.

a net bit error rate at or below (better than) the required 10^{-3} target. In both plots, the ordinate axis is placed on a logarithmic scale to appreciate the steady exponential decline of the cumulative error rate. The convergence speed of the critically-damped case over the underdamped case is also confirmed in the BER waveforms. Specifically, the critically-damped loop achieves a 2×10^{-3} BER in just over 40 samples, while the underdamped loop needs around 80 samples to reach this same BER. The trade-off to this convergence speed advantage is reduced noise suppression, as discussed previously in Chapter 4, and as seen in the comparison plots of this chapter.

In summary, the simulation results presented in this chapter establish the method of regulating the BER performance of an iterative, log-likelihood ratio based error correction decoder by constructing an error signal comparing decoder output LLR magnitude to an ideal LLR, and feeding back a conditioned and accumulated version of this error signal to the decoder, which effectively actuates the number of decoder iterations executed to

decode each received codeword. The next chapter examines an alternative method to the present, where a controlled number of parity check bits are nulled as a mechanism to tune the decoder BER.

Chapter 6

Method II : Feedback Using Variable Parity Bit Nulling

A technique commonly used to extend or shorten (puncture) a code is to add or remove, respectively, parity bits from transmitted codewords prior to transmission [25]. The decoder then takes these additional or missing parity bits into consideration when decoding. In the case of parity bit deletion, the codeword length is decreased in the redundancy portion of the word, which weakens the correction strength of the code. For punctured codes, one method of decoding entails inserting null symbols into the positions where parity bits were removed by the encoder, and allowing the decoder to process the received codeword as usual. In essence, removed parity bits are treated as ambiguous symbols, and can be thought of as victims of equal but opposite channel noise that perfectly cancels bit energy over a correlation window. Their negative impact on the decoding algorithm will be most severe on check nodes to which they are directly adjacent, with the detriment decreasing with graphical distance from these nodes. In numerical terms, the LLR values of nulled parity bits are initialized to zero, which corresponds to a likelihood ratio of 1.

For feedback method II, it is presumed that the codeword is transmitted in its entirety with all message and parity bits intact. The BERLL feedback circuitry then incrementally

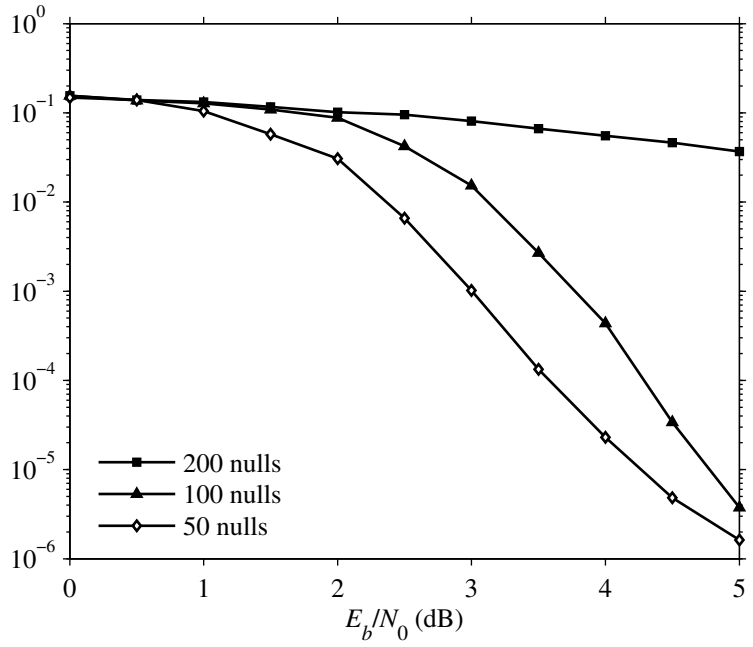


Figure 6.1: Probability of error versus energy-to-noise ratio for the rate 1/2 (576, 288) WiMAX LDPC code for 200, 100 and 50 nulled parity bits.

nulls a sufficient number of received parity bits to achieve the target BER on decoded message bits. The message bits are not nulled simply for the sake of keeping their channel estimates as large as possible, although it is certainly reasonable to expect the decoder to form a decent estimate of a nulled message bit using adjacent codeword bits in the parity check graph. In terms of implementation benefits, as parity bits are nulled, sections of arithmetic and storage logic can be disabled or *shutdown* to save power under high E_b/N_0 conditions, and incrementally enabled as needed under low E_b/N_0 conditions.

Figure 6.1 shows three members of the set of P_e versus E_b/N_0 waterfall performance plots for 200, 100 and 50 nulled parity bits for the common rate 1/2 (576, 288) WiMAX LDPC code used throughout this dissertation for closed loop simulation. The number of decoding iterations used for the waterfall plots was set to 15, which is also the value used

for iteration number to examine loop response for feedback method II. As expected, BER performance decreases as the number of nulled parity bits increases.

6.1 Decoder LLR Transfer Curve

Figure 6.2 plots the average LLR magnitude value versus the number of nulled parity bits for the rate 1/2 (576, 288) code for E_b/N_0 ratios from 0 dB to 4 dB in 1 dB increments, and is the method II analogous plot to Figure 5.2 for method I. Comparing Figure 6.2 with Figure 5.2, one of the most obvious differences is the negative slope of the method II plots; for the closed loop this will require a negative value for the post-accumulator gain, k_A, to achieve negative feedback and thus loop convergence. Secondly, the method II parameter range of 0 to 288 (in general for an (n, k) code, 0 to $n - k$) offers a much larger parameter selection size compared to the relatively smaller number of usable values for the decoder iteration parameter of method I. It is also worth noting the larger range of E_b/N_0 values for which the method II LLR transfer plots maintain a near-linear trend versus number of nulled parity bits, compared to the method I counterpart. This extended transfer curve linearity will produce a loop response for method II that is closer to the linear approximation analysis in Chapter 4 than was achieved for method I.

6.2 Simulation Results

Monte Carlo simulations were conducted to validate the use of nulling a variable number of parity bits as a decoder BER tuning mechanism. The C-language software used for

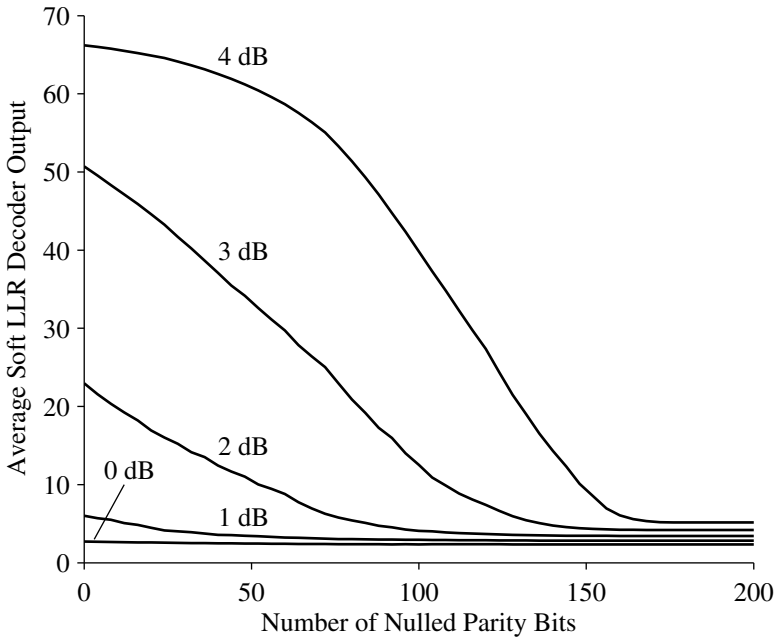


Figure 6.2: Decoder log-likelihood versus number of nulled parity bits.

method I simulations was also used for method II. As with method I, the BER target was set at $P_0 = 10^{-3}$ and the channel energy-to-noise ratio was set at $E_b/N_0 = 3$ dB. Due to the different underlying LLR transfer characteristic for method II, the post-accumulator gain was set at $k_A = -5.0$, while the decoder output LLR gain was set to $k_D = 0.11$. To isolate the effects of varying the number of nulled parity bits on the loop response, the number of decoder iterations was kept constant at 15.

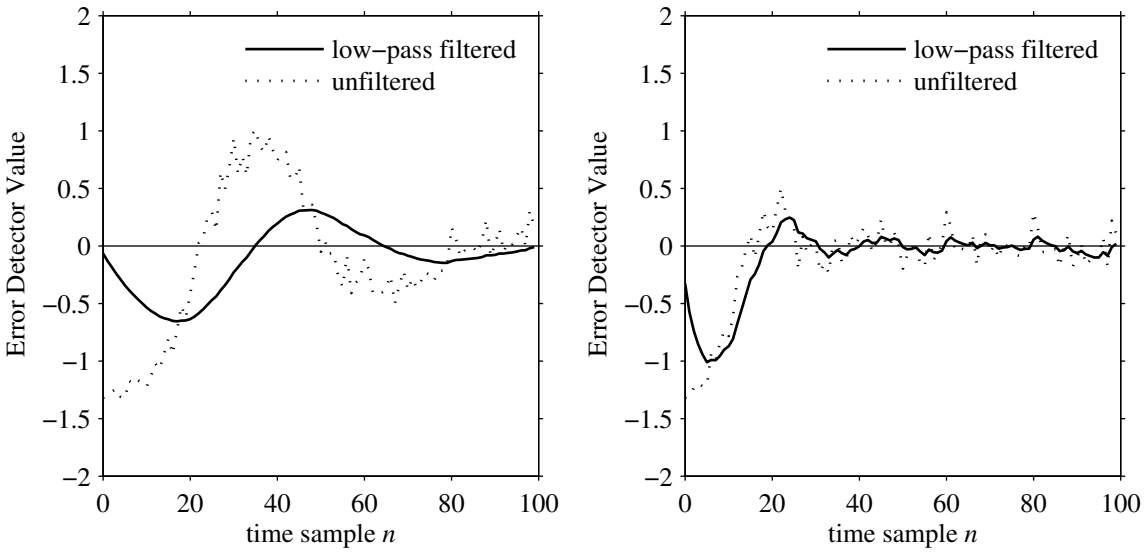
As with method I simulations, two different pairs of loop filter (a, b) coefficients were selected. The coefficients $(a, b) = (0.95, 0.05)$ were chosen to demonstrate an underdamped response, while $(a, b) = (0.75, 0.25)$ were chosen to demonstrate a (nearly) critically-damped response. Additionally, each BERLL update processed $N = 200$ decoder codewords, for a total of 57,600 LLR magnitudes being averaged to construct each LLR estimate when running the rate 1/2 (576, 288) LDPC code. Table 6.1 summarizes the simulation parameter setup for feedback method II.

Table 6.1: Feedback method II simulation parameters.

Parameter	Value
P_0	1×10^{-3}
E_b/N_0	3 dB
$\lvert L_0\rvert$	6.907 (derived)
(a, b)	$\begin{cases} (0.95, 0.05), \text{underdamped} \\ (0.75, 0.25), \text{critically-damped} \end{cases}$
k_A	-5.0
k_L	0.11
N	200 codewords

Figure 6.3 is the method II counterpart to method I Figure 5.3, and shows the unfiltered and filtered BER error detector signals for underdamped (left) and critically-damped (right) loop configurations. For the specific loop filter gains selected, the underdamped error settles after 100 samples, while the critically-damped error only takes about 30 samples to match the same error value as the underdamped case. The clear noise suppression advantage of the underdamped loop over the critically-damped loop is seen via the smooth, less noisy underdamped filtered error signal compared to the critically-damped version which allows a visible amount of high-frequency content to propagate to the output.

Figure 6.4 plots the number of nulled decoder parity bits—the loop control mechanism for method II—for the underdamped and critically-damped loops. The large range of available parameter variation on the nulled parity bit value allows for a continuous-time looking loop response, less-hampered by quantization effects. Both the underdamped and critically-damped loops settle at a value of about 60 nulled parity bits to achieve a 10^{-3} BER for an E_b/N_0 of 3 dB, using 15 decoder iterations. Unlike the method I setup, where the loop had to increase the decoder input parameter to improve BER performance since

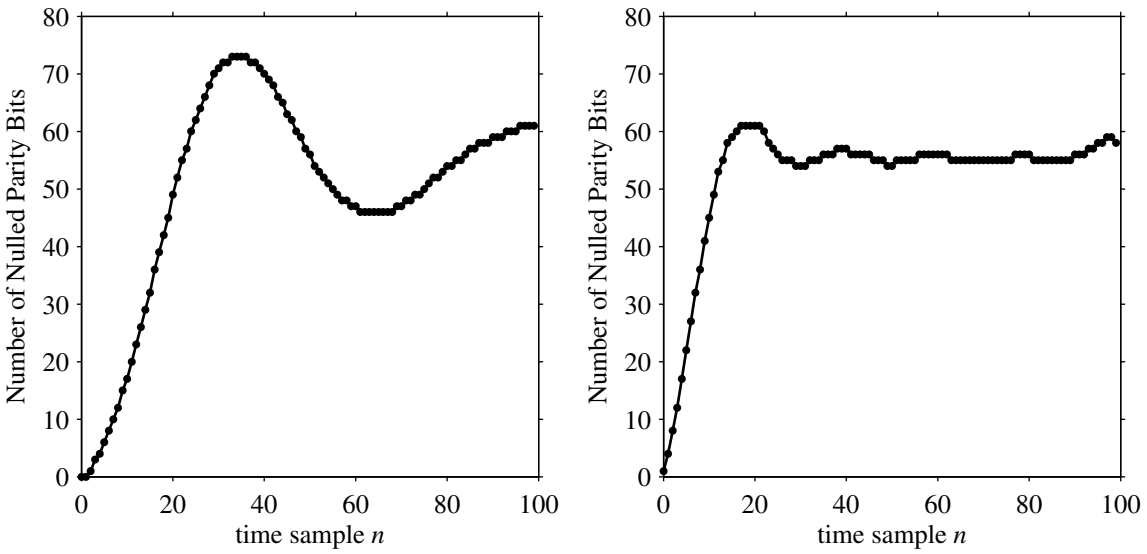


(a) Underdamped (b) Critically-damped

Figure 6.3: Method II BER error signals versus loop sample index for (a) underdamped and (b) critically-damped loop gain values.

the initial parameter value was too low, the method II loop decoder parameter is initialized to a value that produces the best BER performance, zero nulled parity bits. Thus the loop is initialized to an over-achieving state and quickly acts to intentionally degrade its performance, in a controlled manner. The equivalent arrangement for method I would be to initialize the decoder iteration number parameter to a large value, and then let the closed loop reduce it to lower the link quality (increase the BER).

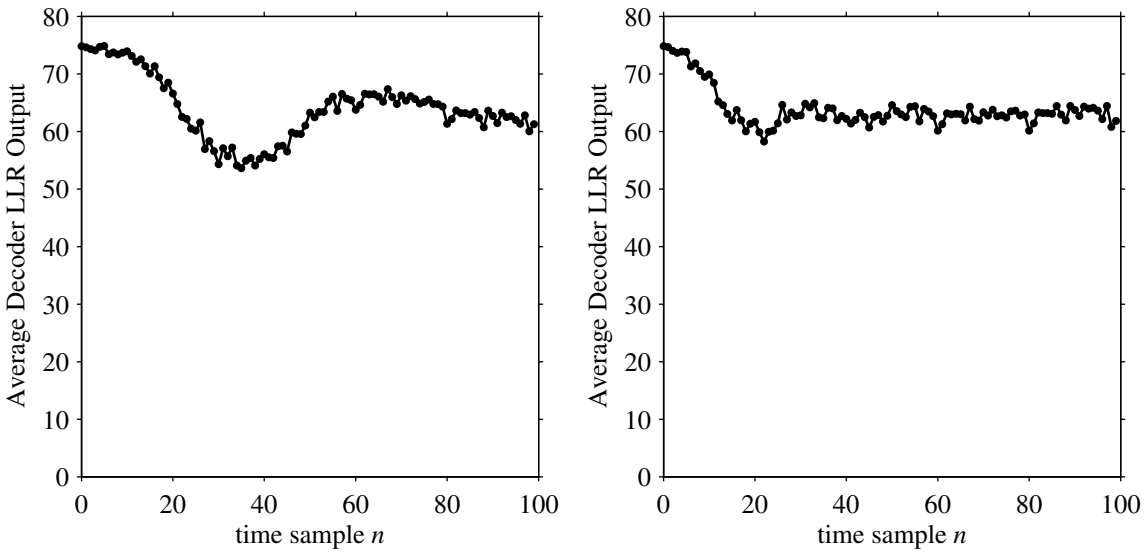
Figure 6.5 is the method II counterpart to Figure 5.5 for method I. Since the method II loop initializes the nulled parity bits parameter to its highest performance level (zero) as discussed above, the LLR values initiate at high levels as well. The quantization effects on the method II LLR graphs are essentially absent compared to the method I case, due to the high degree of granularity available on the decoder input parameter, ranging from 0 to 288 for method II compared with an approximate range of 1 to 30 for method I.



(a) Underdamped (b) Critically-damped

Figure 6.4: Method II number of decoder nulled parity bits versus loop sample index for (a) underdamped and (b) critically-damped loop gain values.

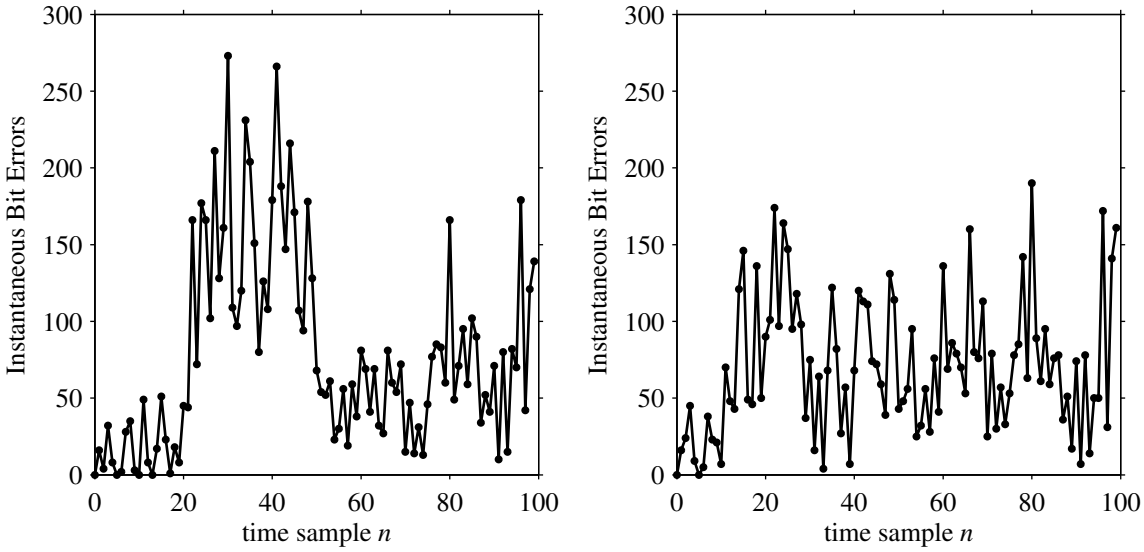
Figure 6.6 is the method II version of method I Figure 5.6, and shows the instantaneous bit error count at each loop time sample. Since the method II loop decoder input parameter is initialized to its maximum BER performance-yielding value, the initial bit errors occur less frequently than the permissible rate. The result after a few dozen time index increments is a controlled increase in the number of nulled parity bits, which is evident as a noticeable increase in the average number of bit errors on both the underdamped and critically-damped setups. For the underdamped case, the bit error count waveform crest between samples 35 and 40 is directly tied to the trough located in the LLR magnitude plot in Figure 6.5, which is in turn a result of the initial overshoot crest in the underdamped number of nulled parity bits in Figure 6.4. The initial value trend for the method II bit error count waveform is in contrast with the method I case, where initial bit errors were quite high relative to the target BER.



(a) Underdamped (b) Critically-damped

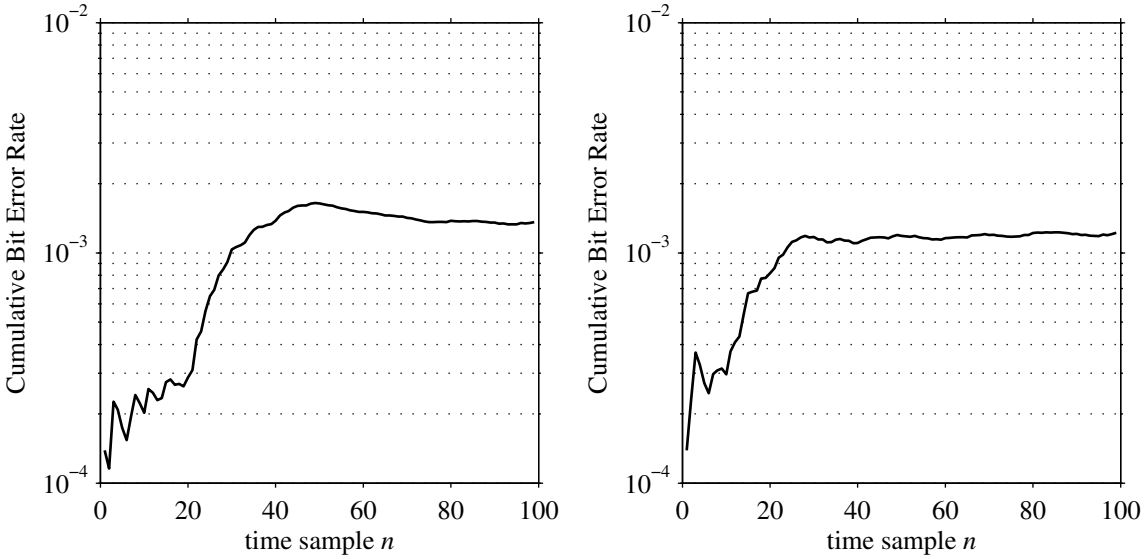
Figure 6.5: Method II decoder output log-likelihood ratio magnitude versus loop sample index for (a) underdamped and (b) critically-damped loop gain values.

Figure 6.7 is the final verification plot for method II and corresponds to Figure 5.7 for method I, providing confirmation that the cumulative, running BER for method II is approaching the target BER of 10^{-3}. Whereas the method I loop due to initial setup starts at poor BER performance followed by an exponential improvement with successive samples, the method II loop initially produces a BER of nearly 10^{-4}, an order of magnitude smaller than required. As far as the error rate is concerned, an overshoot for both underdamped and critically-damped configurations is seen in Figure 6.7. At the 100 sample mark, a slow but steady BER decline toward 10^{-3} is present. The conclusion reached for this chapter is that by varying the number of nulled parity bits used for error correction in a log-likelihood ratio-based decoder, it is possible to compare the scaled output LLR magnitudes from the decoder with an ideal LLR derived from a target BER for the end purpose of tuning the decoder using a standard feedback control loop to meet the target BER.



(a) Underdamped (b) Critically-damped

Figure 6.6: Method II instantaneous bit error count versus loop sample index for (a) underdamped and (b) critically-damped loop gain values.



(a) Underdamped (b) Critically-damped

Figure 6.7: Method II cumulative bit error rate versus loop sample index for (a) underdamped and (b) critically-damped loop gain values.

Chapter 7

Conclusions

7.1 Summary

In this dissertation, the feasibility of dynamically adjusting the bit error rate (BER) level on decoded bits output from an error correction code (ECC) decoder using a standard second order digital control system was explored. It was confirmed that the native decoder log-likelihood ratio (LLR) could be compared to a nominal LLR derived from a target BER to generate an error signal, which served as the control loop input. The output of the bit error rate locked loop (BERLL), after proper signal filtering, was appropriately scaled and passed back to a configuration port on the ECC decoder to adjust its error correction performance until the desired BER level was achieved.

The non-decoder portion of the loop used familiar signal processing blocks, specifically fixed gains, an infinite impulse response (IIR) loop filter and an accumulator. The topology of the BERLL was shown to be similar to that of an ordinary phase locked loop (PLL). For analysis purposes, the decoder was modeled as a fixed gain device, as far as input parameter to output LLR characteristics were concerned.

Two feedback methods were proposed and examined as proof-of-concept for the BERLL. For the first method, the loop automatically adjusted the number of decoder iterations to

control output BER. For the second method, the loop automatically nulled some number of parity bit positions until the desired BER was achieved. Loop coefficients were selected that demonstrated both underdamped and critically-damped transient response behavior.

7.2 Topics for Further Study

Many items remain to be explored at the conclusion of this dissertation. Viewing the output of the decoder, it is of interest to determine alternate means of estimating BER using existing decoder variables—or perhaps new variables created by observing real-time decoder behavior—and determining how to form error signals appropriate to the loop; the list of existing decoder stopping criterion in Chapter 3 would serve as a good starting point. Conversely, viewing the configuration side of the ECC decoder, it is of equal interest to explore and enumerate a larger set of decoder parameters that could potentially be used to adjust decoder output BER. Furthermore, determining efficient and effective methods of simultaneously changing *multiple* decoder parameters is also intriguing. In this last case, maintaining a decoder configuration state vector would be necessary.

Another avenue to explore would be using different data modulation and noise models other than BPSK over AWGN. In the present era of mobile devices, the Rayleigh fading channel model would be a good candidate as the channel degradation source. Alternate ECC technologies are also of interest, especially Turbo codes since they also rely on LLR variables for normal decoder operation.

As a final topic for further study, an implementation of the proposed BERLL concept using hardware or software operating on real signals would be of interest. With a real im-

plementation in hand, system-level variables such as power consumption, loop transient response time, etc., could be explored and quantified. From a hands-on engineering perspective, the implementation phase is perhaps the most interesting, as ideas on paper are transformed into a working device.

REFERENCES

[1] C. E. Shannon, "*A mathematical theory of communication*," Bell System Technical Journal, vol. 27, pp. 379–423 and 623–656, Jul. and Oct. 1948.

[2] C. Berrou, A. Glavieux, P. Thitimajshima, "*Near Shannon limit error-correcting coding and decoding: Turbo-Codes*," *Proc. IEEE Int. Conf. Commun.*, vol. 2, pp. 1064–1070, May 1993.

[3] R. G. Gallager, "*Low-density parity-check codes*," *IRE Trans. Inform. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[4] R. G. Gallager, *Low Density Parity Check Codes*. MIT Press, Cambridge, MA, 1963.

[5] D. J. C. MacKay, R. M. Neal, "*Near Shannon limit performance of low density parity check codes*," *Electronics Letters*, vol. 32, no. 18, Aug. 1996.

[6] D. J. C. MacKay, "*Good error correcting codes based on very sparse matrices*," *IEEE Trans. Inform. Theory*, vol. 45, pp. 399–431, Mar 1999.

[7] R. M. Tanner, "*A recursive approach to low complexity codes*," *IEEE Trans. Inform. Theory*, vol. 27, pp. 533–547, Sep. 1981.

[8] Y. Kou, S. Lin, M. Fossorier, "*Low-density parity-check codes based on finite geometries: A rediscovery and new results*," *IEEE Trans. Inform. Theory*, vol. 47, no. 7, pp. 2711–2736, Nov. 2001.

[9] T. J. Richardson, M. A. Shokrollahi, R. L. Urbanke, "*Design of capacity-approaching irregular low-density parity-check codes*," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.

[10] B. Vasic, W. E. Ryan, *CRC Handbook for Coding and Signal Processing for Recording Systems*, CRC Press, 2004.

[11] J. Hagenauer, "*Rate compatible punctured convolutional codes (RCPC codes) and their applications*," *IEEE Trans. Commun.*, vol. 36, no. 4, pp. 389–400, Apr. 1988.

[12] J. Hagenauer, E. Offer, L. Papke, "*Iterative decoding of binary block and convolutional codes*," *IEEE Trans. Commun.*, vol. 44, pp. 1261–1271, Oct. 1996.

[13] R. Y. Shao, S. Lin, M. P. C. Fossorier, "*Two simple stopping criteria for turbo decoding*," *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1117–1120, Aug. 1999.

[14] F.-M. Li, A.-Y. Wu, "*A new stopping criterion for efficient early termination in turbo decoder designs*," Proc. Int. Symp. on Intelligent Signal Processing and Communication Systems, Hong Kong, Dec. 2005.

[15] J. F. Cheng, "*Turbo decoder assisted frame error rate estimation*," Proc. 54th VTC Fall Conf., vol. 4, pp.2492–2496, Atlantic City, NJ, USA, 2001.

[16] C. Schurgers, L. Van der Perre, M. Engels, H. De Man, "*Adaptive turbo decoding for indoor wireless communication*," Proc. URSI Int. Symp. on Signals, Systems, and Electronics, Pisa, Italy, 1998.

[17] E. Rives, L. Joiner, "*Bit error rate locked loops using log-likelihood decoders*," *to appear in Proc. 2011 Eighth Int. Conf. on Information Technology: New Generations (ITNG)*, Las Vegas, NV, Apr. 2011.

[18] P. Hoeher, I. Land, U. Sorger, "*Log-likelihood values and Monte Carlo simulation–some fundamental results*," *in Proc. Int. Symp. on Turbo Codes & Rel. Topics*, pp. 43–46, Brest, France, Sep. 2000.

[19] X.-Y. Hu, E. Eleftheriou, D.-M. Arnold, A. Dholakia, "*Efficient implementations of the sum-product algorithm for decoding LDPC codes*," *Proc. IEEE Globecom*, pp. 1036–1036E, Nov. 2001.

[20] IEEE Std 802.16e-2005, *IEEE Standard for Local and metropolitan area networks, Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems*, *IEEE Computer Society*, Dec. 2005.

[21] IEEE Std 802.11n-2006, *IEEE Standard for Local and metropolitan networks, Part 11: Wireless LAN Medium Access Control (MAC), and Physical Layer (PHY) specifications: Enhancements for Higher Throughput*, *IEEE Computer Society*, Dec. 2006.

[22] ITU-T Recommendation G.821 (1996). *Error performance of an international digital connection operating at a bit rate below the primary rate and forming part of an integrated services digital network*, Geneva, 1996.

[23] ITU-T Recommendation G.113 (2001). *Transmission impairments due to speech processing*, Feb. 2001.

[24] F. M. Gardner, *Phaselock Techniques*, 3rd ed. Wiley-Interscience, Hoboken, NJ, 2005.

[25] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, Prentice-Hall, Upper Saddle River, NJ, 1995.

[26] S. Lin, D. J. Costello Jr., *Error Control Coding*, 2nd ed. Pearson Prentice Hall, Upper Saddle River, NJ, 2004.

[27] S. J. Orfanidis, *Introduction to Signal Processing*, Prentice-Hall, Upper Saddle River, NJ, 1996.

[28] D. E. Hocevar, *A reduced complexity decoder architecture via layered decoding of LDPC codes*, IEEE Workshop on Signal Processing Systems (SIPS), 2004, pp. 107–112.

[29] Y.-M. Chang, A. I. V. Casado, M.-C. F. Chang, R. D. Wesel, *Lower-complexity layered belief-propagation decoding of LDPC codes*, IEEE International Conference on Communications (ICC), 2008, pp. 1155–1160. Beijing.