Masterarbeit

The Tradeoff of Latency, Reliability, and Rate for Spinal Codes

Nikita Popov

347863

Institut für Telekommunikationssysteme
Technische Universität Berlin
10. August 2018

Erstgutachter: Prof. Dr. Giuseppe Caire
Zweitgutachter: Prof. Dr.-Ing. Rafael Schaefer
Bearbeitungszeitraum: 26. Februar 2018 bis 27. August 2018

Erklärung zur eigenständigen Anfertigung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Die selbstständige und eigenständige Anfertigung versichert an Eides statt:

Berlin, den 10. August 2018

.....

Abstract

Spinal codes [22] are a new class of "rateless" codes, which have the ability to generate an arbitrarily long sequence of symbols from a fixed-length message. These symbols can be transmitted until decoding succeeds at the receiver, allowing the code to automatically adapt to changing channel conditions and specific realizations of the noise process.

In this work, we investigate various practical aspects of the spinal coding construction, such as the handling of tail effects, choice of puncturing scheme and stopping criteria, as well as constellation mapping. We consider how geometric and probabilistic shaping can be used to reduce the gap to capacity, and what the correct criteria for the optimization of the constellation shape are. The performance of short spinal codes is also compared to theoretical limits on variable-length feedback coding.

Spinal codes essentially work by using a hash function to generate a random tree code, with each edge producing an infinite symbol sequence through a deterministic random number generator. Tree codes can be decoded using sequential decoding, mostly employed in the context of convolutional codes. We consider the application of different sequential decoding algorithms to spinal codes, with the goal of reducing the decoding complexity. We evaluate both established decoders such as the M-algorithm and the stack algorithm, as well as newer decoders proposed specifically for use with spinal coding.

Sequential decoding complexity is known to increase exponentially above a cutoff rate lower than capacity. However, there are claims that fixed-length spinal codes approach BSC and AWGN capacity with polynomial time decoding complexity [4]. We show that this is only possible due to a tradeoff between decoding complexity and error probability convergence behavior. Finally, we consider the theoretical limits of sequential decoding when applied to variable-length codes, in which case the cutoff rate can be surpassed.

Zusammenfassung

Spinal Codes [22] sind eine neue Klasse von “ratenfreien” Codes, welche eine beliebig lange Sequenz von Symbolen für eine Nachricht fester Größe generieren können. Diese Symbole können anschließend solange übertragen werden, bis der Empfänger die Nachricht erfolgreich dekodiert. Dies erlaubt eine automatische Anpassung an sich ändernde Kanalparameter, sowie an die konkrete Realisierung des Rauschens.

In dieser Arbeit untersuchen wir einige praktische Aspekte von Spinal Codes, wie etwa die Terminierung des Codes, die Wahl von Punktierung und Haltekriterien, sowie die verwendete Konstellation. Wir diskutieren wie geometrische und probabilistische Anpassung der Konstellation die Rate des Codes verbessern können, und welche Optimierungskriterien hierfür zu verwenden sind. Weiterhin vergleichen wir das Verhalten von kurzen Spinal Codes mit theoretischen Grenzen für die Kanalkodierung mit Codes von variabler Länge und Feedback.

Spinal Codes generieren einen zufälligen Kodierungsbaum mittels einer Hash-Funktion, wobei jede Kante mittels eines deterministischen Zufallszahlengenerators beliebig viele Symbole erzeugen kann. Baumartige Codes können sequentiell dekodiert werden, eine Methode welche derzeitig hauptsächlich auf Faltungscodes angewandt wird. Wir untersuchen die Anwendung von verschiedenen sequentiellen Dekodierungsalgorithmen auf Spinal Codes, mit dem Ziel die Dekodierungskomplexität zu reduzieren. Hierbei betrachten wir sowohl klassische Dekodierer, wie etwa den M-Algorithmus und Stapel-Algorithmus, als auch neuere Dekodierer welche speziell für die Verwendung mit Spinal Codes vorgeschlagen wurden.

Es ist bekannt, dass die Komplexität von sequentieller Dekodierung oberhalb einer Cutoff-Rate, welche geringer als die Kanalkapazität ist, exponentiell ansteigt. Jedoch gibt es auch Resultate, dass Spinal Codes mit konstanter Länge die Kapazität der BSC und AWGN Kanäle mit einem polynomiellen Dekodierer erreichen können [4]. Wir zeigen, dass dies nur durch eine Abwägung der Dekodierungskomplexität und dem Konvergenzverhalten der Fehlerwahrscheinlichkeit möglich ist. Zuletzt betrachten wir die theoretischen Grenzen von sequentieller Dekodierung von Codes mit variabler Länge, in welchem Fall die Cutoff-Rate überschritten werden kann.

Contents

Chapter 1

Introduction

Wireless communication systems operate over channels whose parameters are generally not known in advance and may vary over time due to various fading effects. Currently, unknown and varying channel conditions are typically handled by estimating channel parameters and choosing the used block code, rate and other coding parameters from a set of pre-determined options. Hybrid automatic repeat request (HARQ) schemes additionally employ an error-detecting code to request retransmission of erroneously decoded codewords.

An alternative is the use of rateless codes, which, for a given fixed-length message, can produce codewords of varying length, such that shorter codewords are always prefixes of longer codewords. This enables the transmitter to continue sending symbols until the decoder is able to successfully decode the message and signal the end of the transmission through some form of feedback channel. The advantage is that the code automatically adjusts not just to varying channel parameters, but also to specific realizations of the channel. This allows variable-length codes to operate at average rates close to channel capacity even for relatively small message sizes, while fixed-length block codes approach capacity only asymptotically.

In this work, we investigate *spinal codes*, a novel family of rateless codes due to Perry et al. [22]. Spinal codes are based on three primitives: Sequential application of a *hash function* to portions of the message induces a random coding tree, for each edge of which a *pseudo-random number generator* (PRNG) generates an infinite stream of output bits, which is converted to channel symbols by a *constellation mapping function*. As such, the spinal coding construction has some similarity to convolution codes, another type of tree code. The two main differences are that spinal codes are based on a hash function rather than a convolution, and that they don't generate a fixed number of symbols for each tree edge.

The use of a non-linear hash function with very large internal state implies that spinal codes operate over a genuine tree and do not exhibit a Trellis structure (or rather, the Trellis structure is so large as to not be meaningful). This also makes the use of maximum likelihood decoding methods like the Viterbi algorithm impractical. Instead sequential decoding is employed, which refers to a category of heuristic tree search algorithms, which use a metric to guide a partial exploration of the coding tree. Sequential decoding is best known for its application to the decoding of convolutional codes with large constraint lengths.

The remainder of this work is structured as follows. In the remainder of this chapter we will introduce the channel model for variable-length codes with feedback, as well as relevant fundamental coding limits, and discuss related work.

In chapter 2 we will introduce the spinal code construction in detail, as well as investigate a number of aspects of their practical realization. In particular we discuss the choice of hash function and random number generator, different ways of handling tail effects, as well as the choice of transmission and puncturing scheme. Particular focus is placed on the used constellation mapping: Geometric and probabilistic shaping can be used to reduce the gap to capacity for the AWGN channel. However, the constellation optimization needs to be carried out against different criteria than for fixed-length codes.

Chapter 3 introduces the sequential decoding method, and compares the performance of different algorithms as applied to spinal codes. This includes both classics like the M-algorithm and stack algorithm, as well as decoders that have been proposed specifically in the context of spinal coding. The goal is to identify the decoding algorithm which requires the lowest computational effort to achieve a certain rate.

Chapter 4 considers theoretical results regarding decoding complexity and achievable rate. Sequential decoding complexity is known to increase exponentially above a cutoff rate lower than capacity [14] [2]. On the other hand, there are claims that fixed-length spinal codes approach capacity for the BSC and AWGN channels with polynomial time decoding complexity [4]. We discuss how these results may be reconciled, and further investigate the achievable rate of sequentially decoded codes in the variable-length coding regime.

Finally, chapter 5 summarizes and concludes. Appendix A lists used simulation parameters.

1.1 Channel Coding Model and Fundamental Limits

When working with variable-length codes with feedback there are a number of possible channel models, as well as definitions for common coding parameters. We will lean heavily on the terminology used by Polyanskiy et al. [24] here. First, a non-anticipatory channel is given by input and output alphabets $\mathcal{X}$ and $\mathcal{Y}$, and a sequence of conditional probability kernels $P_{Y_i|X^iY^{i-1}}$. We will generally limit our considerations to memoryless channels with

$$P_{X_i|X^iY^{i-1}} = P_{Y_i|X_i} = P_{Y_1|X_1} \quad \forall i \geq 1$$

and finite $\mathcal{X}$. If $\mathcal{Y}$ is also finite, this is known as a discrete memoryless channel (DMC). Polyanskiy now gives the following two definitions for variable-length coding with feedback:

Definition 1. An (l, M, ϵ) *variable-length feedback (VLF)* code, where l is a positive real, M is a positive integer and $0 \leq \epsilon \leq 1$, is defined by:

1. A space $\mathcal{U}$ with $|\mathcal{U}| \leq 3$ and a probability distribution P_U on it, defining a random variable U which is revealed to both transmitter and receiver before the start of transmission.

2. A sequence of encoders $f_n : \mathcal{U} \times \{1, \ldots, M\} \times \mathcal{Y}^{n-1} \to \mathcal{X}$, defining channel inputs $X_n = f_n(U, W, Y^{n-1})$, where $W \in \{1, \ldots, M\}$ is the equiprobable message.

3. A sequence of decoders $g_n : \mathcal{U} \times \mathcal{Y}^n \to \{1, \ldots, M\}$ providing the best estimate of W at time n.

4. A non-negative integer-valued random variable τ, a stopping time of the filtration $\mathscr{G}_n = \sigma\{U, Y_1, \ldots, Y_n\}$, which satisfies $\mathbb{E}[\tau] \leq l$.

The final decision $\hat{W}$ is computed at the time instant τ by $\hat{W} = g_\tau(U, Y^\tau)$ and must satisfy $\mathbb{P}[\hat{W} \neq W] \leq \epsilon$.

Definition 2. An (l, M, ϵ) *variable-length feedback code with termination (VLFT)*, where l is positive real, M is a positive integer and $0 \leq \epsilon \leq 1$, is defined similarly to VLF codes with an exception that condition 4 in Definition 1 is replaced by:

4'. A non-negative integer-valued random variable τ, a stopping time of the filtration $\mathscr{G}_n = \sigma\{W, U, Y_1, \ldots, Y_n\}$, which satisfies $\mathbb{E}[\tau] \leq l$.

The fundamental limit of channel coding for these two types of codes may be defined by:

$$M_f^*(l, \epsilon) = \max\{M : \exists(l, M, \epsilon)\text{–VLF code}\}$$
$$M_t^*(l, \epsilon) = \max\{M : \exists(l, M, \epsilon)\text{–VLFT code}\}$$

The random variable U in these definitions is used to simplify randomization based proofs, but is ultimately not very relevant for our considerations.

Under the VLF coding paradigm, arbitrary feedback may be transmitted noiselessly, but the decision to stop transmission is made using only the channel outputs Y_i observed at the receiver, while VLFT codes may also use the actual message W. VLFT codes derive their name from being equivalent to VLF codes over a modified channel, which has an additional, perfectly reliable, single-use termination symbol. Polyanskiy's motivation for this model is that communication control, including termination, is often handled through separate layers and protocols, which may have characteristics very different from the payload channel.

While these definitions are very general, spinal codes fall into one of two special cases: *Stop-feedback codes* are VLF codes where the encoder functions satisfy

$$f_n(U, W, Y^{n-1}) = f_n(U, W), \tag{1.1}$$

that is feedback is only used to stop the transmission when the decoder is ready to decode. *Fixed-to-variable codes* (FV codes) are VLFT codes which satisfy (1.1) with stopping time

$$\tau = \inf\{n \geq 1 : g_n(U, Y^n) = W\},$$

that is they stop as soon as decoding is performed correctly (as determined by a friendly genie). As such, these are zero-error codes.

Polyanskiy shows the following bounds for VLF and VLFT codes over a DMC with capacity C and error probability $0 < \epsilon < 1$:

$$\frac{lC}{1-\epsilon} - \log l + O(1) \leq \log M_f^*(l, \epsilon) \leq \frac{lC}{1-\epsilon} + O(1)$$
$$\log M_f^*(l, \epsilon) \leq \log M_t^*(l, \epsilon) \leq \frac{lC + \log l}{1-\epsilon} + O(1)$$

The achievability bounds also hold restricted to stop-feedback codes. These results should be considered in comparison to a similar bound for codes of fixed blocklength n without feedback [23], given by

$$\log M^*(n,\epsilon) = nC - \sqrt{nV}Q^{-1}(\epsilon) + O(\log n),$$

where V is the channel dispersion and $Q(x) = \int_x^\infty \frac{e^{-y^2}}{\sqrt{2\pi}} dy$. In this case the backoff from ϵ-capacity is dominated by the $1/\sqrt{n}$ term. Variable-length feedback codes eliminate this penalty and allow approaching capacity at much smaller average blocklengths.

For zero-error VLFT codes over a DMC with capacity C, the bounds

$$lC + O(1) \leq \log M_t^*(l,0) \leq lC + \log l + O(1)$$

are shown, with the achievability bounds also holding when restricted to FV codes. As such, VLFT codes can achieve capacity at very short block lengths. In fact, the somewhat artificial availability of a noiseless termination symbol allows capacity to be exceeded at very short block lengths.

Our simulations for spinal codes will mostly follow the zero-error VLFT with stop-feedback model (FV codes). Our default message size is $\log M = 256$, which is sufficiently large for capacity to be achievable under this model. As such, the ordinary channel capacity is a reasonable comparison point for our purposes. In section 2.5 we will consider how spinal codes can be applied to the more realistic VLF stop-feedback model.

As a final note, when we show "average rate" results, we are referring to the quantity $R = \mathbb{E}[(\log M)/\tau]$ rather than $R = (\log M)/\mathbb{E}[\tau]$. Some discussion on the difference between these definitions can be found in [33]. For the case where we are considering a single channel over which multiple transmissions are performed the latter quantity is more meaningful, as it provides the expected ratio of bits to channel uses across many transmitted messages. We only use the former definition here, because we become aware of this distinction too late to adjust all results. By Jensen's inequality we know that $\mathbb{E}[(\log M)/\tau] \geq (\log M)/\mathbb{E}[\tau]$. In our particular setup, the difference between the two quantities tends to be less than one per percent.

1.2 Related Work

Spinal codes have been first introduced by Perry et al. [21] [22], with additional achievable rate proofs provided in [4]. Since then spinal codes have been investigated in a number of directions: Different decoders with lower complexity than the M-algorithm have been proposed, such as the forward stack decoder [41], sliding feedback decoder [39] and sliding window decoder [40]. Transmission protocols that take into account the latency of feedback in realistic communication systems [12], as well as unreliable feedback [17] have been developed. The application of spinal codes to decode-and-forward relay channels [44] [42], as well as massive MIMO systems [35] has been investigated. Variations of the spinal coding construction, such as two-way spinal codes [43] and spinal codes with unequal error protection [45] were also suggested.

Spinal codes make use of sequential decoding [38], which has been explored extensively for its applications to convolutional coding. Examples of sequential decoding algorithms are the Fano algorithm [8], the stack algorithm [16] [46] and the adaptive effort algorithm [31]. An overall survey is available at [1]. Furthermore there are many theoretical results on the complexity of sequential decoding, in particular the exponential complexity above the cutoff rate [14] [2] and boundedness below the cutoff rate [7] [27] [15].

Of course, spinal codes are not the first rateless coding scheme. LT codes [18] and the Raptor [30] based on them approach capacity for the erasure channel and can be extended to the BSC and AWGN channels [20]. Another rateless code based on a layering construction is Strider [10]. An alternative to intrinsically rateless codes are incremental redundancy HARQ schemes based on punctured blockcodes [26] [28].

Chapter 2

Spinal Codes

Fundamentally, spinal codes are a type of tree code, with the significant difference that each tree branch generates an infinite stream, rather than a fixed number of output symbols. A finite prefix of each output stream is transmitted, with the prefix length increasing until decoding can succeed at the receiver.

This coding scheme is based on three primitives, shared between transmitter and receiver: A hash function, which determines the internal state at each tree node, a deterministic pseudo-random number generator (RNG), which generates an infinite stream of bits from the internal state, and finally a constellation mapper, which converts the bits into output symbols for a given channel.

The hash function

$$h : \{0,1\}^\nu \times \{0,1\}^k \to \{0,1\}^\nu$$

takes a ν-bit state and a k-bit message fragment. Given an n-bit message $\mathbf{m} = m_1 \dots m_n$, we denote by $\mathbf{m}_i = m_{ki+1} \dots m_{k(i+1)}$ the i-th group of k consecutive bits of the message. The internal states are then given by

$$\mathbf{s}_i = h(\mathbf{s}_{i-1}, \mathbf{m}_i), \quad i = 1, \dots, n/k$$

where $\mathbf{s}_0$ is an arbitrary initial state known to both the encoder and decoder. The states $\mathbf{s}_i$, also called *spines*, are used to seed a random number generator

$$\mathrm{RNG} : \{0,1\}^\nu \times \mathbb{N} \to \{0,1\}^c,$$

which produces an infinite sequence of c-bit numbers, indexed by a natural number. Finally, the mapper

$$\phi : \{0,1\}^c \to \mathcal{X}$$

converts each c-bit number into an output symbol compatible with the given channel. The entire encoding procedure is illustrated in Figure 2.1, where

$$x_{i,j} = \phi(\mathrm{RNG}(\mathbf{s}_i, j)), \quad i = 1, \dots, n/k, \quad j \in \mathbb{N}$$

denotes the j-th output symbol generated from the i-th spine. The transmitted message can now be constructed as a non-repeating sequence of x_{ij}s. The simplest scheme is to

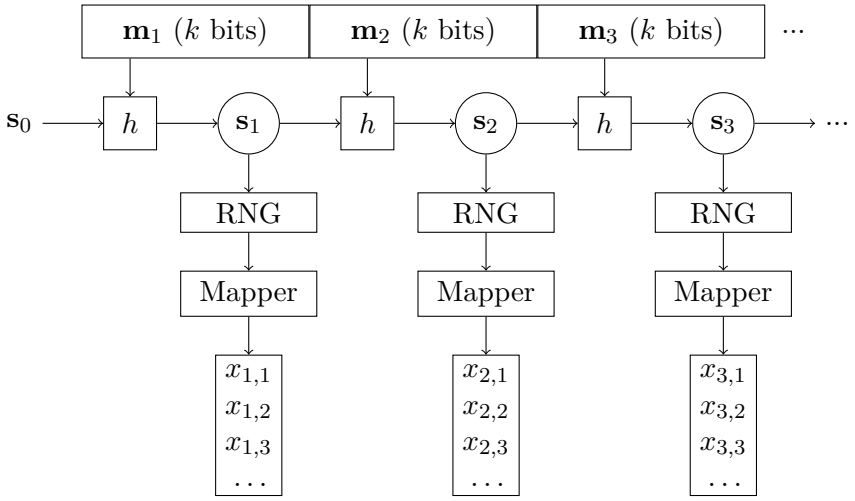


Figure 2.1: Encoding process for spinal codes. Spines $\mathbf{s}_i$ are generated through consecutive applications of a hash function to message fragments $\mathbf{m}_i$. Output symbols $x_{i,j}$ are generated from spines using an RNG and constellation mapper.

subdivide transmission into passes, where each pass transmits one symbol from each spine. In this case the transmitted message is

$$\mathbf{x} = \underbrace{x_{1,1}x_{2,1}\ldots x_{n/k,1}}_{\text{Pass 1}}\underbrace{x_{1,2}x_{2,2}\ldots x_{n/k,2}}_{\text{Pass 2}}\ldots \in \mathcal{X}^{\omega},$$

with transmission continuing until either the receiver decodes successfully, or some other stopping criterion is reached.

In the following various practical considerations of the general spinal coding scheme will be discussed.

2.1 Hash Function and Random Number Generator

As written above, the RNG produces a sequence of c-bit numbers. Practical RNGs tend to generate outputs of either 32 or 64 bits, which can be converted into a c-bit sequence through use of a shift register. While the hash and RNG are conceptually separate, the hash function can be used as the basis for the RNG implementation. One possibility is to use

$$h(\mathbf{s},0),\ \ h(\mathbf{s},1),\ \ h(\mathbf{s},2),\ \ \ldots$$

as the underlying bit stream for the RNG. Another option is the recursive structure

$$h(\mathbf{s},0),\ \ h(h(\mathbf{s},0),0),\ \ h(h(h(\mathbf{s},0),0),0),\ \ \ldots$$

For a strong hash function, both variants are equally good. While the RNG can be based on the hash function, this is not necessarily desirable. Nowadays many PRNGs with both high

performance and good statistical properties are known. Examples are SplitMix64 (using 64-bit internal state) and xoroshiro128+ (using 128-bit internal state).

For the choice of hash function there are two general possibilities: Cryptographic hash functions are guaranteed to have good avalanching properties, as well as preimage and collision resistance, but tend to be very slow. Non-cryptographic hashes (typically intended for hash-based lookup structures) can have many different performance and avalanching characteristics, but generally do not offer preimage or collision resistance.

Assuming random messages and a high quality RNG, even very weak hash functions can in principle be used for spinal coding. As an extreme example, consider the DJBX33A hash function given by

$$s_i = h(s_{i-1}, m_i) = 33 \cdot s_{i-1} + m_i \pmod{2^{64}}$$

for some arbitrary non-zero s_0. For message fragment sizes $k \leq 5$ this hash function will perform identically to a strong cryptographic hash, while for $k \geq 6$ an average rate reduction will be observed. Given the structure of the function, this behavior is to be expected: For $k \leq 5$ we have $2^k < 33$, so that collisions can only be introduced through the modular arithmetic. For $k \geq 6$ on the other hand we have $2^k > 33$ and collisions can be caused trivially, for example (m_1, m_2) and $(m_1 + 1, m_2 - 33)$ will collide for any $0 \leq m_1 \leq 32$ and $33 \leq m_2 < 2^k$.

This sheds some light on the practical requirements on the hash function: While both cases are subject to collisions, the distinction is in the distance between the first differing spine and the spine where the collision occurs. In the case of modular collisions this distance will be large and it is unlikely that both messages are still considered by the (sequential) decoder at the time of collision. In the latter case, the distance is small and it is likely that both messages will still be candidates and the decoder will need additional symbols to reliably distinguish them.

Finally, consideration should also be given to the case where the messages are not random, but controlled by an adversary aiming to disproportionally increase decoding complexity through the exploitation of hash collisions. To achieve this it is not sufficient to find arbitrary collisions, instead ones with low collision distance in the above sense are necessary. Even then, such collisions result only in a minor complexity increase, as the messages can still be distinguished based on spines prior to the collision. A large complexity increase would only be possible if collisions in multiple consecutive spines can be constructed, i.e. there exists (m_i, m_{i+1}) and (m'_i, m'_{i+1}), with $m_i \neq m'_i$, $m_{i+1} \neq m'_{i+1}$ and $s_i = s'_i$, $s_{i+1} = s'_{i+1}$. Even for very weak functions like the one discussed above, this is generally not possible.

Based on the above discussion, it does not appear that adversarially chosen messages pose a viable means for a complexity attack, and consequently the use of a cryptographic hash (which would prevent collision-based attacks by construction) is not advantageous. If the use of a cryptographic hash is combined with a sufficiently large secret initial state shared between transmitter and receiver, then the message will be encrypted as part of the coding scheme. It is not clear whether this has any useful applications, but is interesting in the context of the recent trend of pushing encryption to lower levels of the network stack (such as the QUIC protocol [13] providing transport-level encryption).

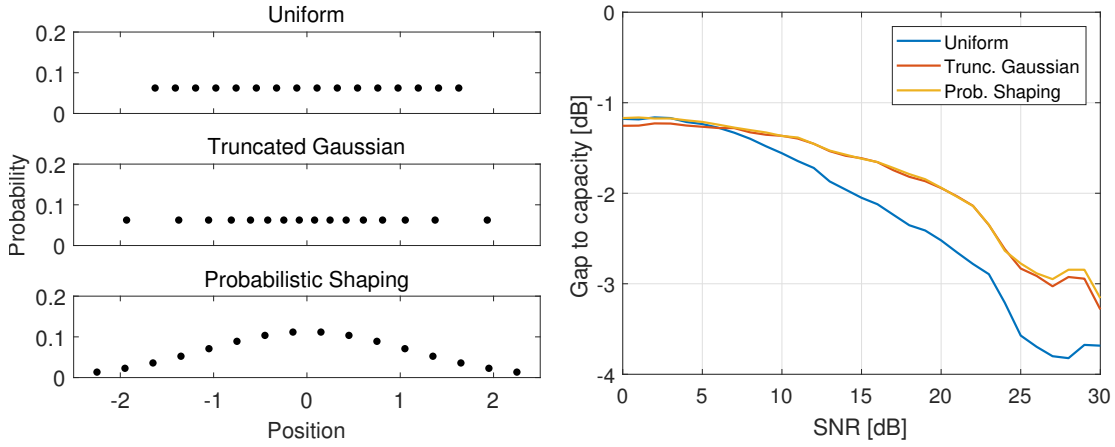


Figure 2.2: Left: Examples for uniform, truncated Gaussian and probabilistically shaped constellations of size 2^4. Right: Gap to capacity[1]of spinal codes using different constellations of size 2^6. Truncated Gaussian uses $\beta = 2$. Probabilistic shaping is optimized for $R_{\mathrm{VLFT}}(L = 16)$.

2.2 Constellation Mapping

The spinal coding scheme produces an infinite sequence of bits for each spine, which can be subdivided and mapped to channel symbols in an arbitrary way. For the BSC the bits can be used directly. For the AWGN channel groups of c' bits are mapped to channel symbols through a constellation mapping function $\phi : \{0,1\}^{c'} \to \mathcal{X}$, where $\mathcal{X}$ is a constellation of size $|\mathcal{X}| = 2^c$ with probability mass function $Q : \mathcal{X} \to \mathbb{R}$ denoting the probability with which a certain constellation symbol should be generated. We assume that the channel is subject to an average power constraint

$$\mathbb{E}[|X|^2] = \sum_{x \in \mathcal{X}} Q(x)|x|^2 \leq P.$$

We will now consider a number of different constellations using the ASK constellation

$$\mathcal{U} = \{\pm 1, \pm 3, \ldots, \pm(2^c - 1)\}$$

as the basis. The first and simplest is the *uniform* constellation, using equidistant and equiprobable points, that is $\mathcal{X} = \Delta_{\min} \cdot \mathcal{U}$ and $Q(x) = \frac{1}{2^c}$, where

$$\Delta_{\min} = \sqrt{\frac{P}{\mathbb{E}[|U|^2]}} = \sqrt{\frac{3P}{2^{2c} - 1}}$$

is a scaling factor that satisfies the power constraint with equality, based on the variance of the discrete uniform distribution. Taking into account that the capacity-achieving input

[1]The gap to capacity specifies how much more noise a capacity-achieving code could handle at the same rate R. For the AWGN it is given by $\mathrm{SNR}/(2^{2R} - 1)$, typically in dB.

distribution for the AWGN channel is Gaussian, it is not surprising that the uniform constellation leaves a significant gap to capacity. There are two general methods to remedy this problem: The first is geometric shaping, which uses a non-uniform spacing of constellation points while keeping a uniform probability distribution. The second is probabilistic shaping, which retains equidistant points, but uses a non-uniform pmf. Figure 2.2 illustrates the different constellation types and shows their impact on spinal coding.

For geometric shaping, a candidate is the *truncated Gaussian* constellation given by

$$x_i \propto \Phi^{-1}\left(\gamma + (1-2\gamma)\frac{i+\frac{1}{2}}{2^c}\right), \quad \gamma = \Phi(-\beta),$$

where Φ is the CDF of the standard normal distribution and β determines at which point the distribution is truncated. For $c \to \infty$ and $\beta \to \infty$ the proportionality factor used above tends to $\sqrt{P}$, though for practical values of c and β an explicit normalization is necessary to satisfy the power constraint.

For probabilistic shaping we combine equidistant points $\mathcal{X}_\Delta = \Delta \cdot \mathcal{U}$ together with a Maxwell-Boltzmann distribution given by

$$Q(x) = A_\nu e^{-\nu|x|^2}, \quad A_\nu = \frac{1}{\sum_{x\in\mathcal{X}_\Delta} e^{-\nu|x|^2}},$$

where for any $\Delta \geq \Delta_{\min}$ there exists a unique ν such that the power constraint is satisfied with equality. As this $\nu(\Delta)$ is strictly monotonically increasing, the value can be determined efficiently through binary search. For $\Delta = \Delta_{\min}$ we have $\nu = 0$, recovering the uniform constellation.

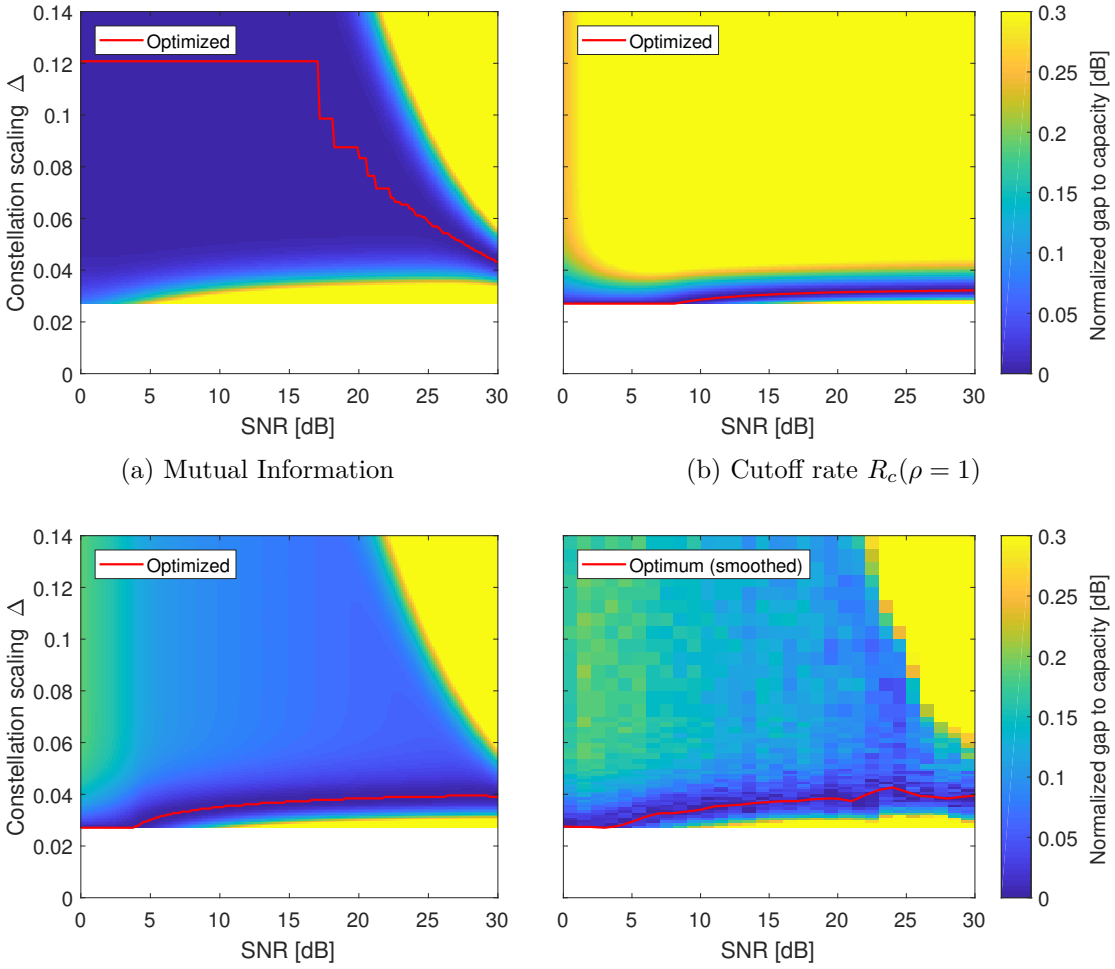
It is now possible to optimize the parameter Δ, though it is not entirely obvious what it should be optimized for. Normally, as suggested by Böcherer in [5], we should optimize for the mutual information, that is

$$\Delta_I = \max_\Delta I(X_\Delta; X_\Delta + Z) = \max_\Delta H(X_\Delta + Z),$$

where $Z \sim \mathcal{N}(0, \sigma^2)$ is the AWNG noise variable. As the mutual information is unimodal in Δ, the optimum can be efficiently found using golden section search.

Figure 2.3 (a) illustrates how the mutual information varies with Δ. The plot shows the gap to capacity, normalized by subtracting the smallest gap for each SNR. That is, it shows how much worse a certain choice of Δ is than the optimum. We can see that especially at lower SNRs a wide Δ range produces very similar results, to the point that the combination of golden section search and numerical integration makes the optimization procedure numerically unstable. This also means that only few different values of Δ are necessary to cover the whole SNR range with good quality.

Unfortunately, when comparing these results with the actual behavior of spinal codes shown in Figure 2.3 (d), we see that the mutual information is not a good predicator for the Δ dependence of the average rate. The Δ_I chosen by optimization for maximum mutual information will result in rates that are significantly better than the uniform constellation for high SNR, but will actually perform worse at low SNR.



(a) Mutual Information

(b) Cutoff rate $R_c(\rho = 1)$

(c) VLFT rate prediction $R_{\mathrm{VLFT}}(L = 16)$

(d) Average rate in spinal code simulation

Figure 2.3: Impact of spacing factor Δ on the performance of probabilistically shaped constellations of size 2^6. All plots show a normalized gap to capacity, where the smallest gap for each SNR has been subtracted. That is, they show how much worse a certain Δ is than the best value at this SNR. Additionally the best values are highlighted in red. The results show that mutual information (a) is a very bad predictor for the actual simulation results (d). Using the cutoff rate $R_c(\rho = 1)$ (b) shows a better, but still incorrect optimization behavior. The VLFT rate $R_{\mathrm{VLFT}}(L = 16)$ is a good predictor for both the optimal Δ and the behavior of the normalized gap to capacity.

An alternative optimization target is the computational cutoff rate $R_c(\rho)$, which is the rate at which the ρ-th moment of the decoding complexity of a sequential decoder starts to increase exponentially. As discussed in section 4.1, it is given by

$$R_c(\rho) = \frac{E_0(\rho)}{\rho} = -\frac{1}{\rho} \log \sum_{y \in \mathcal{Y}} \left[\sum_{x \in \mathcal{X}} Q(x) P(y \mid x)^{\frac{1}{1+\rho}} \right]^{1+\rho},$$

where $E_0(\rho)$ is the Gallager error function and $P(\cdot|\cdot)$ denotes the channel transition probability. Usually the cutoff rate for $\rho = 1$ is considered, which is the rate at which the average decoding complexity diverges. As spinal codes employ sequential decoding, $R_c(\rho = 1)$ is a meaningful optimization target.

Figure 2.3 (b) shows the dependence of $R_0(\rho = 1)$ on Δ. The prediction for the optimum Δ is better in this case, in that the uniform constellation should be used at low SNR and a small Δ at higher SNRs. However, the predicted values are too low, and additionally $R_c(1)$ reacts very sensitively to changes in Δ, which is not the case in the spinal code simulation.

In section 4.3 we will consider the performance of sequential decoding for variable length codes under the VLFT channel model and arrive at the expression

$$R_{\mathrm{VLFT}}(L) = \int_0^\infty (\ln L) e^{-(\ln L)\rho} R_c(\rho) d\rho$$

depending on a complexity parameter L. Figure 2.3 (c) shows the results for $R_{\mathrm{VLFT}}(L = 16)$. In this case the predicted optimum matches that of the simulation (discounting puncturing artifacts at high SNR), and the impact of deviations in Δ on the rate is predicated reasonably well.

Finally, the mapping from RNG outputs to constellation symbols should be discussed. For uniform or geometrically shaped constellations, we can choose $c' = c$ and map the $2^{c'}$ different RNG outputs to the 2^c constellation symbols bijectively. As we are not interested in recovering the RNG bit stream at the decoder, but rather only need to distinguish the symbol streams for different $\mathbf{s}_i$, this mapping can be performed in an arbitrary fashion and does not benefit from using schemes such as binary reflected Gray codes.

When using probabilistic shaping, we must generate output symbols according to the pmf $Q(\cdot)$. While this can be challenging when applying probabilistic shaping to other coding schemes, the flexibility of constellation mapping for spinal codes comes to our advantage here: Denoting by $F_i = \sum_{j \le i} Q(x_j)$ the cumulative distribution, we define

$$J_i = \left\{ j \in \{1, \ldots, 2^{c'}\} \,\middle|\, F_{i-1} < \frac{j - \frac{1}{2}}{2^{c'}} \le F_i \right\}$$

and map all RNG outputs contained in J_i to symbol x_i. For a sufficiently large $c' > c$, this allows approximating the output distribution to arbitrary precision.

Going back to Figure 2.2 (right) we can conclude that geometric shaping and probabilistic shaping produce essentially the same results, and both perform significantly better than uniform at high SNR. One advantage of probabilistic shaping is that equidistant constellations require less resolution during analog-to-digital conversion.

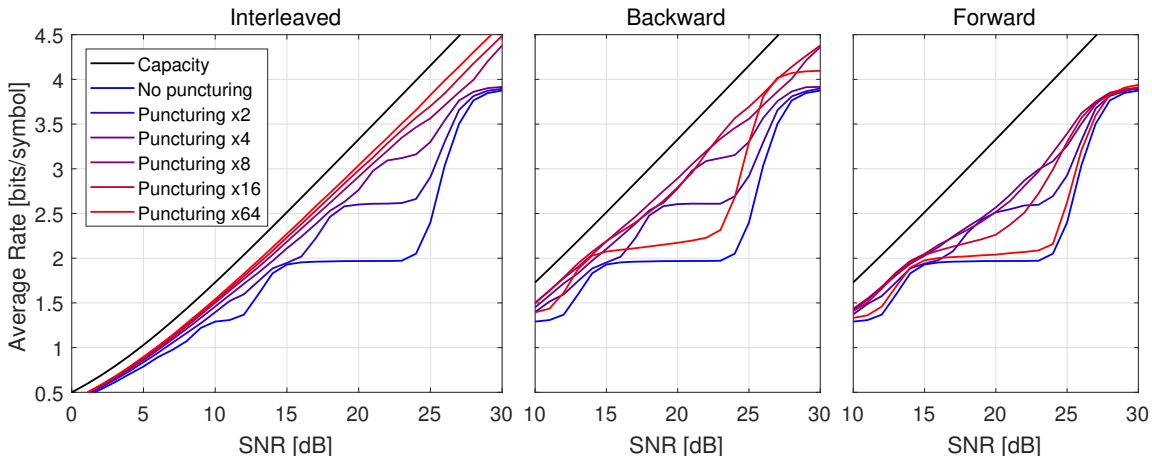


Figure 2.4: Comparison of average rate of $n = 256, k = 4$ spinal code on the AWGN channel, using different puncturing factors ($p = 1, 2, 4, 8, 16, 64$) and patterns (interleaved, backward, forward). Puncturing factor $p = 64$ is the maximum for our parameters, resulting in a decode attempt after every symbol.

2.3 Transmission Sequence and Puncturing

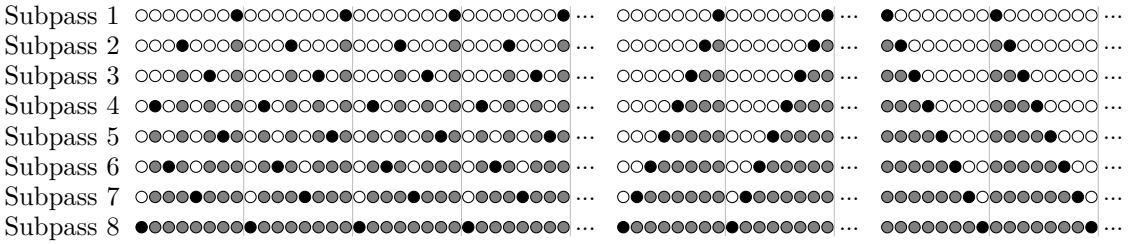
The symbols x_{ij} produced by the spinal code need to be transmitted in some order and decoding attempts have to be performed at defined points. The simplest scheme is to transmit the codeword

$$\mathbf{x} = \underbrace{x_{1,1}x_{2,1}\dots x_{n/k,1}}_{\text{Pass 1}}\underbrace{x_{1,2}x_{2,2}\dots x_{n/k,2}}_{\text{Pass 2}}\dots \in \mathcal{X}^{\omega},$$

and perform one decoding attempt after each complete pass has been received. Denoting by L the number of passes until decoding succeeds, the rate (of that particular transmission) will be $R = k/L$, with the maximum possible rate being $R_{max} = k$ at $L = 1$. For a typical value of $k = 4$ and the AWGN channel in high-SNR regime, this will already fall below capacity.

More importantly, at high rates $R = k/L$ for integer L will leave large gaps between allowable rates, with the first values for $k = 4$ being $R = 4, 2, 1, 0.5, 0.25, \dots$. While at a certain noise level reliable decoding at rate $R = 3.5$ may be possible, the transmission will most likely use $R = 2$ as the next lowest available rate (though $R = 4$ is also possible if the channel conditions are favorable). The lowest curve in Figure 2.4 (left) shows the detrimental effect of this rate quantization, even at lower SNR levels.

To resolve this problem, we can employ puncturing by a factor of p. Unlike puncturing for convolutional codes, this does not refer to dropping some output symbols entirely. Rather, each pass is subdivided into p subpasses containing $\frac{n}{kp}$ symbols each and a decoding attempt will be performed after each subpass. Specifically, we use the following scheme:



(a) Interleaved (b) Backward (c) Forward

Figure 2.5: Different puncturing patterns at puncturing factor $p = 8$. Black circles denote symbols sent in each subpass, while grey circles are already received symbols. A decoding attempt will be performed after each subpass.

Given a permutation π of $1, \ldots, p$ we transmit the sequence

$$\mathbf{x} = \underbrace{x_{\pi_1,1} x_{\pi_1+p,1} x_{\pi_1+2p,1} \ldots x_{\pi_1+\frac{n}{k}-p,1}}_{\text{Subpass 1}} \underbrace{x_{\pi_2,1} x_{\pi_2+p,1} x_{\pi_2+2p,1} \ldots x_{\pi_2+\frac{n}{k}-p,1}}_{\text{Subpass 2}} \ldots,$$

that is, first we transmit all symbols at $i \equiv \pi_1 \pmod{p}$, then at $i \equiv \pi_2 \pmod{p}$, and so on. We have tested three different puncturing patterns illustrated in 2.5:

- Forward: This is the identity permutation $\pi = (1, 2, \ldots, p)$.
- Backward: This is the reversed identity permutation $\pi = (p, p-1, \ldots, 1)$.
- Interleaved: Starting with $\pi^{(1)} = (1)$, the permutation is constructed recursively by

$$\pi_i^{(p)} = \pi_{\lfloor i/2 \rfloor}^{(p/2)} + \begin{cases} \frac{p}{2} & \text{if } i \text{ odd} \\ 0 & \text{otherwise} \end{cases}.$$

  For $p = 8$ this yields $\pi = (8, 4, 6, 2, 7, 3, 5, 1)$.

Figure 2.4 compares the average rate of spinal codes for the AWGN channel for different puncturing factors and patterns. Clearly puncturing is necessary to achieve good rates at high SNR. $p = 8$ is the lowest factor (for the shown SNR range) which does not show clear plateau effects due to the rate quantization. Even then increasing the puncturing factor until a decode is performed after every received symbol still yields significant rate increases at high SNR. On the other hand, every doubling of the puncturing factor will also approximately double the decoding effort, leading to diminishing returns. For the depicted SNR range $p = 8$ is a reasonable choice, while a lower factor would be preferable for a smaller SNR range.

The comparison of the different puncturing patterns shows that the interleaved pattern performs best, as is to be expected. Interestingly the backward pattern shows identical behavior up to $p = 8$, past which performance degrades, presumably because the decoder is unable to proceed past the large gap of symbols that have not been sent yet.

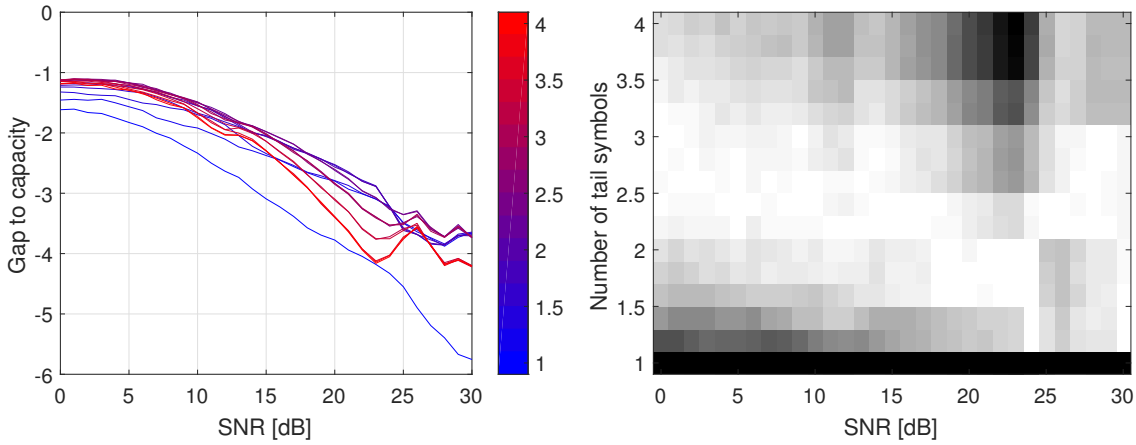


Figure 2.6: Gap to capacity for AWGN channel for different numbers of tail symbols, represented as a line graph in the left figure and a heatmap in the right. In the right figure the gap to capacity has been normalized for each SNR. White denotes the minimum gap and black the maximum gap.

The forward pattern performs worst, especially at high puncturing levels. In fact, at the highest puncturing level, which in this case corresponds to simply transmitting all symbols in the original order and attempting decoding after each, the performance is nearly the same as performing no puncturing at all. The reason for this is the special significance of the last spine to the decoding process. As will be discussed in the next section, this is the spine with the most uncertainty and thus the bottleneck of the decoding process.

2.4 Tail Symbols

The nature of spinal codes as tree codes results in an uneven distribution of reliability across the message. Because the spine is constructed through successive hashing of message fragments, this means that a message fragment $\mathbf{m}_k$ influences all symbols x_{ij} where $i \geq k$. In particular, all transmitted symbols have a dependency on the first message fragment $\mathbf{m}_1$, while the last message fragment $\mathbf{m}_{n/k}$ only influences the symbols of the last spine.

An analogous problem exists for convolutional codes. The first few bits have higher reliability because they start from a known initial state. The final bits have lower reliability because no subsequent bits are available to correct them. For convolution codes this problem is solved through a form of termination: Zero termination appends additional message bits to ensure that the final state is zero. For non-recursive codes, this is achieved by appending zero bits to the message. An alternative is the use of tail-biting, which defines that the start and end state must match (effectively treating the message as periodic), thus removing the tail effect entirely, at the cost of more expensive decoding.

The preimage resistance of good hash functions implies that for spinal codes it is not possible to force the final state to be zero or enforce matching start and end states. However, the general notion of termination is still applicable and there are at least two methods to

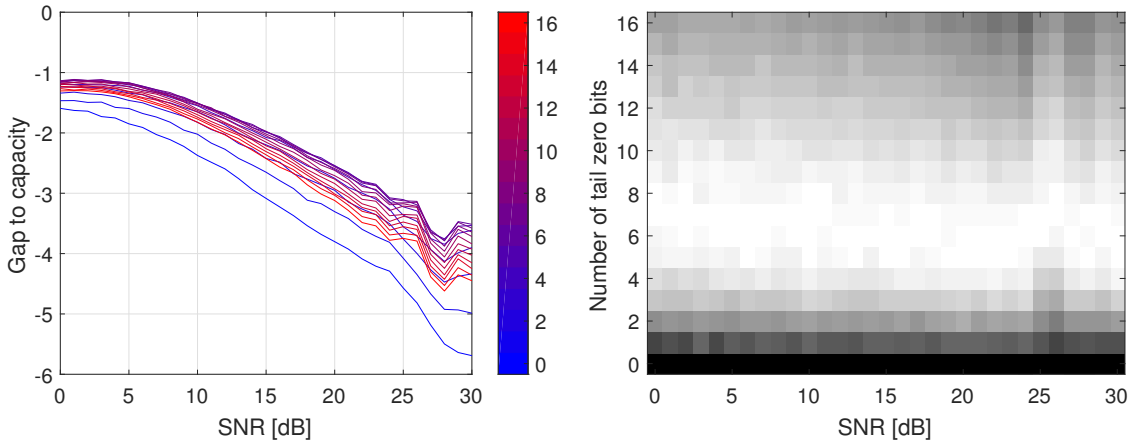


Figure 2.7: Gap to capacity for AWGN channel for different number of trailing zero bits, represented as a line graph in the left figure and a heatmap in the right. In the right figure the gap to capacity has been normalized for each SNR. White denotes the minimum gap and black the maximum gap.

realize it:

First, we can require that the last t bits of the message are zero. This is similar to zero termination for convolutional codes, but does not cause the end state to be zero or any defined value. Instead it has the effect of both removing information-carrying bits from the less reliable tail of the message, and providing additional prior information for the decoder.

Second, it is possible to send T times more symbols for the last spine than for others, counteracting the lower per-symbol reliability by sending more symbols.

We have evaluated both of these approaches. Figure 2.6 shows the impact of sending more symbols for the last spines, for factors between $T = 1$ and $T = 4$. A fractional factor T is to be understood as an average value. For example, for $T = 1.5$ we will send two tail symbols on the first pass, one on the second pass, two on the third, one on the 4th, and so on.

Figure 2.6 left shows the AWGN gap to capacity for different T. We can see that not sending additional tail symbols ($T = 1$) comes with a large performance penalty. Figure 2.6 right illustrates which factors T are best for different SNRs, with light regions denoting small gaps to capacity and dark regions denoting large gaps. While the optimum is SNR dependent, values between $T = 2.0$ and 2.5 tend to be good.

The concept of tail symbol repetition can be extended by distributing the additional symbols across multiple trailing spines, rather than only the last one. However, in our experiments doing so did not result in an improvement. Additional tail symbols benefit decoding not by making the decoder explore a message it previously would not have, but by making it chose the correct message among its final candidates. As such, concentrating additional symbols on the last spine is preferable.

Figure 2.7 analyzes the second proposed scheme, by varying the number of trailing message bits forced to zero between $t = 1$ and $t = 16$. In this case the gap to capacity

reduces with increasing t until $t = 7$, at which point the rate loss due to the unusable bits exceeds the benefit of increased decoding reliability. Unlike tail symbol repetition, the optimum value for t shows no SNR dependence here.

Ultimately, under optimal parameter choice both schemes show essentially the same behavior. Tail zeroing has the advantage that the best parameter choice is SNR independent. The disadvantage is that it requires explicit support in the decoder, which needs to take into account that the last message bits have fixed values, while tail repetition will be automatically supported by a decoder that already has to deal with a variable number of symbols due to puncturing.

2.5 Non-Zero Error Coding

Until now we have considered spinal codes in the context of zero-error VLFT coding. While the VLFT model is useful to decouple the coding procedure from the choice of the stopping time, it is not a realistic transmission model, as it requires the decoder to already know the transmitted message for the purpose of determining the stopping time. While this does not matter asymptotically, it produces unrealistic results at short blocklengths.

Here we will consider the more practical VLF coding model with a permissible block error probability ϵ, which requires the stopping time to be determined based on the output symbols observed at the decoder only. There are two general approaches that can be pursued: One is to perform a reliability estimation as part of the decoding procedure and stop once the computed error probability falls below the specified limit. The other is to include a separate error-detecting code with the message and stop transmission once no error is detected at the decoder.

If possible, the first method is preferable especially at short blocklengths, as the reduction in the number of usable bits due to inclusion of an error-detecting code produces the largest rate losses here. An example where accurate reliability estimation is possible is the reliability output Viterbi decoder (ROVA) [25], which produces an accurate posterior probability of the message corresponding to the found ML codeword. Together with an extension of this method to tail-biting codes [37], the performance of short variable-length convolutional codes is considered in [36].

As we are not aware of a good reliability estimation method for sequential decoding, we will evaluate the generally applicable method of an error-detecting code here. Specifically we prepend a γ-bit cyclic redundancy check (CRC) code to the message and stop decoding once it is satisfied, where γ is chosen large enough to reduce the block error probability below ϵ.

Figure 2.8 (left) shows the average rate[2] over the average blocklength for the capacity $C = 0.5$ BSC channel. The VLF and VLFT achievability and converse bounds as given in [24] have been included, where we use $\epsilon = 10^{-3}$ for the VLF model. To meet this error bound the spinal code is combined with a 13-bit CRC code.

[2]Unlike all other graphs, the rate is defined as $R = \frac{n}{\mathbb{E}[N]}$ here, with N being the block length. Otherwise the results would not be comparable to the VLF/VLFT bounds.

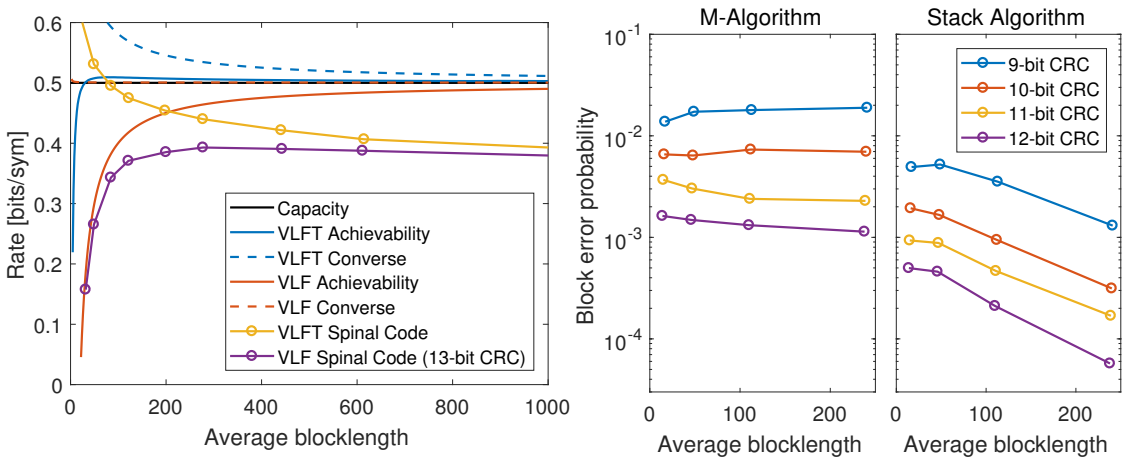


Figure 2.8: Left: Rate over average blocklength for BSC with crossover probability $\delta = 0.11$. The VLF results are for probability of error $\epsilon \leq 10^{-3}$. Achievability and converse bounds are due to [24]. Right: Block error probability over average blocklength for AWGN channel with 10dB SNR. Differently sized CRC codes are compared, with decoding using the M-algorithm or stack algorithm. In the latter case error probability reduces with blocklength.

We can see that in the VLFT coding paradigm the average rate decreases with blocklength. The reason is that at short blocklengths we derive more benefit from the ability to respond to specific channel noise realizations.[3] At long blocklengths the effect of channel dispersion averages out.

When the VLF model is used, this effect is counteracted by the rate loss due to inclusion of the CRC checksum, such that there is a blocklength at which the rate is maximized (here at approximately 300 symbols). However, as long as the average blocklength is not too small (here above 200 symbols) there is little change in the rate.

Figure 2.8 (right) shows how the block error probability changes with average blocklength for an AWGN channel at 10dB SNR. The error probability is shown for CRC sizes between 9 and 12 bits and for two different decoding algorithms. The interesting effect here is that for the M-algorithm the error probabilities are much higher and stay constant with blocklength, while for the stack algorithm the probability decreases with blocklength.

In this context, the essential difference between the two algorithms is that the M-algorithm always produces a candidate message (unless this is explicitly prevented, e.g., by limiting the minimal metric), while the stack algorithm may abort without producing a candidate. As such, the M-algorithm will test many more messages against the CRC, resulting in more errors. For the stack algorithm, error decreases with blocklength. While this is generally expected behavior, in this case the primary factor is likely the drop in average rate rather than the error exponent.

[3] Additionally the noiseless termination symbol impacts the rate at very short blocklengths. For example, given a random 1-bit message, the decoder can achieve a rate of 2/3 bits per symbol by guessing the two possible values in sequence, even if it completely disregards the channel outputs.

Chapter 3

Decoders

The decoding method for spinal codes originally introduced in [22] is the "bubble decoder", which is a variation on the M-algorithm, a well-known representative of the sequential decoding paradigm used for the decoding of tree codes. While this algorithm has a simple implementation and predictable performance, it is also computationally demanding.

In this chapter we will introduce the sequential decoding method, and investigate the performance of different algorithms when applied to spinal codes. In particular we consider the established M-algorithm, the stack algorithm and the adaptive effort algorithm, as well as two newer algorithms proposed specifically for use with spinal codes.

We consider the decoding problem at a fixed point in time, after a certain number of symbols have been received, with the decoder either producing a candidate message or indicating that more symbols are needed for successful decoding. If a candidate message is produced, decoding can either be stopped based on the stopping criteria discussed in section 2.5, or attempted again after more symbols have been received.

For a given message $\mathbf{m} \in \{0,1\}^n$, let $\mathbf{x}(\mathbf{m})$ denote the corresponding transmitted codeword (up to the current time) and $\mathbf{y}$ the received symbols. By $\mathbf{x}_i$ and $\mathbf{y}_i$ we denote the symbols associated with the i-th spine $\mathbf{s}_i$. It should be noted that the symbols in $\mathbf{x}_i$ and $\mathbf{y}_i$ are generally not consecutive in transmission order, but we can always determine which symbols belong to which spine based on the transmission schedule shared between encoding and decoder. Furthermore, due to puncturing or use of tail symbols, the number of symbols is not necessarily the same for all spines.

3.1 Sequential Decoding and the Fano Metric

Spinal codes have some similarity to convolutional codes. While for convolutional codes with short constraint lengths, efficient maximum likelihood (ML) decoding is possible using the Viterbi [34] or BCJR [3] algorithms, the more complex structure of spinal codes precludes this. Insofar as the term is still meaningful in this context, the constraint length of spinal codes is arbitrarily large, as each bit will influence the output for *all* following bits.

An alternative way of decoding convolution codes is *sequential decoding*, which tries to approximate ML decoding by exploiting the tree structure of the code. Starting with the

root of the tree, at each step the decoder maintains a limited set of tree paths it has explored, and then decides which of these paths to extend to the next message fragment based on a metric, until either a leaf is reached or the decoder gives up.

This general idea can be realized in a number of different ways, some of which are discussed below. Two extremes of this approach are to either consider all paths, which recovers brute-force ML decoding, or to always only follow the edge with the largest metric at each step, which is fast, but will only find the correct message at sufficiently low rate. Practical sequential decoding algorithms operate between these two extremes. For a survey of many algorithms see [1].

Sequential decoding requires a metric operating on message prefixes to guide exploration of the coding tree. We will motivate it here and refer to Massey [19] for a more rigorous treatment. This metric is inspired by ML decoding, which finds a message $\hat{\mathbf{m}}$ satisfying

$$\hat{\mathbf{m}} = \operatorname*{arg\,max}_{\mathbf{m}' \in \{0,1\}^n} P(\mathbf{x}(\mathbf{m}') \mid \mathbf{y}) = \operatorname*{arg\,max}_{\mathbf{m}' \in \{0,1\}^n} \frac{P(\mathbf{y} \mid \mathbf{x}(\mathbf{m}'))}{P(\mathbf{y})} P_{\mathbf{m}'}, \tag{3.1}$$

where we will assume that the messages are equiprobable with $P_{\mathbf{m}} = 2^{-n}$. For a memoryless channel, we have $P(\mathbf{y} \mid \mathbf{x}(\mathbf{m}')) = \prod_{i=1}^{n/k} P(\mathbf{y}_i \mid \mathbf{x}_i(\mathbf{m}'))$ and will further require that

$$P(\mathbf{y}) = \prod_{i=1}^{n/k} P_0(\mathbf{y}_i), \quad \text{where} \quad P_0(\mathbf{y}_i) := \sum_{\mathbf{x}_i} P(\mathbf{y}_i \mid \mathbf{x}_i) Q(\mathbf{x}_i), \tag{3.2}$$

and $Q(\cdot)$ is the probability density of the symbols produced by the constellation mapper. In other words, the probability of the output is independent for each spine and is determined by the constellation mapper and channel transition structure only. Due to their randomized structure, this assumption holds for spinal codes. Equation (3.1) is thus equivalent to maximizing the metric

$$M(\mathbf{m}) := \sum_{i=1}^{n/k} \log \frac{P(\mathbf{y}_i \mid \mathbf{x}_i(\mathbf{m}))}{P_0(\mathbf{y}_i)} + \log P_{\mathbf{m}}. \tag{3.3}$$

Denoting by $\mathbf{m}_{..j} := \mathbf{m}_1, \ldots, \mathbf{m}_j$ the prefix of $\mathbf{m}$ consisting of the first j spines, we note that for $i \leq j$, $\mathbf{x}_i(\mathbf{m}) = \mathbf{x}_i(\mathbf{m}_{..j})$ only depends on the prefix, and that $P_{\mathbf{m}_{..j}} = 2^{-kj}$ is the probability of an arbitrary message having this prefix. Using this, we can extend the metric to work on message prefixes by

$$M_{..j}(\mathbf{m}_{..j}) := \sum_{i=1}^{j} \left[\log \frac{P(\mathbf{y}_i \mid \mathbf{x}_i(\mathbf{m}_{..j}))}{P_0(\mathbf{y}_i)} - k \right]. \tag{3.4}$$

This is known as the Fano metric, with the quantity k being referred to as the *bias*. The form given here is somehow unusual, as it is expressed in terms of spines. A more typical formulation in terms of individual symbols is

$$\hat{M}_{..j}(m_{..j}) := \sum_{i=1}^{j} \left[\log \frac{P(y_i \mid x_i(m_{..j}))}{P_0(y_i)} - R \right], \tag{3.5}$$

which is equivalent to (3.4) if $\forall i : \mathbf{y}_i \in \mathcal{Y}^L$, such that $R = \frac{k}{L}$. For spinal codes this is not necessarily the case, as due to puncturing or additional tail symbols, some $\mathbf{y}_i$ may have more symbols than others.

Both forms have an intuitive interpretation. In both cases the logarithmic term computes an information density. The expectation of the per-symbol information density is R, so subtracting R as a bias centers the metric around zero, allowing the comparison of metrics for different-length prefixes. Similarly in the per-spine formulation we know that each spine carries k bits of information, so these are subtracted to renormalize the metric. Both versions can be used, though we have found that the per-symbol formulation (with R taken as the overall rate for the whole codeword) has slightly better rate performance.

Some sequential decoding algorithms only need to compare the metric between paths of equal length. In this case, noting that the $\log P_0(\mathbf{y}_i)$ terms and the biases will be the same for all paths, we can reduce the metric to

$$M'_{\cdot\cdot j}(\mathbf{m}_{\cdot\cdot j}) := \sum_{i=1}^{j} \log P(\mathbf{y}_i \mid \mathbf{x}_i(\mathbf{m}_{\cdot\cdot j})). \tag{3.6}$$

This is practically significant as $M'_{\cdot\cdot j}(\cdot)$, or a form that is equivalent under maximization, is for many common channel types both cheap to compute and independent of channel parameters. $\log P_0(\mathbf{y}_i)$ on the other hand requires explicit evaluation of the sum (3.2), or some form of approximation.

3.1.1 Metric for Specific Channels

Let us now evaluate the metric for the channels we are primarily interested in, namely the BSC and AWGN channels. For a BSC with crossover probability $\delta \leq \frac{1}{2}$, $\mathbf{x}_i, \mathbf{y}_i \in \{0,1\}^{L_i}$, and assuming that channel inputs are equiprobable, we have

$$\begin{aligned} P(\mathbf{y}_i \mid \mathbf{x}_i) &= \delta^{H_i}(1-\delta)^{L_i - H_i} \\ P_0(\mathbf{y}_i) &= 2^{-L_i}, \end{aligned}$$

where $H_i := H(\mathbf{y}_i, \mathbf{x}_i)$ is the Hamming distance. As such, the Fano metric becomes

$$\begin{aligned} M^{BSC}_{\cdot\cdot j}(\mathbf{m}_{\cdot\cdot j}) &= \sum_{i=1}^{j} \left[H_i \log \delta + (L_i - H_i)\log(1-\delta) + L_i - k \right] \\ &= \sum_{i=1}^{j} \left[H_i \left(\log \delta + 1 - \tfrac{k}{L_i} \right) + (L_i - H_i) \left(\log(1-\delta) + 1 - \tfrac{k}{L_i} \right) \right], \end{aligned}$$

where $\frac{k}{L_i} = R$ if L_i is the same for all spines. Here the Fano metric separates into two terms, one for each matching bit, and another for each differing bit. As such, it is easy to compute, but depends on CSI in the form of the parameter δ.

The reduced metric is given by

$$M'^{BSC}_{\cdot\cdot j}(\mathbf{m}_{\cdot\cdot j}) = \sum_{i=1}^{j} \left[H_i(\log \delta - \log(1-\delta)) + L_i \log(1-\delta) \right],$$

where the last term is input-independent and the factor $\log \delta - \log(1-\delta) < 0$ only rescales the metric. Under maximization this metric is thus equivalent to

$$M'^{BSC}_{..j}(\mathbf{m}_{..j}) = -\sum_{i=1}^{j} H_i. \tag{3.7}$$

For the real-valued AWGN channel with noise distributed by $\mathcal{N}(0, N)$, we have

$$P(\mathbf{y}_i \mid \mathbf{x}_i) = \frac{1}{\sqrt{2\pi N}} \exp\left(-\frac{\|\mathbf{y}_i - \mathbf{x}_i\|^2}{2N}\right),$$

where $\|\cdot\|$ is the L_2 norm. For the AWGN the distribution of mapped symbols $Q(\mathbf{x}_i)$ will not be equidistributed in the general case. Even under that assumption, no meaningful simplification of $P_0(\mathbf{y}_i)$ is possible without approximation. As such, the Fano metric retains the form (3.4). For the reduced metric

$$M'^{AWGN}_{..j}(\mathbf{m}_{..j}) = \sum_{i=1}^{j} \left[-\|\mathbf{y}_i - \mathbf{x}_i\|^2 \frac{\log e}{2N} + \log \frac{1}{\sqrt{2\pi N}}\right]$$

on the other hand, it is once again possible to neglect the constant offset and the rescaling under maximization, yielding the CSI-independent expression

$$M'^{AWGN}_{..j}(\mathbf{m}_{..j}) = -\sum_{i=1}^{j} \|\mathbf{y}_i - \mathbf{x}_i\|^2. \tag{3.8}$$

With these general considerations out of the way, we will now consider specific sequential decoding algorithms.

3.2 M–Algorithm

The M-algorithm, also known as beam search in the context of heuristic search, maintains a set P_i of at most M paths of equal depth i, starting with a set P_0 containing only the root node. At each step, all paths in P_i are expanded by the 2^k possible next message fragments, yielding $2^k M$ new paths of depth $i+1$. Of these only the M paths with highest metric are retained in P_{i+1} for the next step. This is repeated until the leaf level is reached, in which case the highest metric path in $P_{n/k}$ is returned as the most likely message.

Each step performs $2^k M$ node expansions and the best M paths can be found in $O(2^k M)$ metric comparisons using a fast selection algorithm. The total time complexity for all $\frac{n}{k}$ steps is thus $O(2^k M \frac{n}{k})$ expansions and $O(2^k M \frac{n}{k})$ comparisons. It should be noted that each expansion involves one hash calculation, as well as the generation of L symbols and their metric using the RNG and constellation mapper, where L refers to the number of symbols per spine. The space complexity is $O(2^k M)$ for the intermediate sets (storing spines and path metrics) and $O(M \frac{n}{k})$ for backtracking information (which edges were followed at each step).

The M-algorithm has a number of advantages: First, as it only compares equal-length paths, it can use the reduced metric, which is usually efficient to compute and CSI independent. Second, it does not require special data structures (like a heap), but does need selection. Finally, its performance is very predictable, as always the same number of node expansions are performed. At the same time, this is also the main drawback: Even if the correct message could have been found by following a single path with particularly high prefix metrics, the algorithm is not able to adapt to this condition and will still perform all $2^k M \frac{n}{k}$ expansions.

3.2.1 Bubble Decoder

Perry et al. [22] propose an extension of the M-algorithm specifically in the context of spinal code decoding: For each path, the next D levels of the decoding tree (2^{kD} nodes) starting from the end of the path are stored. The highest metric in the leaf level of this partial decoding tree is defined to be the overall metric of the path.

Apart from this, the procedure matches the M-algorithm: At each step, the M partial decoding trees are extended by one level to $2^{k(D+1)}$ nodes, which may be reinterpreted as $2^k M$ paths with partial decoding trees of size 2^{kD}. Of these the M paths with the largest metric (i.e., the largest leaf metric) are retained. For $D = 0$ this reduces to the ordinary M-algorithm.

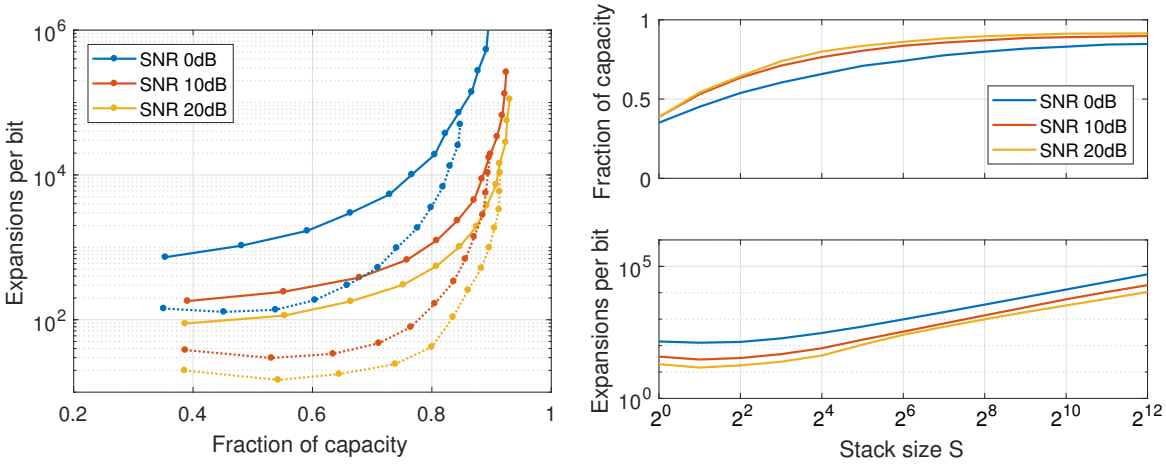
Effectively, the behavior is similar to using the M-algorithm with $M' = 2^{kD} M$, with the difference being in the selection of which paths are retained. Rather than picking the best $2^{kD} M$ paths, the best M sub-trees of size 2^{kD} are used. This makes path selection less accurate, but reduces the cost of the selection (as it is performed over $2^k M$ rather than $2^{k(D+1)} M$ elements) and reduces the necessary backtracking storage (M rather than $2^{kD} M$ per step). As such, it may be seen as an approximation of the M-algorithm. In our experiments, this trade-off does not appear to be favorable, especially as the coarse-grained depth parameter D does not allow for fine tuning.

3.3 Stack Algorithm

The stack algorithm [16] [46] maintains a "stack" of at most S paths sorted by metric, starting with only the root node. At each step, the path with the largest metric is removed from the stack, expanded, and the 2^k new paths are placed back on the stack. Decoding ends when the path at the top of the stack reaches the leaf level, in which case it is returned as the most likely message.

Once the stack size limit S has been reached, it becomes necessary to discard paths. Denote by M_{min} the lowest metric currently in the stack and by $M_{discard}$ the highest metric ever discarded. When attempting to add a new path with metric M_{new}, if $M_{new} \leq M_{min}$ or $M_{new} \leq M_{discard}$, the path is discarded. Otherwise, the lowest metric path is discarded from the stack, and the new path added. Due to the $M_{discard}$ threshold, the stack may run empty, in which case decoding is aborted.

As the stack algorithm compares between paths of different length, it uses the Fano



(a) Comparison of rate and complexity for M-algorithm (solid) and stack decoder (dotted)

(b) Impact of stack size of stack decoder on rate and complexity

Figure 3.1: Left: Number of node expansion per message bit for a certain achieved fraction of capacity for the AWGN channel at different SNRs, using the M-algorithm (solid) and stack decoder (dotted). Each point corresponds to a variation of beam width M or stack size S from 2^0 to 2^{12} respectively. Right: Performance of the stack decoder with varying stack size S. This is the same data represented in a different way.

metric. At each step the locally optimal decision of expanding the highest metric path is performed. The $M_{discard}$ threshold ensures that this stays true even once the stack runs full. Abortion due to stack underflow occurs exactly when this property can no longer be guaranteed.

While the term "stack" is used for historical reasons, the most natural implementation of the stack algorithm is based on a min-max-heap, the max-heap property being used to extract the next path to expand and the min-heap property being used to discard the lowest metric path. Together with insertion, these operations have $O(\log S)$ time complexity. As heaps are relatively expensive, approximate implementations based on binning are also possible.

While the M-algorithm has an easy to determine constant decoding complexity, the number of expansions performed by the stack decoder will depend both on the transmitted and the received codewords. At rates sufficiently below capacity, it is expected that the stack algorithm only needs to explore the single path with highest metric at each step. Conversely, under unfavorable conditions the stack algorithm could end up exploring the entire decoding tree, given $S \geq 2^k \frac{n}{k}$. Even at smaller stack sizes, if $S = 2^k\gamma$, the stack algorithm can explore at most $2^{k\gamma}$ nodes, which is unacceptably large even for small γ.

While this makes the worst case complexity of the stack algorithm grow exponentially with S, this is not the case for the average decoding complexity observed in practice. Figure 3.1 (right) shows how the choice of stack size impacts the achieved rate and the decoding complexity, in terms of the number of node expansions per bit. While it is not entirely

obvious from the doubly-logarithmic plot, the complexity increases only linearly with S.

Figure 3.1 (left) provides a comparison between the M-algorithm and the stack algorithm, by showing the average number of node expansions necessary to achieve a certain fraction of capacity. We can observe that except at very high rates, the dynamic complexity of the stack algorithm allows it to perform approximately one order of magnitude less node expansions than the fixed complexity M-algorithm. It should be noted however, that the number of node expansions is not an entirely fair measure: First, the heap management performed by the stack algorithm will usually be more expensive than the selection performed by the M-algorithm. Second, the Fano metric used by the stack algorithm is more expensive to compute.[1] Because these factors depend heavily on implementation details, we will nonetheless use the number of expansions as our proxy for overall complexity.

Another effect that can be seen in Figure 3.1 is that the stack algorithm saturates at a lower rate than the M-algorithm, especially at low SNR. We are not entirely sure why this occurs, but conjecture that this is related to the reliance of the Fano metric on channel parameters. In our experiments we provided the stack decoder with the actual noise variance of the channel. Possibly this prevents the stack decoder from effectively making use of low-noise realizations of the channel.

3.4 Forward Stack Decoder

The forward stack decoder (FSD) algorithm proposed by Yang et al. [41] for the purpose of decoding spinal codes combines ideas of the stack algorithm and the M-algorithm. The FSD algorithm divides the coding tree into layers of depth D. Within each layer, starting from a seed of at most M paths of length iD, the stack algorithm is used until either the stack size S is exceeded or until a path to the next layer (of length $(i+1)D$) reaches the top of the stack. The best M paths that reach the next layer are used to seed the next layer. If there are none such paths, decoding is aborted.[2]

Based on the premise that for small D the bias for different-length paths within one layer is insignificant, the authors suggest to use the reduced ML metric rather than the Fano metric in the per-layer stack algorithm. While it is not explicitly mentioned in the paper, a reduced form that does not include any constant offsets, such as (3.7) and (3.8) needs to be used.

Just like the stack algorithm, the FSD decoder has variable complexity, however the use of stack overflow as an immediate abortion criterion makes the worst-case more amenable: There may be at most $O(S\frac{n}{kD})$ expansions and the stack management can be performed in $O(S \log S \frac{n}{kD})$ comparisons, if a max-heap is used. The storage requirement is $O(S)$ for the stack and $O(M\frac{n}{k})$ for backtracking information.

In [41] the FSD decoder is compared to the M-algorithm and stack algorithm using

[1] The impact of the Fano metric can be reduced by observing that it only needs to be computed once per symbol and can be reused across decoding attempts.

[2] It is important for the performance of the algorithm that decoding is not aborted immediately on stack overflow. Instead decoding is still started at the next layer if there are any paths that reach it, even though none of them arrived at the top of the stack.

a spinal code of message length $n = 48$, fragment size $k = 8$, mapper bits $c = 10$, no puncturing and no tail symbols. The stack decoder was configured to immediately abort on stack overflow and used a very large stack size to compensate. Under this specific configuration we were able to reproduce the results from the paper, which show that all three algorithms have the same rate performance, but the FSD algorithm has the lowest complexity except at very low SNR.

Unfortunately, these parameters are not very practical. The use of the reduced metric despite the comparison between paths of different length becomes particularly problematic when spines have a differing number of output symbols. This prevents the FSD algorithm from making efficient use of puncturing and tail symbols. To compensate for the lack of puncturing the authors use a large message fragment size $k = 8$, which results in a much higher baseline computational load. Based on the results from our discussion of non-zero error coding in section 2.5, the overhead of error detection at a message size of $n = 48$ is still very high. While all decoding methods perform worse with increasing message size, the average rate deteriorates particularly quickly for the FSD algorithm.

The overall effect is that the rate at which the FSD algorithm saturates when applied to our standard configuration ($n = 256, k = 4$, puncturing with factor $p = 8$) is approximately $1.5\times$ lower than that of the stack algorithm. Our conclusion is that while the FSD algorithm may be useful for some very specific configurations, it is not a good choice for a general purpose spinal code decoding algorithm.

3.4.1 Sliding Feedback Decoder

The sliding feedback decoder (SFD) proposed by Xu et al. [39] in the context of spinal codes, operates on a similar principle as the forward stack decoder. Instead of dividing the coding tree into non-overlapping layers of depth D, the SFD decoder uses a sliding window of size D. Starting from a single path of length i, the stack algorithm (with reduced metric) is used to find a path of length $i + D$. Then the length $i + 1$ prefix of this path is taken and the procedure is restarted. As such, the path is always extended by one edge at each step, but the extension is guided by looking D edges ahead.

At least as described, the SFD decoder does not limit the size of the internal stack. As such, each step may expand up to $O(2^{kD})$ nodes, for an overall worst-case complexity of $O(2^{kD}\frac{n}{k})$. Just as with the stack algorithm, the average complexity is expected to be lower. The storage requirement is $O(2^{kD})$ for the stack and $O(\frac{n}{k})$ for backtracking information, as (apart from the contents of the stack) only a single path is considered.

In [39] the authors compare the performance of the SFD decoder to the FSD algorithm and the M-algorithm, for spinal codes with $n = 32, k = 4$ and no puncturing. The claim is that all three algorithms provide the same rate performance, while SFD has lower complexity than the FSD algorithm, especially at low SNR. Unfortunately we were not able to reproduce these results, possibly due to an error in our implementation, or a hidden difference in parametrization.

When testing the SFD decoder with our standard spinal coding parameters the rate performance for the suggested value of $D = 3$ was similar to the FSD algorithm. Unlike the FSD algorithm, the rate does not saturate at a low level and can be improved by choosing

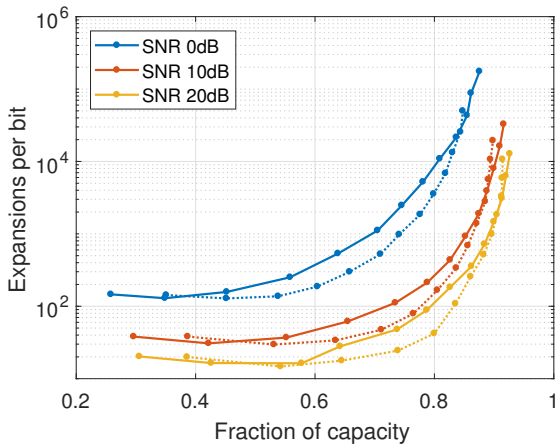


Figure 3.2: Rate versus complexity for adaptive effort algorithm (solid) and stack decoder (dotted). Each dot corresponds to a variation of the beam width M or the stack size S between 2^0 and 2^{12}. For the adaptive effort algorithm, the threshold $T = 20\sigma^2$ and maximum number of threshold reductions $r = 10$ were used.

larger values of D. However, the decoder complexity increases rapidly, so that both the M-algorithm and the stack algorithm have lower complexities for the same rate.

3.5 Adaptive Effort Decoder

The FSD and SFD decoders described in the previous section attempt to combine the M-algorithm and stack-decoder to reap the benefits of both schemes. The use of the stack decoder allows the computational effort to be variable, while preserving the general scaffold of the M-algorithm makes use of the cheaper and CSI independent reduced metric possible, and limits the worst-case complexity.

A different approach suggested by Simmons [31] in the context of convolutional codes, is an adaptive effort variation on the M-algorithm using an additional threshold parameter T: After the $m \leq M$ paths at a certain depth i have been expanded to $m2^k$ paths of depth $i+1$, the best metric $M_{\max}$ among them is determined and only paths with metric above $M_{\max} - T$ are retained. In case the resulting number of paths is larger than the limit M, Simmons suggest to continue reducing T by a factor $\alpha = 0.9$ until the number of surviving paths falls below M.

For the application to convolutional codes M is chosen sufficiently large that threshold reductions occur rarely. For spinal codes, we expect that they will be common for decoding attempts at high rates. To avoid wasting computational efforts on decoding attempts that are unlikely to succeed, we limit the maximum number of threshold reductions to r, so that the threshold may not be reduced below $T_{\min} = T\alpha^r$. If the number of paths within threshold $T_{\min}$ still exceeds M, the decoding attempt is aborted.

The adaptive effort algorithm has the same worst-case number of node expansions as the M-algorithm, that is $O(2^k M \frac{n}{k})$. The worst-case number of metric comparisons is

Decoder	Expansions	Comparisons	Storage (Step)	Storage (Backtracking)
M-algorithm	$O(2^k M \frac{n}{k})$	$O(2^k M \frac{n}{k})$	$O(2^k M)$	$O(M \frac{n}{k})$
Bubble decoder	$O(2^{k(D+1)} M \frac{n}{k})$	$O(2^k M \frac{n}{k})$	$O(2^{k(D+1)} M \frac{n}{k})$	$O(M \frac{n}{k})$
Stack algorithm	$O(2^{Sk/2^k})$	$O(2^{Sk/2^k} \log S)$	$O(S)$	$O(S \frac{n}{k})$
FSD	$O(S \frac{n}{kD})$	$O(S \log S \frac{n}{kD})$	$O(S)$	$O(M \frac{n}{k})$
SFD	$O(2^{kD} \frac{n}{k})$	$O(kD2^{kD})$	$O(2^{kD})$	$O(\frac{n}{k})$
Adaptive Effort	$O(2^k M \frac{n}{k})$	$O(2^k M(\frac{n}{k} + r))$	$O(2^k M)$	$O(M \frac{n}{k})$

Table 3.1: Worst-case time and space complexities for different decoding algorithms. Average complexities may be lower. Especially the time complexities for the stack algorithm are misleading.

$O(2^k M(\frac{n}{k} + r))$. However, it has two advantages: First, the average number of expansions and comparisons is expected to be much lower than the worst-case complexity. Second, while the number of comparisons is (if we ignore the r term) the same under big-O notation, the practical performance will be better, as selection algorithms like Floyd-Rivest, while asymptotically linear, have large constant factors.

Figure 3.2 compares the adaptive effort algorithm with the stack algorithm in terms of node expansions at a given fraction of capacity. We choose $T = 20\sigma^2$ and $r = 10$, while the beam width M is varied. Here σ^2 refers to the noise variance of the channel and is assumed to be known (though the ability to reduce the threshold makes this less important than for the stack algorithm).

From the comparison we observe that the adaptive effort algorithm performs similarly to the stack algorithm at high rates and within a factor of two at lower rates. Due to the high heap management overhead of the stack decoder, the practical performance is better than what this comparison would indicate.

3.6 Overall Comparison

Table 3.1 compares the worst-case time and space complexities of the different decoding algorithms. Apart from the M-algorithm and bubble decoder, the average case time complexities are expected to be lower, as the decoding effort is variable. As these complexities do not directly relate to the achieved rate at a given level of computation, it has hard to draw meaningful conclusions from them.

Figure 3.3 shows an overall comparison of the M-algorithm, stack decoder and adaptive effort decoder. The decoder parameters have been chosen to be constant across the SNR range, but produce similar results, as seen in the gap to capacity in the left part of the figure. Matching the rate curves exactly would require adjusting the decoder parameters for each SNR level. This would yield a fairer comparison, but a less practical mode of operation.

The computational effort normalized to the M-algorithm is shown on the right, both in

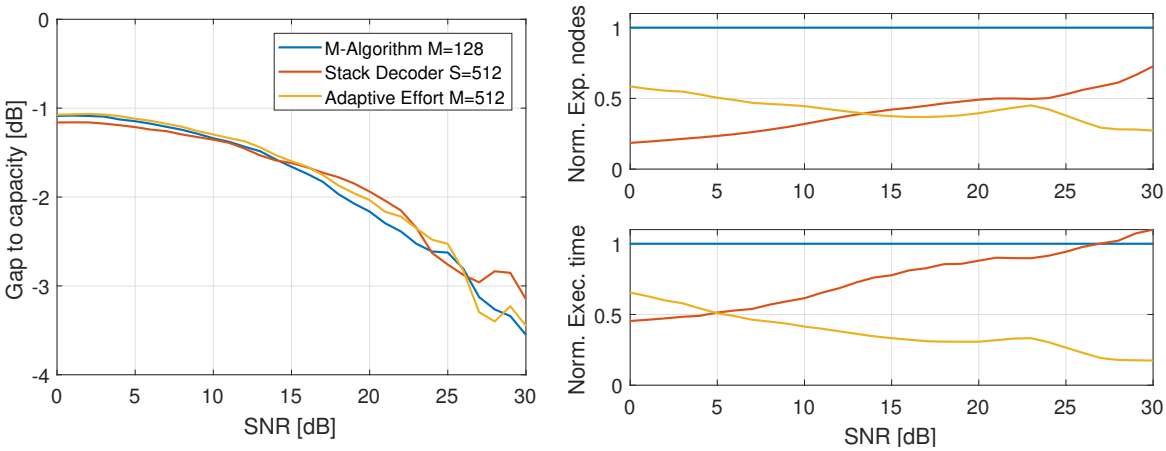


Figure 3.3: Rate and decoding complexity comparison of M-algorithm, stack decoder and adaptive effort decoder, with parameters chosen to yield similar rate curves. The decoding complexity shows both the number of expanded nodes (top) and the execution time of our implementation (bottom), normalized to the M-algorithm as baseline.

terms of the number of expanded nodes, and in terms of the actual execution time of our decoder implementations. While the latter is subject to implementation details, we include these results here to show that the number of expanded nodes is not an entirely unbiased comparison metric, because the constant overhead of the algorithms differs. As a rule, the heap management of the stack algorithm is most expensive, followed by the selection performed by the M-algorithm, followed by the adaptive effort decoder, which only performs simple maximization and filtering.

Our conclusion is that the adaptive effort algorithm is likely the best overall choice. It performs consistently better than the M-algorithm, while also being competitive with the stack algorithm and having a simpler implementation, in particular by not using the Fano metric and not employing a heap. As such, it is also a viable choice for a hardware implementation. In [11] a hardware implementation of the M-algorithm for spinal coding is considered, where the needed selection circuitry is identified as the primary issue.

An important sequential decoding algorithm which we have not discussed is the Fano algorithm [8], which uses a dynamically adjusted threshold and backtracking to explore the coding tree while only storing a single path. The low memory requirement makes this decoder interesting for hardware implementations, but not a good choice for the software implementations considered here, as it explores paths multiple times, which is considerably more expansive for hash-based spinal codes than for convolutional codes.

Chapter 4

Decoding Complexity and Achievable Rate

Spinal codes employ sequential decoding, which has a number of known bounds on the distribution and moments of the computational complexity, as well as the maximal rate that can be achieved at sub-exponential complexity. For common channel types, this computational cutoff rate is significantly lower than capacity.

On the other hand, Balakrishnan et al. [4] claim that spinal codes can approach capacity both for the BSC and AWGN channel using a decoder that is polynomial in the message size. This appears to be in contraction to the known sequential decoding bounds.

Simulation results show that in practical configurations, spinal codes operate above the cutoff rate, but also do not approach capacity. In the following, we will summarize the known theoretical results and discuss how they may be reconciled.

4.1 Complexity of Sequential Decoding

An important result on the complexity of sequential decoding is due to Jacobs and Berlekamp [14], who establish that the decoding complexity is approximately Pareto distributed and the ρ-th moment of the complexity diverges above a certain rate $R_c(\rho)$. We will summarize these results and the basic idea behind the proof in the following.

In this context, a sequential decoder is characterized by 1) the choice of next edge to explore being independent from received symbols deeper in the tree, and 2) at least one computation being necessary for each node of every examined path. For example, the Viterbi decoder satisfies the first property, but not the second, as it does not perform one computation for each "examined" path. A helpful thought experiment to consider is: Given a burst of N initial erasures on a binary erasure channel, would the decoder be able to advance past the error burst in (on average) less than $\frac{1}{2}2^{NR}$ computations?

The derivation of the complexity is based on a lower bound on the error probability for list-of-L decoding due to Shannon, Gallager and Berlekamp [29]: Given a DMC and block code of M codewords with block size N and a decoder producing a list of L candidate messages, denote the probability that the transmitted message is not in the list by $P_e(L)$.

If for any $\rho > 0$

$$\frac{M}{L} \geq 2^{N\hat{E}_0'(\rho)+O_1(\sqrt{N})} \tag{4.1}$$

holds, then

$$P_e(L) \geq 2^{-N(\hat{E}_0(\rho)-\rho\hat{E}_0'(\rho))-O_2(\sqrt{N})}, \tag{4.2}$$

where $\hat{E}_0(\rho)$ is the concave hull of the Gallager error function

$$E_0(\rho) = \max_Q E_0(\rho, Q), \quad E_0(\rho, Q) = -\log \sum_{y\in\mathcal{Y}} \left[\sum_{x\in\mathcal{X}} Q(x)P(y \mid x)^{\frac{1}{1+\rho}} \right]^{1+\rho}$$

and $\hat{E}_0'(\rho) = \partial\hat{E}_0(\rho)/\partial\rho$.[1] $Q(\cdot)$ denotes the distribution of the channel inputs and $P(\cdot|\cdot)$ the channel transition structure.

Defining $R = (\log M)/N$ and $R_\rho = \hat{E}_0(\rho)/\rho$, for any $R \geq R_\rho$ there exists an N such that (4.1) is satisfied. For the smallest such N it is shown that (4.2) may be written

$$P_e(L) \geq L^{-\rho}2^{-O(\sqrt{\log L})}, \tag{4.3}$$

that is the error probability is approximately Pareto distributed in L. The relation to sequential decoding is now the following: Considering only the first N symbols (for N chosen as above), the sequential decoder will consider paths of length N in some order. If the decoder may examine at most L paths (ignoring duplicates), then the probability that the correct path is never examined is lower bounded by $P_e(L)$, even if an optimal order is chosen.

As the t-th moment of the Pareto distribution diverges for $t \geq \rho$, it follows that for $R \geq R_\rho$ the ρ-th moment of L diverges, and with it the computational complexity of sequential decoding. Denoting by the *computational cutoff rate* $R_c(\rho)$ the rate at which sequential decoding complexity becomes at least exponential in the block size, this imposes the bound $R_c(\rho) \leq R_\rho = \hat{E}_0(\rho)/\rho$.

Much later, Arikan [2] proposed an alternative derivation, which is not based on error exponent bounds. Arikan consider the problem of guessing the value of a random variable X by sequentially asking questions of the form "Is $X = x$?" in a specified order. Denoting by $G(x \mid y)$ the number of guesses until $x \in \mathcal{X}$ is guessed given $y \in \mathcal{Y}$, Arikan considers the moments $\mathbb{E}[G(X \mid Y)^\rho], \rho > 0$ of the guessing function. Based on a clever application of Hölder's inequality, it is shown that

$$\mathbb{E}[G(X \mid Y)^\rho] \geq (1 + \log|\mathcal{X}|)^{-\rho} \sum_{y\in\mathcal{Y}} \left[\sum_{x\in\mathcal{X}} P_{X,Y}(x, y)^{\frac{1}{1+\rho}} \right]^{1+\rho}.$$

[1] This is Theorem 2 in [29], taking into account the definition $R = \log(M/L)/N$ and that $\sup_{\rho>0}(E_0(\rho) - \rho R) = \sup_{\rho>0}(\hat{E}_0(\rho) - \rho R)$ is satisfied at $R = \hat{E}_0'(\rho)$, which exists for any $R \leq C$ [9]. While the form given in [29] is specific to DMCs, a variation also holds for continuous output channels [32]. Tighter non-asymptotic bounds are discussed in the same paper.

If X is uniformly distributed with $P_X(x) = 1/|\mathcal{X}|$, the bound can be written as

$$\mathbb{E}[G(X \mid Y)^\rho] \geq (1 + \log|\mathcal{X}|)^{-\rho}|\mathcal{X}|^{-\rho} \sum_{y\in\mathcal{Y}} \left[\sum_{x\in\mathcal{X}} P_X(x)P_{Y|X}(y \mid x)^{\frac{1}{1+\rho}}\right]^{1+\rho}. \tag{4.4}$$

As before, a sequential decoder will examine paths at some depth N one by one in a given order, fitting the definition of a guessing function. Even if this order is chosen optimally, the ρ-th moment of the number of considered paths is subject to the bound (4.4). Specializing for a DMC, the bound may be written as

$$\mathbb{E}[G(X \mid Y)^\rho] \geq (1 + NR)^{-\rho}2^{-N[E_0(\rho)-\rho R]},$$

resulting in a cutoff rate $R_c(\rho) \leq E_0(\rho)/\rho$. This is tighter than the bound by Jacobs and Berlekamp for the unusual case where $E_0(\rho)$ is non-concave. The result also holds for memoryless channels with finite input but continuous output.

There are also converse results due to Falconer [7], Savage [27] and Jelinek [15], which establish that there exist tree codes for which the sequential decoding complexity per message bit can be upper bounded by a constant for $R < E_0(\rho)/\rho$. Together, both bounds imply $R_c(\rho) = E_0(\rho)/\rho$, for all $\rho > 0$.

We can now consider $R_c(\rho)$ for specific channels. For the BSC channel with crossover probability δ and $Q(0) = Q(1) = \frac{1}{2}$, we have

$$\begin{aligned} R_c(\rho,\delta) &= -\frac{1}{\rho}\log \sum_{y\in\{0,1\}} \left[\frac{1}{2}\sum_{x\in\{0,1\}} P(y \mid x)^{\frac{1}{1+\rho}}\right]^{1+\rho} \\ &= -\frac{1}{\rho}\log 2\left(\frac{1}{2}\right)^{1+\rho}\left(\delta^{\frac{1}{1+\rho}} + (1-\delta)^{\frac{1}{1+\rho}}\right)^{1+\rho} \\ &= 1 - \frac{1+\rho}{\rho}\log\left(\delta^{\frac{1}{1+\rho}} + (1-\delta)^{\frac{1}{1+\rho}}\right). \end{aligned}$$

In particular, for $\rho = 1$ the cutoff rate becomes

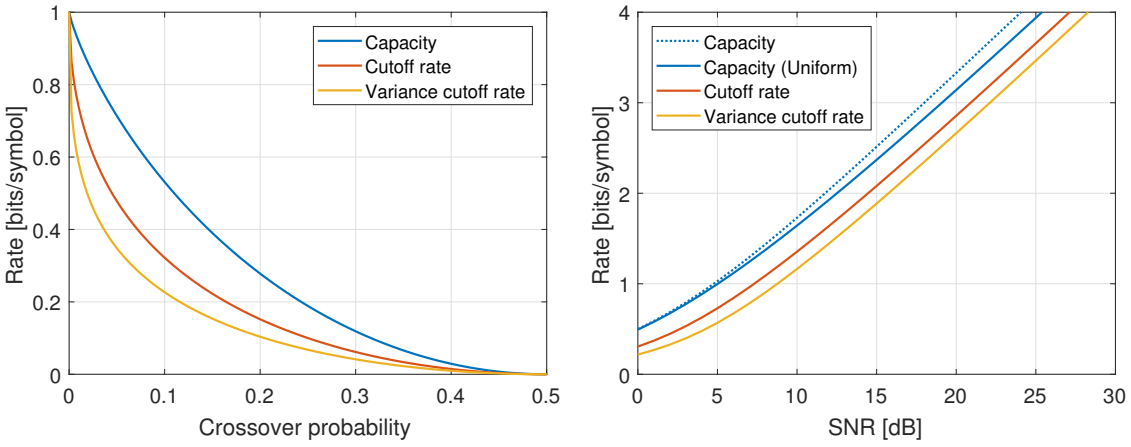
$$R_c(1,\delta) = 1 - 2\log\left(\sqrt{\delta} + \sqrt{1-\delta}\right). \tag{4.5}$$

For the real-valued AWGN with noise distribution $\mathcal{N}(0, N)$ and arbitrary $Q(x)$ we have

$$R_c(\rho, N, Q) = -\frac{1}{\rho}\log\int dy \frac{1}{\sqrt{2\pi N}}\left[\sum_{x\in\mathcal{X}} Q(x)\exp\left(-\frac{(y-x)^2}{2N(1+\rho)}\right)\right]^{1+\rho}$$

For the case of integral ρ and introducing $n := 1+\rho$ to simplify the notation, we can evaluate the integral by writing

$$R_c(\rho, N, Q) = -\frac{1}{\rho}\log\int dy \frac{1}{\sqrt{2\pi N}}\prod_{i=1}^{n}\sum_{x_i\in\mathcal{X}} Q(x_i)\exp\left(-\frac{(y-x_i)^2}{2Nn}\right),$$



(a) BSC channel

(b) Real AWGN channel

Figure 4.1: Comparison of channel capacity C and cutoff rates $R_c(\rho = 1)$ and $R_c(\rho = 2)$, beyond which sequential decoding complexity (or its variance) increases exponentially. The AWGN channel uses a uniform constellation of 2^6 symbols. The capacity for non-uniform constellations has been included as a dotted line.

and noting that the exponential may be written as

$$-\frac{1}{2Nn}\sum_{i=1}^{n}(y-x_i)^2 = -\frac{1}{2N}\left(y-\frac{1}{n}\sum x_i\right)^2 - \frac{1}{2N}\left[\frac{1}{n}\sum x_i^2 - \left(\frac{1}{n}\sum x_i\right)^2\right].$$

The first term of the exponential together with the factor $(2\pi N)^{-\frac{1}{2}}$ forms a Gaussian density function and integrates away, leaving

$$R_c(\rho, N, Q) = -\frac{1}{\rho}\log\prod_{i=1}^{n}\sum_{x_i\in\mathcal{X}} Q(x_i)\exp\left\{-\frac{1}{2N}\left[\frac{1}{n}\sum_{i=1}^{n}x_i^2 - \left(\frac{1}{n}\sum_{i=1}^{n}x_i\right)^2\right]\right\}.$$

For $\rho = 1$ this evaluates to

$$R_c(1, N, Q) = -\log\sum_{x_1\in\mathcal{X}}\sum_{x_2\in\mathcal{X}} Q(x_1)Q(x_2)\exp\left(-\frac{(x_1-x_2)^2}{8N}\right). \tag{4.6}$$

Figure 4.1 compares the cutoff rates $R_c(\rho = 1)$ and $R_c(\rho = 2)$ for the BSC and AWGN channels with the respective capacities. Clearly, there is a significant gap between the cutoff rate and channel capacity. It should be noted that for the AWGN channel, the cutoff rates are constellation dependent, and Figure 4.1 shows results for the uniform constellation. We refer back to section 2.2 on constellation mapping for a discussion on probabilistically shaped constellations. A shaped constellation only provides a minor improvement on the cutoff rate and does not close the gap to capacity.

4.2 Achievable Rate for Spinal Codes

Balakrishnan, Iannucci, Perry and Shah [4] have analyzed the achievable rate of spinal codes. In particular, they claim that spinal codes can approach capacity on both the BSC and AWGN channels using a decoder that is polynomial in the message size. We will reproduce the two main theorems (adjusted for notation) in the following. We remind that k is the message fragment size and ν the size of the hash state.

Theorem 1 (Achievable Rate BSC)**.** *Consider an n-bit message encoded with a spinal code with $k \geq 1$ and $\nu = \Theta(k^2 \log n)$ operating over a BSC with crossover probability $p \in (0, 1/2)$. Then, the M-algorithm with $M = n^{O(k^3)}$ decodes all but the last $O(k^3 \log n)$ message bits successfully with probability at least $1 - 1/n^2$, achieving a rate*

$$R \geq C - O\left(\frac{C^2}{k}\right), \quad \textit{where } C = 1 - H(p).$$

It is further noted that by appending $O(k^3 \log n)$ tail bits the erroneous trailing bits can be avoided. If further $n \geq k^5$, then the rate loss due to the tail bits is $o(1/k)$, so that rates within $O(1/k)$ of capacity may be achieved.

Theorem 2 (Achievable Rate AWGN)**.** *Consider an AWGN channel with noise variance bounded below by σ^2_{min}. Consider a spinal code constrained to have average power $\leq P$, with $k > \frac{1}{2}\log(1 + P/\sigma^2_{min})$. Let the code map n message bits to coded symbols using a truncated Gaussian distribution with a certain choice for β and c (omitted here). Let $\nu = \Theta(k^2 c \log n)$, and let the M-algorithm decoder operate with $M = n^{O(k^2)}$. Then the decoder will correctly decode all but the last $O(k^2 \log n)$ message bits with probability at least $1 - 1/n^2$ within time T such that the induced rate $R = n/T$ satisfies*

$$R \geq C_{AWGN}(P) - O(1/k).$$

An additional requirement not mentioned above is that the hash function for the spinal code is chosen uniformly at random from a family $\mathcal{H}$ of hash functions with the *pairwise independence* property. That is, for every $\mathbf{m} \neq \mathbf{m}' \in \{0,1\}^k$ and $\mathbf{s}, \mathbf{s}' \in \{0,1\}^\nu$

$$\mathbb{P}[h(\mathbf{s}_0, \mathbf{m}) = \mathbf{s}, h(\mathbf{s}_0, \mathbf{m}') = \mathbf{s}'] = \mathbb{P}[h(\mathbf{s}_0, \mathbf{m}) = \mathbf{s}]\mathbb{P}[h(\mathbf{s}_0, \mathbf{m}') = \mathbf{s}'] = 2^{-2\nu},$$

where the randomness is over the seed $\mathbf{s}_0$. This assumption is integral to the proof, which is based on the observation that Gallager's random coding error exponent results [9] also hold for only pairwise independent codebooks.

In the above theorems, the parameter k can be used to control the gap to capacity $O(1/k)$. Additionally, k must be chosen sufficiently large such that there exists an $L \in \mathbb{N}$ so that k/L is close to capacity (otherwise Claim 4 from the paper does not hold). This means that k must be much larger than the minimum values given in the theorems. Together with the fact that $M = n^{O(k^3)}$ for the BSC and $M = n^{O(k^2)}$ for the AWGN respectively, this means that the decoding complexity, while polynomial, is also impractically large.

An important property of the proof is that it does *not* make use of the variable-length property of spinal codes. Instead the proof treats the spinal code as a block code with block size $N = \frac{n}{k}L$, where L is chosen so that $R = k/L$ is sufficiently close to capacity. As spinal codes are treated as block codes, and the proof is based on the M-algorithm, the bounds on sequential decoding complexity from the previous section apply. This appears to be a contradiction, as the spinal coding theorems would imply that sub-exponential decoding for $R_c(1) \leq R < C$ is possible. We will try to resolve this contradiction in the following.

We still start by outlining the main part of the proof (Lemma 6 in [4]): Consider a transmitted message $\mathbf{m}$ and any other message $\mathbf{m}'$ that differs from it in any of the first k bits. From the pairwise independence of the hash if can be shown that these messages will also be uniformly distributed and pairwise independent, so that Gallager's random coding exponent results apply.

We now consider only the first ik bits (first iL symbols). The probability that any $\mathbf{m}'$ has higher metric than $\mathbf{m}$ is upper bounded by $P_e(i) \leq 2^{-iLE_R}$, where $E_R = \sup_\rho(E_0(\rho) - \rho R)$ is the random coding error exponent.[2] Then, choosing $i^* \geq (6 \log n)/(LE_R)$ we have $P_e(i^*) \leq n^{-6}$. As such, with probability $1 - n^{-6}$ only messages that do not differ from $\mathbf{m}$ in the first k bits have higher likelihood. There are at most $2^{(i^*-1)k}$ such messages. Choosing $M = 2^{i^*k} = n^{6k/(LE_R)} = n^{6R/E_R}$ for the M-algorithm thus ensures that the correct message is not pruned out with probability $1 - n^{-6}$.

This argument is then carried out inductively, assuming that the messages match in the first jk bits, but differ in one of the bits $jk + 1$ to $jk + k$. Taking the union bound over the n/k decoding stages, we have that apart from the last i^*k bits, the correct message is found with probability $\geq 1 - 1/(kn^5)$. Finally, it is shown that if ν is sufficiently large, the possibility of hash collisions has no impact on this result.

To put the result into context, it is useful to a make a comparison to brute-force ML decoding of a random code with message size n and block size nR. Splitting the message into groups of k bits and encoding each independently, we have a decoding error probability of $P_e \leq 2^{-kRE_R}$ for each group. Choosing $k \geq (\gamma \log n)/(RE_R)$ we have $P_e \leq n^{-\gamma}$. Over all $n/k \leq nRE_R/(\gamma \log n) \leq nRE_R/\gamma$ groups, we thus have a total error probability of $P_e \leq (RE_R/\gamma)n^{-\gamma+1}$. The decoding complexity for each group is $2^k = n^{\gamma/(RE_R)}$, so the overall complexity is $(RE_R/\gamma)n^{1+\gamma/(RE_R)}$. For $\gamma \geq 2$ we have a decoder with polynomial complexity and probability of error falling polynomially. To ensure that $n \geq k$ we need to require that $n/(\log n) \geq \gamma/(RE_R)$. Clearly, such an n exists for all $R < C$ (though it may be very large).

This construction illustrates that it is possible to convert an exponential time decoder with exponentially decreasing error probability into a polynomial time decoder with polynomially decreasing error probability, and this is essentially what the spinal coding theorems do. This also gives us a way to resolve the apparent contradiction with the sequential decoding complexity results.

[2]The proof goes to some effort to characterize the error exponent E_R for rates close to capacity to arrive at the specific form of the theorems above. For the AWGN it is claimed (Appendix B of [4]) that $E_0(Q) = C_{\text{AWGN}} - \epsilon$ is achievable for a truncated Gaussian constellation with sufficiently large β and c. However, this should only be possible for $\rho \to 0$ rather than $\rho = 1$ [9]. Ultimately this has little impact on the main argument of the proof.

Both the considerations by Jacobs and Berlekamp, and those by Arikan consider the case where the sequential decoder must find the correct message without error. If we follow the former methodology and for simplicity assume that decoding complexity is exactly Pareto distributed with CDF $F(L) = 1 - L^{-\rho}$ where ρ such that $R = R_\rho$, we can modify the result to allow a probability of error ϵ by using an upper-truncated Pareto distribution [6] with

$$F(L) = \frac{1 - L^{-\rho}}{1 - \hat{L}^{-\rho}}, \quad 1 \leq L \leq \hat{L},$$

where $\hat{L}$ is the truncation parameter implied by $\epsilon = \hat{L}^{-\rho}$. While for the Pareto distribution moments $t \geq \rho$ do not exist, all moments of the truncated distribution are well defined with

$$\mathbb{E}[L^t] = \frac{\rho}{\rho - t} \frac{1 - \hat{L}^{t-\rho}}{1 - \hat{L}^{-\rho}},$$

where the cases $\rho = 0$ and $\rho = t$ should be evaluated under $\lim_{\rho \to 0}(1 - L^{-\rho})/\rho = \ln L$. Writing the moments in terms of $\epsilon = \hat{L}^{-\rho} < 1$ we have

$$\mathbb{E}[L^t] = \frac{\rho}{\rho - t} \frac{1 - \epsilon^{1 - \frac{t}{\rho}}}{1 - \epsilon},$$

with the special cases

$$\mathbb{E}_{\rho \to 0}[L^t] \to \infty \quad \text{and} \quad \mathbb{E}_{\rho \to t}[L^t] = \frac{\ln \epsilon}{t(1 - \epsilon)}.$$

We can now consider the impact of different convergence behaviors for ϵ on $\mathbb{E}[L^t]$. If we require exponential convergence $\epsilon = 2^{-\gamma n}$, then

$$\mathbb{E}[L^t] = \begin{cases} O\left(2^{(\frac{t}{\rho} - 1)\gamma n}\right) & \text{for } 0 < \rho < t \\ O(n) & \text{for } \rho = t \\ O(1) & \text{for } \rho > t \end{cases}.$$

If we require polynomial convergence $\epsilon = n^{-\gamma}$, then

$$\mathbb{E}[L^t] = \begin{cases} O\left(n^{(\frac{t}{\rho} - 1)\gamma}\right) & \text{for } 0 < \rho < t \\ O(\log n) & \text{for } \rho = t \\ O(1) & \text{for } \rho > t \end{cases}.$$

At this point we should remind that while we use equality everywhere, in the context of sequential decoding complexity these are lower bounds (and the analysis carried out here is not rigorous in any case). As such, the results for $\rho = t$ are likely not particularly meaningful. The important behavior we observe is that for $0 < \rho < t$ the decoding complexity is exponential for exponential error convergence and polynomial for polynomial error convergence, which resolves the contradiction with the spinal coding theorems.

To summarize, it appears that the results on the achievable rate of spinal codes are correct and consistent with modified sequential decoding bounds that take a finite error probability into account, but ultimately also not very meaningful, as the promised polynomial decoding complexity is obtained only as a tradeoff for polynomial error probability convergence.

4.3 Sequential Decoding of Variable-Length Codes

The discussion in the previous section has considered spinal codes as fixed-length block codes of block size $N = \frac{n}{k}L$, where the number of passes L is chosen such that $R = \frac{k}{L}$. In this mode of operation, we have seen that practical operation of spinal codes is limited to rates below the cutoff rate $R_c(1)$. However, simulation of spinal codes shows that rates above the cutoff rate are achievable. The reason for this is that our simulations are based on an entirely different channel model: In Polyanskiy's terms, we are considering zero-error VLFT coding.

Consider a variable length code for which decoding is attempted at a decreasing sequence of rates $R_1, R_2, \ldots$ until at some R_i the codeword is correctly decoded to the transmitted message (as determined by an oracle). Denoting by $\mathcal{E}_R$ the event that decoding does not succeed at rate R, the average rate is given by

$$\begin{aligned}
\mathbb{E}[R] &= R_1\mathbb{P}[\mathcal{E}_{R_1}^c] + R_2\mathbb{P}[\mathcal{E}_{R_2}^c, \mathcal{E}_{R_1}] + R_3\mathbb{P}[\mathcal{E}_{R_3}^c, \mathcal{E}_{R_2}, \mathcal{E}_{R_1}] + \ldots \\
&= R_1(1 - \mathbb{P}[\mathcal{E}_{R_1}]) + R_2(1 - \mathbb{P}[\mathcal{E}_{R_2} \mid \mathcal{E}_{R_1}])\mathbb{P}[\mathcal{E}_{R_1}] \\
&\qquad + R_3(1 - \mathbb{P}[\mathcal{E}_{R_3} \mid \mathcal{E}_{R_2}, \mathcal{E}_{R_1}])\mathbb{P}[\mathcal{E}_{R_2} \mid \mathcal{E}_{R_1}]\mathbb{P}[\mathcal{E}_{R_1}] + \ldots \\
&= R_1 - (R_1 - R_2)\mathbb{P}[\mathcal{E}_{R_1}] - (R_2 - R_3)\mathbb{P}[\mathcal{E}_{R_2} \mid \mathcal{E}_{R_1}]\mathbb{P}[\mathcal{E}_{R_1}] + \ldots \\
&= R_1 - \sum_{i=1}^{\infty}(R_i - R_{i+1})\mathbb{P}\left[\bigcap_{j=1}^{i}\mathcal{E}_{R_j}\right] \\
&\geq R_1 - \sum_{i=1}^{\infty}(R_i - R_{i+1})\mathbb{P}[\mathcal{E}_{R_i}].
\end{aligned}$$

The bound in the last line is expected to be quite good, based on the intuition that if decoding succeeds at rate R_i it will very likely also succeed at rate $R_{i+1} < R_i$.

For the case of spinal codes in particular, based on the results by Jacobs and Berlekamp, we know that the sequential decoding complexity is approximately Pareto distributed according to equation (4.3). Once again we will make the simplifying assumption that the Pareto distribution is followed exactly. If we limit the maximum complexity of a decoding attempt to L, we then have

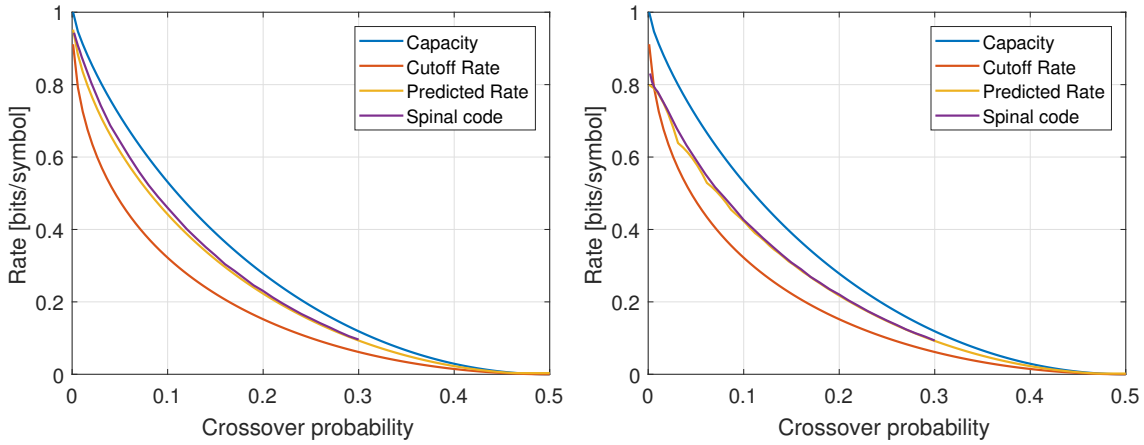
$$\mathbb{P}[\mathcal{E}_{R_\rho}] = L^{-\rho}, \quad \text{for } R_\rho = \frac{E_0(\rho)}{\rho}.$$

Under this distribution, the average rate becomes

$$\mathbb{E}[R] \geq R_1 - \sum_{i=1}^{\infty}(R_i - R_{i+1})L^{-\rho(R_i)},$$

where $\rho(R)$ is implicitly given by $R = R_\rho$ for $R \leq C$ and $\rho = 0$ for $R > C$. For the transmission schedule of spinal codes, we have $R_1 = pk$ and $R_i = R_1/i$, where p is the puncturing factor. The expected rate reduces to

$$\mathbb{E}[R] \geq R_1\left[1 - \sum_{i=1}^{\infty}\frac{1}{i(i+1)}L^{-\rho(R_1/i)}\right]. \tag{4.7}$$



(a) With puncturing (factor $p = 8$)

(b) Without puncturing

Figure 4.2: Comparison of capacity, cutoff rate, predicted rate and simulation results for spinal codes for the BSC channel. The simulation uses $k = 4$ and the M-algorithm at $M = 256$. The predicated rate uses equation (4.7) with $L = 16$.

If we now consider $R_1 \to \infty$ (that is, neglecting rate quantization effects) we can switch from summation to integration. Using the substitution $R = R_1/i$ and the fact that $\rho(R) = 0$ for $R > C$ this yields

$$\mathbb{E}[R] \geq C - \int_0^C L^{-\rho(R)} dR. \tag{4.8}$$

In this form it is clearly visible that $\mathbb{E}[R] \to C$ for $L \to \infty$. However, the implicitly given $\rho(R)$ makes this integral inconvenient to evaluate. By computing the total differential of $f(\rho, R) = E_0(\rho) - \rho R = 0$ we find that

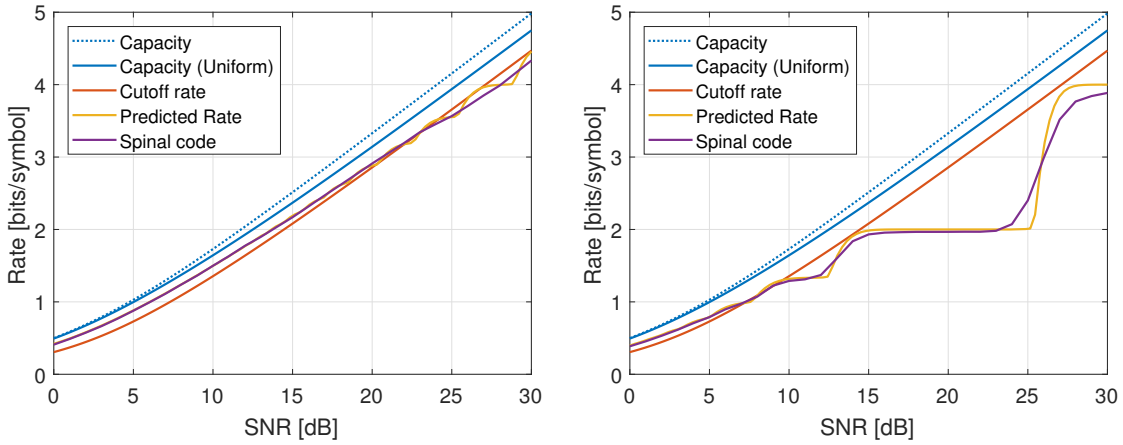
$$\frac{dR}{d\rho} = \frac{1}{\rho}\frac{\partial E_0(\rho)}{\partial \rho} - \frac{E_0(\rho)}{\rho^2} = \frac{\partial}{\partial \rho}\left[\frac{E_0(\rho)}{\rho}\right],$$

which allows us to perform a change of variables in (4.8) towards integration over ρ:

$$\begin{aligned}
\mathbb{E}[R] &\geq C - \int_\infty^0 L^{-\rho}\frac{dR}{d\rho} d\rho = C + \int_0^\infty L^{-\rho}\frac{\partial}{\partial \rho}\left[\frac{E_0(\rho)}{\rho}\right] d\rho \\
&= C + \left[L^{-\rho}\frac{E_0(\rho)}{\rho}\right]_{\rho=0}^{\infty} + \ln L \int_0^\infty L^{-\rho}\frac{E_0(\rho)}{\rho} d\rho \\
&= \int_0^\infty (\ln L) e^{-(\ln L)\rho}\frac{E_0(\rho)}{\rho} d\rho
\end{aligned} \tag{4.9}$$

Here we have used that

$$\left[\frac{E_0(\rho)}{\rho}\right]_{\rho\to 0} = \left[\frac{\partial E_0(\rho)}{\partial \rho}\right]_{\rho\to 0} = C \quad \text{and} \quad \left[\frac{E_0(\rho)}{\rho}\right]_{\rho\to\infty} = 0.$$



(a) With puncturing (factor $p = 8$)

(b) Without puncturing

Figure 4.3: Comparison of capacity, cutoff rate, predicated rate and simulation results for spinal codes for the AWGN channel. All calculations have been performed using a uniform constellation of size 2^6. The simulation uses $k = 4$ and the M-algorithm at $M = 256$. The predicated rate uses equation (4.7) with $L = 16$. The capacity for non-uniform constellations has been included as a dotted line.

A way to interpret the integral in equation (4.9) is that

$$\mathbb{E}[R] \geq \mathbb{E}_\rho[R_\rho] = \mathbb{E}_\rho\left[\frac{E_0(\rho)}{\rho}\right],$$

where $\mathbb{E}_\rho[\cdot]$ is an expectation over $\rho \sim \text{Exp}(\ln L)$, that is ρ is distributed exponentially with rate parameter $\lambda = \ln L$.

Figures 4.2 and 4.3 show the result of evaluating equation (4.7) for the BSC and AWGN channels. In both cases we compare against capacity, the ordinary cutoff rate and simulation results for spinal codes. We also consider both the cases with and without puncturing.

We can see that for all configurations, we can provide a relatively good prediction of the performance of the spinal code. One notable difference is that the effect of rate quantization is much more pronounced in our prediction than it is in the simulation. For the AWGN channel, spinal codes at $p = 8$ already show a nearly smooth rate curve, while we still predict pronounced plateau effects at high SNR.

One aspect that remains unclear is the choice of L. Figures 4.2 and 4.3 show results for $L = 16$, which has been chosen because it matches the simulation results well. While this calls the value of this rate "prediction" into question, it is notable that a fixed value of L provides good results across two different channel types and a wide range of channel parameters.

Chapter 5

Conclusion

In this thesis, we have investigated various aspects of rateless spinal codes with the goal of improving their rate performance and reducing decoding complexity. Part of our work was to investigate already known aspects in more detail, such as puncturing, tail symbols and error-detection. Our larger contributions are:

- The application of probabilistic shaping to the constellation used by spinal codes, which requires different optimization than for fixed-length codes.
- The investigation of many sequential decoder algorithms and identification of the adaptive effort decoder as a promising candidate.
- The contextualization of achievable rate results for spinal codes, and their reconciliation with sequential decoding bounds.

Ultimately, rateless codes have two main appeals: First, they can automatically adjust to unknown and changing channel conditions. Second, they can approach capacity at shorter message lengths. The question is thus: Can spinal codes live up to these promises, and is their application practical?

While spinal codes are undoubtedly an elegant coding construction, they also suffer from a number of problems, the most significant we consider to be:

- The reduced reliability of the final bits necessitates the use of additional tail symbols or zero termination.
- A CRC or other error-detecting code is used to decide when decoding is successful. Together with the previous point, this may be a significant rate loss at short message lengths.
- Sequential decoding is subject to fundamental limits. While these can be surpassed for variable-length codes, it is to be expected that codes not based on sequential decoding will need less computational effort to approach capacity.

- While we have presented some improvements, the decoding effort for spinal codes remains high. Even though practical decoders are linear in the message size, the decoding effort per bit is considerable, at least if operation at high rates is desired.

Some of these issues may be overcome. An error-detecting code could be avoided, if a sufficiently good reliability estimation can be performed as part of the decoding process. The decoding effort could be reduced further using better heuristics for the abortion of unlikely to succeed decoding attempts. Furthermore it might be possible to reuse parts of the decoder state between decoding attempts. These are possible topics for further research.

However, other problems are more fundamental to the coding construction. As a tree code with a large state size, both the reduced reliability of the tail bits, and the need to use sequential decoding (and thus be subject to its limitations), do not appear to have any straightforward solutions. The methods that are used to avoid these issues for convolutional codes (tail-biting termination and Viterbi decoding) only work for short constraint lengths.

Of course, this does not mean that spinal codes cannot be a useful coding construction—just that the search for good rateless codes is not yet at an end.

Bibliography

[1] J. Anderson and S. Mohan. "Sequential Coding Algorithms: A Survey and Cost Analysis". In: *IEEE Transactions on Communications* 32.2 (1984), pp. 169–176.

[2] E. Arikan. "An inequality on guessing and its application to sequential decoding". In: *IEEE Transactions on Information Theory* 42.1 (1996), pp. 99–105.

[3] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. "Optimal decoding of linear codes for minimizing symbol error rate (Corresp.)" In: *IEEE Transactions on Information Theory* 20.2 (1974), pp. 284–287.

[4] H. Balakrishnan, P. Iannucci, J. Perry, and D. Shah. "De-randomizing Shannon: The Design and Analysis of a Capacity-Achieving Rateless Code". In: *ArXiv e-prints* (June 2012). arXiv: `1206.0418 [cs.IT]`.

[5] G. Böcherer, F. Steiner, and P. Schulte. "Bandwidth Efficient and Rate-Matched Low-Density Parity-Check Coded Modulation". In: *IEEE Transactions on Communications* 63.12 (2015), pp. 4651–4665.

[6] David R Clark. "A Note on the Upper-truncated Pareto distribution". In: *Casualty Actuarial Society E-Forum*. 2013, pp. 1–22.

[7] D. D. Falconer. "A hybrid coding scheme for discrete memoryless channels". In: *The Bell System Technical Journal* 48.3 (1969), pp. 691–728.

[8] R. Fano. "A heuristic discussion of probabilistic decoding". In: *IEEE Transactions on Information Theory* 9.2 (1963), pp. 64–74.

[9] R. Gallager. "A simple derivation of the coding theorem and some applications". In: *IEEE Transactions on Information Theory* 11.1 (1965), pp. 3–18.

[10] Aditya Gudipati and Sachin Katti. "Strider: Automatic Rate Adaptation and Collision Handling". In: *Proceedings of the ACM SIGCOMM 2011 Conference*. SIGCOMM '11. Toronto, Ontario, Canada: ACM, 2011, pp. 158–169.

[11] Peter A. Iannucci, Kermin Elliott Fleming, Jonathan Perry, Hari Balakrishnan, and Devavrat Shah. "A Hardware Spinal Decoder". In: *Proceedings of the Eighth ACM/IEEE Symposium on Architectures for Networking and Communications Systems*. ANCS '12. Austin, Texas, USA: ACM, 2012, pp. 151–162.

[12] Peter Anthony Iannucci, Jonathan Perry, Hari Balakrishnan, and Devavrat Shah. "No Symbol Left Behind: A Link-layer Protocol for Rateless Codes". In: *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*. Mobicom '12. Istanbul, Turkey: ACM, 2012, pp. 17–28.

[13] Jana Iyengar and Martin Thomson. *QUIC: A UDP-Based Multiplexed and Secure Transport*. Internet-Draft draft-ietf-quic-transport-13. Work in Progress. Internet Engineering Task Force, June 2018. 120 pp.

[14] I. Jacobs and E. Berlekamp. "A lower bound to the distribution of computation for sequential decoding". In: *IEEE Transactions on Information Theory* 13.2 (1967), pp. 167–174.

[15] F. Jelinek. "An upper bound on moments of sequential decoding effort". In: *IEEE Transactions on Information Theory* 15.1 (1969), pp. 140–149.

[16] F. Jelinek. "Fast Sequential Decoding Algorithm Using a Stack". In: *IBM Journal of Research and Development* 13.6 (1969), pp. 675–685.

[17] Z. Li, R. Liu, R. Duan, and Y. Hou. "The truncated transmission of spinal codes with imperfect feedback in block-fading channel". In: *2015 International Conference on Wireless Communications Signal Processing (WCSP)*. 2015, pp. 1–5.

[18] Michael Luby. "LT Codes". In: *Proceedings of the 43rd Symposium on Foundations of Computer Science*. FOCS '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 271–280.

[19] J. Massey. "Variable-length codes and the Fano metric". In: *IEEE Transactions on Information Theory* 18.1 (1972), pp. 196–198.

[20] R. Palanki and J. S. Yedidia. "Rateless codes on noisy channels". In: *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.* 2004, pp. 37–.

[21] Jonathan Perry, Hari Balakrishnan, and Devavrat Shah. "Rateless Spinal Codes". In: *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*. HotNets-X. Cambridge, Massachusetts: ACM, 2011, 6:1–6:6.

[22] Jonathan Perry, Peter A. Iannucci, Kermin E. Fleming, Hari Balakrishnan, and Devavrat Shah. "Spinal Codes". In: *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. SIGCOMM '12. Helsinki, Finland: ACM, 2012, pp. 49–60.

[23] Y. Polyanskiy, H. V. Poor, and S. Verdu. "Channel Coding Rate in the Finite Blocklength Regime". In: *IEEE Transactions on Information Theory* 56.5 (2010), pp. 2307–2359.

[24] Y. Polyanskiy, H. V. Poor, and S. Verdu. "Feedback in the Non-Asymptotic Regime". In: *IEEE Transactions on Information Theory* 57.8 (2011), pp. 4903–4925.

[25] A. R. Raghavan and C. W. Baum. "A reliability output Viterbi algorithm with applications to hybrid ARQ". In: *IEEE Transactions on Information Theory* 44.3 (1998), pp. 1214–1216.

[26] D. N. Rowitch and L. B. Milstein. “On the performance of hybrid FEC/ARQ systems using rate compatible punctured turbo (RCPT) codes”. In: *IEEE Transactions on Communications* 48.6 (2000), pp. 948–959.

[27] J. E. Savage. “Sequential decoding – the computation problem”. In: *The Bell System Technical Journal* 45.1 (1966), pp. 149–175.

[28] S. Sesia, G. Caire, and G. Vivier. “Incremental redundancy hybrid ARQ schemes based on low-density parity-check codes”. In: *IEEE Transactions on Communications* 52.8 (2004), pp. 1311–1321.

[29] C.E. Shannon, R.G. Gallager, and E.R. Berlekamp. “Lower bounds to error probability for coding on discrete memoryless channels. Part I”. In: *Information and Control* 10.1 (1967), pp. 65 –103.

[30] Amin Shokrollahi. “Raptor Codes”. In: *IEEE/ACM Trans. Netw.* 14.SI (June 2006), pp. 2551–2567.

[31] S. J. Simmons. “Breadth-first trellis decoding with adaptive effort”. In: *IEEE Transactions on Communications* 38.1 (1990), pp. 3–12.

[32] A. Valembois and M. P. C. Fossorier. “Sphere-packing bounds revisited for moderate block lengths”. In: *IEEE Transactions on Information Theory* 50.12 (2004), pp. 2998–3014.

[33] S. Verdu and S. Shamai. “Variable-Rate Channel Capacity”. In: *IEEE Transactions on Information Theory* 56.6 (2010), pp. 2651–2667.

[34] A. Viterbi. “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm”. In: *IEEE Transactions on Information Theory* 13.2 (1967), pp. 260–269.

[35] Liangliang Wang, Yin Sun, Xuming Lu, Xiang Chen, and Hongzhou Tan. “Spinal Codes Based Efficient Rateless Transmission Scheme for Massive MIMO System with Imperfect CSI”. In: *Communications and Networking*. Ed. by Bo Li, Lei Shu, and Deze Zeng. Cham: Springer International Publishing, 2018, pp. 64–74.

[36] A. R. Williamson, T. Chen, and R. D. Wesel. “Variable-Length Convolutional Coding for Short Blocklengths With Decision Feedback”. In: *IEEE Transactions on Communications* 63.7 (2015), pp. 2389–2403.

[37] A. R. Williamson, M. J. Marshall, and R. D. Wesel. “Reliability-Output Decoding of Tail-Biting Convolutional Codes”. In: *IEEE Transactions on Communications* 62.6 (2014), pp. 1768–1778.

[38] John M Wozencraft. *Sequential decoding for reliable communication*. Tech. rep. Research Laboratory of Electronics, Massachusetts Institute of Technology, 1957.

[39] S. Xu, S. Wu, J. Luo, J. Jiao, and Q. Zhang. “Low Complexity Decoding for Spinal Codes: Sliding Feedback Decoding”. In: *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*. 2017, pp. 1–5.

[40] W. Yang, Y. Li, and X. Yu. “Performance of spinal codes with sliding window decoding”. In: *2017 IEEE International Symposium on Information Theory (ISIT)*. 2017, pp. 2203–2207.

[41] W. Yang, Y. Li, X. Yu, and J. Li. “A Low Complexity Sequential Decoding Algorithm for Rateless Spinal Codes”. In: *IEEE Communications Letters* 19.7 (2015), pp. 1105–1108.

[42] W. Yang, Y. Li, X. Yu, and Y. Sun. “Rateless Superposition Spinal Coding Scheme for Half-Duplex Relay Channel”. In: *IEEE Transactions on Wireless Communications* 15.9 (2016), pp. 6259–6272.

[43] W. Yang, Y. Li, X. Yu, and Y. Sun. “Two-way spinal codes”. In: *2016 IEEE International Symposium on Information Theory (ISIT)*. 2016, pp. 1919–1923.

[44] X. Yu, Ying Li, and Weiqiang Yang. “Rateless spinal code for decode-and-forward relay channel”. In: *2015 International Workshop on High Mobility Wireless Communications (HMWC)*. 2015, pp. 71–75.

[45] X. Yu, Y. Li, W. Yang, and Y. Sun. “Design and Analysis of Unequal Error Protection Rateless Spinal Codes”. In: *IEEE Transactions on Communications* 64.11 (2016), pp. 4461–4473.

[46] K. Sh. Zigangirov. “Some sequential decoding procedures”. In: *Problemy Peredachi Informatsii* 2.4 (1966), pp. 13–25.

Appendix A

Simulation Parameters

The spinal coding simulations presented in various figures have too many parameters to be listed inside figure captions. Instead this appendix provides a hopefully exhaustive listing of the used simulation parameters. The following parameters are defaults, which are used if not otherwise mentioned:

- Message length $n = 256$
- Message fragment size $k = 4$
- Constellation size $2^c = 2^6$
- Tail symbols $T = 2$
- If tail zero bits $t \neq 0$ given, then $T = 1$
- Puncturing factor $p = 8$
- AWGN with SNR from 0dB to 30dB in increments of 1dB
- If truncated Gaussian constellation: Truncation parameter $\beta = 2$
- If probabilistically shaped: Optimized for $R_{\mathrm{VLFT}}(L = 16)$
- If stack decoder: Stack size $S = 512$
- If M-algorithm: Beam width $M = 256$
- If CRC: Given as bit-reversed polynomial in implicit +1 notation

Apart from these defaults, the figures have been generated with the following parameters:

Figure 2.2 right: Stack decoder. Uniform constellation, truncated Gaussian constellation, probabilistically shaped constellation.
Figure 2.3 (d): Stack decoder. Probabilistically shaped constellation with Δ varied between 0.027067 and 0.14 using variable-sized increments.
Figure 2.4: Stack decoder. Uniform constellation. Interleaved, backward and forward puncturing patterns with $p = 1, 2, 4, 8, 16, 64$.
Figure 2.6: Stack decoder. Uniform constellation. Tail symbols T varied between 1 and 4

in increments of 0.2.
Figure 2.7: Stack decoder. Uniform constellation. Tail zero bits t varied between 0 and 16 in increments of 1.
Figure 2.8 left: BSC with crossover probability $\delta = 0.11$. Stack decoder. Tail zero bits $t = 6$. Message sizes $n = 24, 32, 48, 64, 96, 128, 192, 256, 512$ including tail zero bits and CRC bits. Puncturing factor $p = n/k$ (maximal). 13-bit CRC polynomial 1fd5.
Figure 2.8 right: AWGN with SNR 10dB. M-algorithm and stack decoder. Uniform constellation. Tail zero bits $t = 6$. Message sizes $n = 32, 64, 128, 256$. CRC polynomials: 9-bit: 1be, 10-bit: 3c9, 11-bit: 7ef, 12-bit: fdd.
Figure 3.1: AWGN with SNR 0dB, 10dB, 20dB. M-algorithm with beam width $M = 2^0$ to 2^{12}. Stack decoder with stack size $S = 2^0$ to 2^{12}. Probabilistic shaping.
Figure 3.2: AWGN with SNR 0dB, 10dB, 20dB. Adaptive effort decoder with threshold $T = 20\sigma^2$, maximum number of threshold reductions $r = 10$ and beam width $M = 2^0$ to 2^{12}. Stack decoder with stack size $S = 2^0$ to 2^{12}. Probabilistic shaping.
Figure 3.3: M-algorithm with $M = 128$. Stack decoder with $S = 512$. Adaptive effort decoder with $T = 20\sigma^2$, $r = 10$ and $M = 512$. Probabilistic shaping.
Figure 4.2: BSC with crossover probability δ between 0.002 and 0.3 with variable increments. M-algorithm. Left: Puncturing factor $p = 8$. Right: $p = 1$.
Figure 4.3: M-algorithm. Uniform constellation. Left: Puncturing factor $p = 8$. Right: $p = 1$.