DIPLOMARBEIT

INFORMATION-GEOMETRIC ANALYSIS OF ITERATIVE RECEIVER STRUCTURES

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Diplomingenieurs

unter der Leitung von

Ao. Univ.Prof. Dipl.-Ing. Dr. Gerald Matz

Univ.Ass. Dipl.-Ing. Clemens Novak

Institut für Nachrichtentechnik und Hochfrequenztechnik

eingereicht an der Technischen Universität Wien

Fakultät für Elektrotechnik

von

Stefan Schwandter

Kudlichgasse 19/11

1100 Wien

Wien, im Jänner 2008 _____

Abstract

This thesis gives an introduction to basic concepts of information geometry, a theory that uses notions of differential geometry to describe information and probabilities. These concepts are applied to the analytic description of an iterative receiver for bit interleaved coded modulation (BICM). The operation of the receiver's subblocks are recognized as implicit projections of input probability distributions on certain sub-manifolds, which for example represent the code. Furthermore, the effects of log-likelihood ratio (LLR) clipping onto the receiver performance are studied using information geometry. Again, clipping has an interpretation as a projection. Finally numerical simulations of a BICM system show the performance improvements due to iterative decoding and the effects of LLR clipping.

Kurzfassung

Diese Diplomarbeit gibt eine Einführung in bestimmte Aspekte der Informationsgeometrie, eine Theorie, in der Konzepte der Differentialgeometrie für eine Beschreibung von Information und Wahrscheinlichkeiten herangezogen werden. Die vorgestellten Konzepte werden für die analytische Beschreibung eines iterativen Empfängers für "bit interleaved coded modulation" (BICM) verwendet. Die Arbeitsweise der Teilblöcke aus denen der Empfänger aufgebaut ist wird als implizite Projektion von Eingangs-Wahrscheinlichkeitsdichtefunktionen auf bestimmte Teilmannigfaltgkeiten, die z.B. den Code repräsentieren, interpretiert. Weiters werden die Auswirkungen von "log-likelihood (LLR) clipping" auf die Leistungsfähigkeit des Empfängers mithilfe der Informationsgeometrie untersucht. Auch hier taugt eine informationsgeometrische Projektion zur Beschreibung des zu untersuchenden Sachverhalts. Schlussendlich zeigen numerische Simulationen die Leistungssteigerung durch das iterative Dekodierverfahren sowie die Auswirkungen des LLR clipping.

Acknowledgements

I hereby wish to thank the people who have been continuously encouraging and helping me over the years of my studies and especially during the time of writing this thesis:

My friends and colleagues, my thesis advisors, and my family.

Contents

1

Introduction

Soft-decoding has been recognized to have performance advantages above hard-decoding. It furthermore enables the use of iterative (turbo) decoders [1], which consist of sub-blocks that exchange soft information about the transmitted bits to improve the decoding performance. The idea is that each block does a local optimization that converges to a global one over the iterations.

Bit interleaved coded modulation (BICM) [2, 3] is a transmission system that uses interleaving on a bit level to reduce the impact of burst errors. It is therefore very applicable to wireless communication channels where it can combat the effects of fading on the receiver performance. For maximum likelihood detection of BICM with a soft decoding algorithm it is mandatory to split up the sequence probabilities into bit probabilities, which is an approximation that leads to sub-optimal performance. This performance penalty can be reduced by the use of an iterative demodulation method [4].

Soft information can be represented in the form of probabilities, or, equivalently, log-likelihood ratios (LLRs). The latter are often more convenient for the use in practical implementations. Restricting their value range generally means a loss of information, and therefore can lead to decreased performance. However, it can be desirable to restrict

LLR values ("LLR clipping") to reduce computational complexity in implementations [5].

Information geometry is a relatively young mathematical field that applies differential-geometric concepts to information theory and statistics. Families of probability distributions are considered as manifolds, on which for example distance measures can be defined. Many important statistical concepts can be interpreted inside this framework.

This text contains an introduction to information geometry and an information-geometric interpretation of BICM decoding and LLR clipping. It is structured as follows:

- **Chapter 1**: This introduction.
- **Chapter 2** introduces several concepts of information geometry that will be used in the later chapters.
- **Chapter 3**: A general information-geometric interpretation of soft-sequence decoding is given, which then serves as the theoretical ground for the interpretation of a practical iterative decoding algorithm for bit interleaved coded modulation.
- **Chapter 4** tackles the problem of log-likelihood ratio clipping. A geometrical interpretation of LLR-clipping is given that could be used for analytically assessing the resulting performance loss.
- **Chapter 5** shows numerical simulation results of iterative demodulation and LLR clipping in a BICM system.
- **Chapter 6** gives a summary and discusses several open points that could be tackled in future research.

2

Information Geometry

2.1 Introduction

Information geometry is a mathematical theory that considers the geometric properties of information and probability, using concepts of differential geometry. It is a relatively young theory that reached its maturity only recently with the work of Shun'ichi Amari in the 1980s.

We will only introduce the notions that are essential for the purpose of describing iterative soft-decoding. Although the probability distributions considered in information geometry are generally distributions of continuous random variables (probability density functions), we will restrict ourselves to distributions of *discrete* random variables (probability mass functions, briefly called "pmfs"), and use the words distribution and pmf synonymously unless stated otherwise. This chapter is mostly based on [6] and to a lesser extent on [7].

2.2 Basics

The first important idea we consider is the concept of a family of probability distributions. A family is a set of distributions that depend on some parameters but apart from that have the same structure. The parameters can be regarded as coordinates of a point in a manifold. In this way, the family is a manifold in which one specific distribution is a point specified by its coordinates.

2.2.1 Exponential Families

Definition 2.2.1. An *exponential family* of discrete probability distributions $p(x)$ on an alphabet $\mathcal{X}$ is the set

$$\mathcal{E} = \left\{ p : p(x) = c \cdot q(x) \exp\left(\sum_1^K \theta_i f_i(x) \right) \right\}. \tag{2.1}$$

Here, $q(x)$ is a given distribution and the normalization constant c is

$$c = c(\theta_1, \ldots, \theta_k) = \left(\sum_a q(x) \exp\left(\sum_1^K \theta_i f_i(x) \right) \right)^{-1}.$$

Furthermore, $f_1, f_2, \ldots, f_K$ are some given functions on $\mathcal{X}$. The set of all these functions, $\{f_i(x)\}_1^K$ is called "sufficient statistics".

The distribution $q(x)$ is itself an element of the exponential family that is "built around" it, as can be seen by setting $\theta_1 = \theta_2 = \ldots = 0$. Any element of $\mathcal{E}$ could play the role of $q(x)$, but if it is necessary to emphasize the dependency of $\mathcal{E}$ on $q(x)$, we will write $\mathcal{E}_q$.

The set $\mathcal{E}$ can be regarded as a K-dimensional manifold. The numbers $\theta_1, \theta_2, \ldots, \theta_K$ can be collected in a vector $\boldsymbol{\theta}$. This vector plays the role of a coordinate system in $\mathcal{E}$, so that each distribution (which is now seen as a point in the manifold $\mathcal{E}$) is specified by one $\boldsymbol{\theta}$. This particular coordinate system, although being one out of many

possible coordinate systems, is given the name "natural" or "canonical" parameter for the exponential family.

The distributions in an exponential family have the following important property: Given two pmfs $p_1, p_2 \in \mathcal{E}$, with natural parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, respectively, the normalized log-convex combination $p(x;t) = \frac{p_1^{(1-t)}(x)p_2^t(x)}{c(t)}$ of the two is again a pmf in the same exponential family. For $0 \leq t \leq 1$,

$$\begin{aligned}
\log p(x;t) &= (1-t)\log p_1(x) + t\log p_2(x) - \log c(t) \\
&= (1-t)[\log c_1 + \log q(x) + \sum_i \theta_{i,1} f_i(x)] + \\
&+ t[\log c_2 + \log q(x) + \sum_i \theta_{i,2} f_i(x)] - \log c(t) \\
&= \log c_t + \log q(x) + \sum_i [(1-t)\theta_{i,1} + t\theta_{i,2}] f_i(x)
\end{aligned}$$

or

$$p(x;t) = c_t \cdot q(x) \exp\left(\sum_i [(1-t)\theta_{i,1} + t\theta_{i,2}] f_i(x)\right),$$

with $c_t = \frac{c_1^{(1-t)} c_2^t}{c(t)}$. For the natural coordinates of the set of resulting distributions, this means

$$\boldsymbol{\theta}_t = (1-t)\boldsymbol{\theta}_1 + t\boldsymbol{\theta}_2.$$

The normalized log-convex combination $p(x;t)$ is a curve connecting p_1 and p_2, parametrized by t. This curve is obviously itself an exponential family and a one-dimensional submanifold of $\mathcal{E}$. It appears as a "straight line" in the natural coordinates $\boldsymbol{\theta}$ and is thus called an "exponential (or e-)geodesic".

In [8], the exponential geodesic is called "exponential segment", and an alternative definition for an exponential family is given:

Definition 2.2.2. An exponential family is a set of probability distributions with the characteristic property that it contains the exponential segment between any two of its members.

An exponential family is not a closed set in general. We will discuss this fact in the context of example 2.2.2. In the following, the closure is referred to as $\mathrm{cl}\{\mathcal{E}\}$.

Example 2.2.1. An example for an exponential family that we will consider in a later chapter in the context of soft-decoding is the family of *factorizable densities*:

$$\mathcal{E}_{\mathcal{F}} = \Big\{ p(\mathbf{x}) \colon \; p(\mathbf{x}) = \prod_{i=1}^{K} p_i(x_i) \Big\}.$$

Here $\mathbf{x} \in \mathcal{X}^K$ is a vector or sequence of length K containing the individual bits $x_i \in \mathcal{X}$, $\mathcal{X} = \{0, 1\}$, $i = 1, \ldots, K$. The sufficient statistics and natural coordinates are in this case $f_i(\mathbf{x}) = x_i$ and $\theta_i = \log \frac{p_i(x_i=1)}{1-p_i(x_i=1)}$, respectively.

We will use the aforementioned alternative definition to show that it is indeed an exponential family:

$$\begin{aligned}
\log p(\mathbf{x}; t) &= (1-t) \log p_1 + t \log p_2 - \log c(t) \\
&= (1-t) \sum_i \log p_{1,i}(x_i) + t \sum_i \log p_{2,i}(x_i) - \log c(t) \\
&= \sum_i (1-t) \log p_{1,i}(x_i) + t \log p_{2,i}(x_i) - \log c(t) \\
&= \sum_i \log p_{1,i}(x_i)^{(1-t)} + \log p_{2,i}(x_i)^t - \log c(t) \\
&= \sum_i \log \underbrace{p_{1,i}(x_i)^{(1-t)} p_{2,i}(x_i)^t}_{p_{t,i}(x_i)} - \log c(t) = \sum_i \log p_{t,i}(x_i) - \log c(t) \\
&= \log \prod_i p_{t,i}(x_i) - \log c(t).
\end{aligned}$$

We can write $p(\mathbf{x})$ as

$$p(\mathbf{x}; \boldsymbol{\theta}) = c(\boldsymbol{\theta}) \cdot \exp\Big\{ \sum_i \theta_i f_i(\mathbf{x}) \Big\} = \prod_i \frac{1}{1 + e^{\theta_i}} \cdot \exp\Big\{ \log \frac{p_i(x_i = 1)}{1 - p_i(x_i = 1)} \cdot x_i \Big\}.$$

Example 2.2.2. Finally we show that the set of discrete distributions $p(x)$, where $x \in \mathcal{X} = \{0, 1, \ldots, N\}$, is by itself an exponential family, without further restrictions

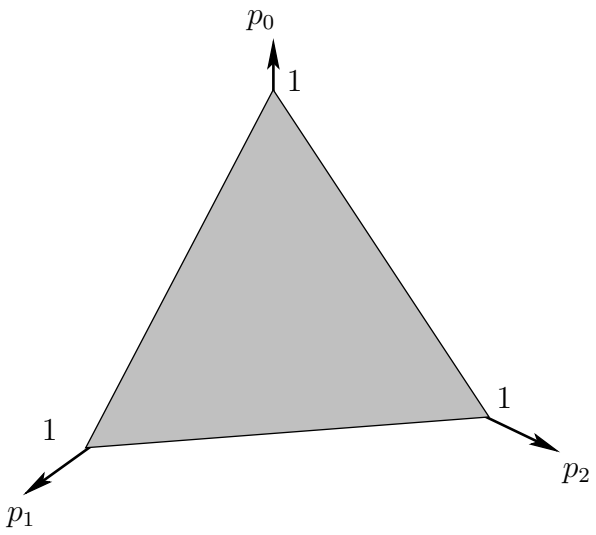

Figure 2.1: Manifold of discrete probability distributions.

on its structure.

Let x be a discrete random variable taking values on the set $\mathcal{X} = 0, 1, \ldots, N$. Let $\mathbf{p} = (p_0, p_1, \ldots, p_N)$ be the probability vector containing the probabilities of each outcome x, $p_i = \mathrm{P}\{x = i\}$. Then any distribution on $\mathcal{X}$ corresponds to one vector $\mathbf{p}$ whose components satisfy $\sum_{i=0}^{N} p_i = 1, p_i > 0$. Therefore the set of all discrete distributions $\mathcal{E}_{\mathcal{D}} = \{\mathbf{p}\}$ over $\mathcal{X}$ is an N-dimensional manifold.[1] For example, if $N = 2$, the manifold is a triangle, i.e., a two-dimensional simplex, in $\mathbb{R}^3$, as shown in Fig. 2.1.

Let's show that this is indeed an exponential family! By putting

$$\theta_i = \log \tfrac{p_i}{p_0}, \qquad i = 1, \ldots, N,$$
$$f_i(x) = \delta_i(x), \quad i = 1, \ldots, N,$$

where $\delta_i(x) = 1$ when $x = i$ and $\delta_i = 0$ otherwise, we can write $p(x)$ in the form (2.1) of an exponential family:

$$p(x; \boldsymbol{\theta}) = c \cdot q(x) \exp\Big\{\sum_i \theta_i f_i(x)\Big\},$$

[1]Note that $\mathbf{p}$ has $N + 1$ elements. The manifold S only has N dimensions because due to the condition $\sum_i p_i = 1, p_i > 0$, N probabilities fully determine the remaining one.

with $q(x) = \frac{1}{N+1}$ and $c = (N+1)p_0$. Here, c and $q(x)$ have been chosen such that $c \cdot q(x) = p_0$ and $\sum_x q(x) = 1$. Indeed,

$$p(x;\boldsymbol{\theta}) = (N+1) \cdot p_0 \cdot \frac{1}{N+1} \cdot \exp\left\{\sum_i \delta_i(x) \log \frac{p_i}{p_0}\right\} = p_0 \cdot \exp\left\{\log \frac{p_x}{p_0}\right\} = p_x.$$

As already mentioned, a exponential family need not be a closed set. In the present example, e.g. the corner points of the triangle that represents the exponential family in the figure are not included in the exponential family. Consider the point corresponding to the distribution $\{p_1 = 1, p_0 = p_2 = 0\}$. As the $\boldsymbol{\theta}$-coordinates are defined as $\theta_i = \frac{p_i}{p_0}$, and $p_0 = 0$, it is obvious that we run into troubles here.

2.2.2 Linear Families

Definition 2.2.3. A *linear family* of probability distributions on $\mathcal{A}$ is the set

$$\mathcal{L} = \left\{p : \sum_x f_i(x)p(x) = \alpha_i, \quad 1 \le i \le K\right\}, \tag{2.2}$$

for any given functions $f_1, f_2, \ldots, f_K$ on $\mathcal{A}$ and corresponding numbers $\alpha_1, \alpha_2, \ldots, \alpha_K$.

In other words, the expected value $\mathbb{E}_p\{f_i(x)\} = \sum_x f_i(x)p(x)$ of the random variable x with respect to the distribution $p(x)$ is restricted to α_i. We can stack the α_i in a vector $\boldsymbol{\alpha}$, the $f_i(x)$ in a vector $\mathbf{f}(x)$, and (2.2) can be written as

$$\mathcal{L} = \{p : \mathbb{E}_p\{\mathbf{f}(x)\} = \boldsymbol{\alpha}\}.$$

In the following, we will often prefer to use this brief notation.

The vector $\boldsymbol{\alpha}$ serves as a coordinate system in the manifold of the linear family. These coordinates are called "mixture coordinates".

A linear family has a characteristic property, analogue to an exponential family: the convex combination[2] of two of its probability distributions is again a distribution

[2]Note that we now use the convex combination instead of the log-convex combination

contained in the linear family. Given $p_1(x), p_2(x) \in \mathcal{L}$, and $0 \leq t \leq 1$, there is

$$p(x;t) = (1-t)p_1(x) + tp_2(x) \in \mathcal{L}.$$

Indeed, as $\mathbb{E}_{p_1}\{\mathbf{f}(x)\} = \boldsymbol{\alpha}$ and $\mathbb{E}_{p_2}\{\mathbf{f}(x)\} = \boldsymbol{\alpha}$, we have

$$\mathbb{E}_{p(x;t)}\{\mathbf{f}(x)\} = \mathbb{E}_{(1-t)p_1+tp_2}\{\mathbf{f}(x)\} = (1-t)\boldsymbol{\alpha} + t\boldsymbol{\alpha}.$$

This means, the convex combination $p(x;t)$ of p_1 and p_2 satisfies the condition

$$\mathbb{E}_{p(x;t)}\{\mathbf{f}(x)\} = \boldsymbol{\alpha},$$

and therefore belongs to the linear family.

In analogy to the section about exponential families we can state the following: $p(x;t)$ is a curve parametrized by t. It is a linear family and a one-dimensional submanifold of $\mathcal{L}$. This one-dimensional linear family appears as a "straight line" in the mixture coordinates $\boldsymbol{\alpha}$, it is called a "linear (or m-)geodesic".

The above mentioned property of linear families can be used as an alternative definition for a linear family. If we call the m-geodesic "linear segment" to be consistent to the previous section about exponential families, we can give the following

Definition 2.2.4. A linear family is a set of probability distributions with the characteristic property that it contains the linear segment between any two of its members.

A linear family of probability distributions is a closed set. This will become clear in example 2.2.4.

Example 2.2.3. A linear family that will be important in the context of soft-decoding is the "family of code-compatible distributions":

$$\mathcal{L}_{\mathcal{C}} = \Big\{ p(\mathbf{x}) \colon \mathbb{E}_p\big\{\mathbf{f}(\mathbf{x})\big\} = \mathbf{0} \Big\}$$

Here, the functions $\mathbf{f} = (f_1, f_2, \ldots, f_F)$ represent the code structure: $f_i(\mathbf{x}) = 0$ are the check equations. This linear family will be discussed in detail in section 3.4.2.

Example 2.2.4. The family of discrete probability distributions which we dealt with in Example 2.2.2, where we regarded it as an exponential family, can also be seen as a linear family. With the same sufficient statistics $f_i(x) = \delta_i(x)$ we have

$$\mathbb{E}_p\{f_i(x)\} = \sum_x p(x)\delta_i(x) = p(i) = p_i = \alpha_i.$$

We see, that the mixture coordinates α_i are in fact the probabilities of the outcomes i. It is this mixture coordinate system that is used in Fig. 2.1.

Furthermore, looking at the same figure, we see that the linear family is a closed set. The borders of the simplex are of course included - for example, the corner point $\boldsymbol{\alpha} = (1, 0, 0)$.

2.2.3 Kullback-Leibler Divergence

In information geometry, the Kullback-Leibler (KL) divergence serves as a distance measure between two probability distributions.

Definition 2.2.5. For two probability mass functions p and q on $\mathcal{X}$, the *KL divergence* is given by

$$D(p\|q) = \sum_{a\in\mathcal{A}} p(x) \log \frac{p(x)}{q(x)}. \tag{2.3}$$

It has properties that are those of a distance:

$$\begin{aligned} D(p\|q) &\geq 0, \\ D(p\|q) &= 0 \text{ if and only if } p = q. \end{aligned}$$

Note however that $D(p\|q) \neq D(q\|q)$, that's why it is not a metric in the strict sense. In fact, KL divergence can be regarded as a nonsymmetric analogue of a squared

Euclidean distance. This will become clear in section 2.2.4.

The KL divergence is not the only possible distance measure for statistical manifolds. It can be regarded as a special case of the so-called *f-divergence* [9]

$$D_f(p\|q) = \sum_x q(x) f\left(\frac{p(x)}{q(x)}\right),$$

for $f(t) = t \log t$ ($f(t)$ has to be a convex function in general).

Example 2.2.5. In the context of sequence log-likelihood ratio clipping, the KL divergence of a distribution and the uniform distribution is important. Let's calculate the KL divergences of the following distribution $p = (0.1, 0.2, 0.4, 0.3)$ from the uniform distribution $\bar{p} = (1/4, 1/4, 1/4, 1/4)$:

$$\begin{aligned}
D(\bar{p}\|p) &= \sum_{i=1}^{4} \bar{p}_i \log \frac{\bar{p}_i}{p_i} \\
&= 0.25 \times (\log(2.5) + \log(1.25) + \log(0.625) + \log(0.833) \\
&= 0.25 \times (1.32 + 0.32 - 0.678 - 0.263) \\
&= 0.176 \text{ bits}, \\
D(p\|\bar{p}) &= \sum_{i=1}^{4} p_i \log \frac{p_i}{\bar{p}_i} \\
&= 0.1 \times \log \frac{0.1}{0.25} + 0.2 \times \log \frac{0.2}{0.25} + 0.4 \times \log \frac{0.4}{0.25} + 0.3 \times \log \frac{0.3}{0.25} \\
&= -0.132 - 0.064 + 0.271 + 0.079 \\
&= 0.154 \text{ bits}.
\end{aligned}$$

Base 2 logarithms have been used, therefore the unit is bits.

Example 2.2.6. We will now show that the KL divergence of a factorizable distribution (see example 2.2.7) is the sum of "partial" KL divergences, for simplicity for the case of two distributions consisting of two marginal pmfs, $p(\mathbf{x}) = p_1(x_1)p_2(x_2)$ and $q(\mathbf{x}) =$

$q_1(x_1)q_2(x_2)$:

$$\begin{aligned}
D(p\|q) &= \sum_{\mathbf{x}} p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} \\
&= \sum_{x_1}\sum_{x_2} p_1(x_1)p_2(x_2) \log \frac{p_1(x_1)p_2(x_2)}{q_1(x_1)q_2(x_2)} \\
&= \sum_{x_2} p_2(x_2) \sum_{x_1} p_1(x_1) \left[\log \frac{p_1(x_1)}{q_1(x_1)} + \log \frac{p_2(x_2)}{q_2(x_2)}\right] \\
&= \underbrace{\sum_{x_2} p_2(x_2)}_{=1} \sum_{x_1} p_1(x_1) \log \frac{p_1(x_1)}{q_1(x_1)} + \sum_{x_2} p_2(x_2) \underbrace{\sum_{x_1} p_1(x_1)}_{=1} \log \frac{p_2(x_2)}{q_2(x_2)} \\
&= D(p_1\|q_1) + D(p_2\|q_2).
\end{aligned}$$

In general, one can write for factorizable distributions $p(\mathbf{x}) = \prod_i p_i(x_i)$ and $q(\mathbf{x}) = \prod_i q_i(x_i)$,

$$D(p\|q) = \sum_i D(p_i\|q_i). \tag{2.4}$$

2.2.4 Projections

Analog to the familiar orthogonal projection in Euclidean geometry, it is possible to define projections in information geometry. A necessary prerequisite is some kind of distance measure, like the KL divergence we have just introduced.

I-projection

Fig. 2.2 shows the so-called *I-projection*. Given some distribution q in an exponential family $\mathcal{M}$ one looks for the distribution p^* in the submanifold Π that is closest to p in terms of KL divergence.[3]

Definition 2.2.6. The *I-projection* of a distribution q onto a closed convex[4] subset Π

[3]The I-projection is often called "e-projection", because the curve connecting q and p^* is an e-geodesic (see for example [7])

[4]A set $\mathcal{C}$ is *convex* if the line segment between any two points in C lies in c. More precisely, if for any $q_1, q_2 \in \mathcal{C}$ and any θ with $0 \leq \theta \leq 1$ the convex combination of q_1 and q_2, $q' = \theta q_1 + (1-\theta)q_2 \in C$. [10]

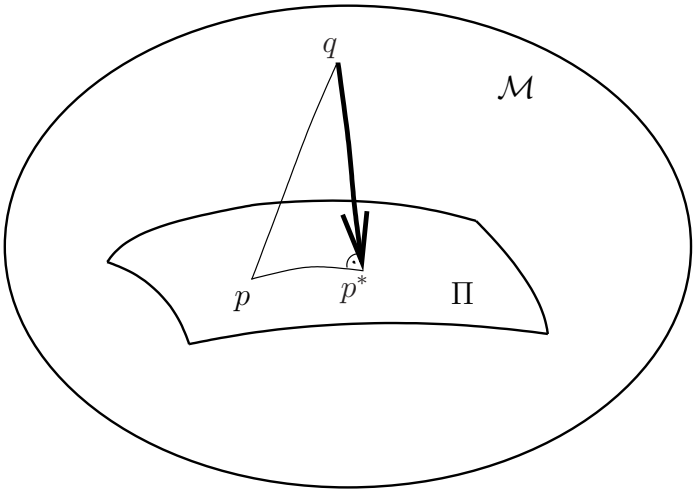

Figure 2.2: I-projection of a distribution q onto a convex submanifold Π.

of distributions on $\mathcal{A}$ is the $p^* \in \Pi$ such that

$$D(p^*\|q) = \min_{p\in\Pi} D(p\|q). \tag{2.5}$$

Supposing that $q(x) > 0$ for all $x \in \mathcal{X}$, the function $D(p\|q)$ is continuous and strictly convex in p, so that p^* exists and is unique (see Fig. 2.3). The *support* of the distribution p is the set $\mathcal{S}(p) = \{x : p(x) > 0\}$. Since Π is convex, among the supports of elements of Π there is one whose support contains all the others. This will be called the support of Π and denoted by $\mathcal{S}(\Pi)$.

For the I-projection there is the following

Theorem 2.2.1. $\mathcal{S}(p^*) = \mathcal{S}(\Pi)$ and $D(p\|q) \geq D(p\|p^*) + D(p^*\|q)$ for all $p \in \Pi$.

This inequality resembles the Pythagorean theorem from Euclidean geometry, considering the interpretation of KL divergence as a sort of squared distance. Fig. 2.4 shows a visualization of the theorem, applied to Euclidean geometry. The convex set Π is represented by a circle (only a sector is shown). p^* is the normal projection of q onto Π. One can easily see that due to the convexity of Π, the angle between the lines pp^* and p^*q must be obtuse, which implies that $l^2_{pq} \geq l^2_{pp^*} + l^2_{p^*q}$ - the same form as

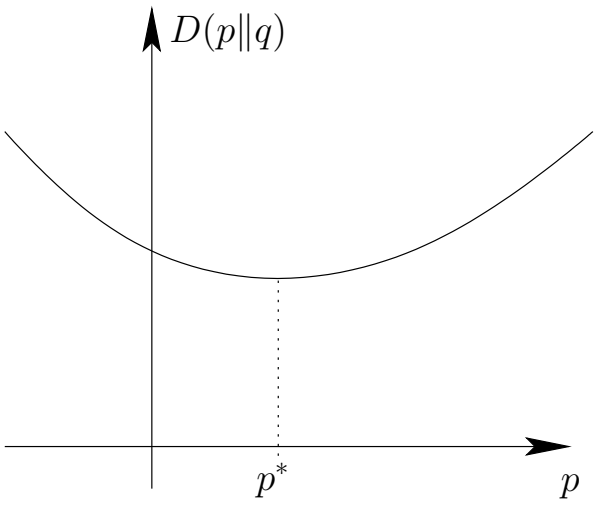

Figure 2.3: Convexity of $D(p\|q)$ as function of p (with I-projection p^*).

the above theorem for the I-projection. This similarity supports the claim that the KL divergence can be of a squared Euclidean distance.

The inequality becomes an equality iff the set we project onto is a linear family:

Theorem 2.2.2. *Pythagorean identity*

The I-projection p^* of q onto a linear family $\mathcal{L}$ satisfies

$$D(p\|q) = D(p\|p^*) + D(p^*\|q), \forall p \in \mathcal{L}.$$

Furthermore, if $\mathcal{S} = \mathcal{A}$ then $\mathcal{L} \cap \mathcal{E}_q = \{p^*\}$.

For a visualization of the above theorem, see figure 2.5. According to [7], the connecting line between q and p^* is itself a (one dimensional) exponential family. It is the exponential family $\mathcal{E}_q$ containing q with the same sufficient statistics f_i as in the linear family $\mathcal{L}$. The intersection point $\mathcal{L} \cap \mathcal{E}_q$ is the I-projection p^*, which means that p^* is of the form (2.1).

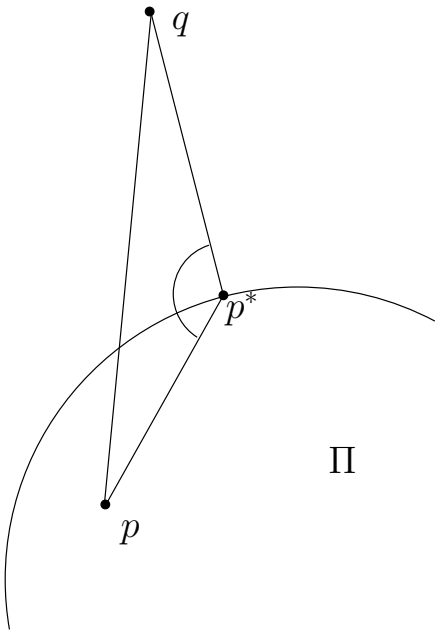

Figure 2.4: Illustration of theorem 2.2.1.

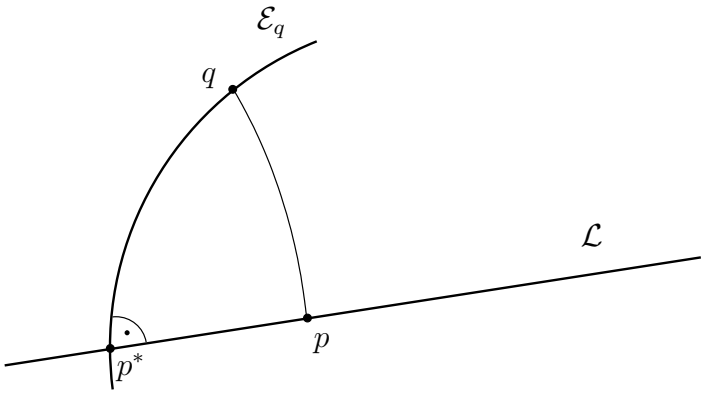

Figure 2.5: I-projection onto a linear family.

Reverse I-Projection

As the KL divergence is nonsymmetric, there are two choices how to define the distance between two distributions p and q. Since we have defined the I-projection as the distribution minimizing the distance $D(p\|q)$ keeping q constant and varying p, there must be a second kind of projection, which is based on the "other" distance between two distributions, in which the arguments of $D(.\|.)$ are reversed. That is called the "reverse I-projection (rI-projection)".

Definition 2.2.7. The *reverse I-projection (rI-projection)*[5] of the distribution p onto a closed, convex subset Π' of distributions on $\mathcal{A}$ is the $q^* \in \Pi'$ such that

$$D(p\|q^*) = \min_{q\in\Pi'} D(p\|q). \tag{2.6}$$

Note that now we minimize over the second argument of the KL divergence, whereas for the I-projection, we minimized over the first argument.

There is again a Pythagorean theorem for the inverse I-projection:

Theorem 2.2.3. The reverse I-projection q^* of the distribution p onto an exponential family $\mathcal{E}$ satisfies the Pythagorean identity

$$D(p\|q) = D(p\|q^*) + D(q^*\|q) \quad \forall q \in \mathcal{E}.$$

We will now show the reverse I-projection of some distribution onto the manifold of factorizable distributions, which we've introduced in Example 2.2.1.

Theorem 2.2.4. Projection onto the set of factorizable distributions

For any distribution p, the rI-projection of p onto

$$\mathcal{E}_{\mathcal{F}} = \Big\{p(\mathbf{x})\colon\ p(\mathbf{x}) = \prod_i p_i(x_i)\Big\},$$

[5]The reverse I-projection is often called "m-projection", because the curve connecting the original distribution and its projection is an m-geodesic.

is the distribution $p_{\mathcal{F}}(\mathbf{x})$ given by the product of the marginal probabilities of $\mathbf{x}$, i.e.:

$$p_{\mathcal{F}}(\mathbf{x}) = \arg\min_{q \in \mathcal{E}_{\mathcal{F}}} D(p\|q) = \prod_i p_i(x_i) = \prod_i \sum_{\mathbf{x} \sim x_i} p(\mathbf{x}').$$

Here, the summation is performed over all elements of $\mathbf{x}$ except x_i.

For any distribution p and for any distribution $r \in \mathcal{E}_{\mathcal{F}}$, we have:

$$D(p\|r) = D(p\|p_{\mathcal{P}}) + D(p_{\mathcal{P}}\|r).$$

Proof. Let us consider a factorizable distribution q and some arbitrary distribution p. Then

$$\begin{aligned} D(p\|q) &= \sum_{\mathbf{x}} p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} = \sum_{\mathbf{x}} p(\mathbf{x}) \left(\log \frac{p(\mathbf{x})}{\prod_i p_i(x_i)} + \log \frac{\prod_i p_i(x_i)}{\prod_i q_i(x_i)} \right) \\ &= D(p\|p_{\mathcal{F}}) + \sum_{\mathbf{x}} p(\mathbf{x}) \sum_i \log \frac{p_i(x_i)}{q_i(x_i)} \\ &= D(p\|p_{\mathcal{F}}) + \sum_i \sum_{\mathbf{x}_i} \underbrace{\sum_{\mathbf{x} \sim x_i} p(\mathbf{x})}_{p_i(x_i)} \log \frac{p_i(x_i)}{q_i(x_i)} \\ &= D(p\|p_{\mathcal{F}}) + \sum_i D(p_i\|q_i). \end{aligned}$$

The left hand term is independent of q while the right hand term is positive for any q and zero for $q_i = p_i$. Hence $D(p\|q)$ is minimum for $q = p_{\mathcal{F}} = \prod_i p_i(x_i)$. □

The family of factorizable distributions is an exponential family, the unique projection onto this subset is therefore an rI-projection, therefore it is no surprise that the Pythagorean theorem (2.2.4) holds.

Example 2.2.7. Let us now consider a distribution for four bit sequences of length 2, $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = (00, 01, 10, 11)$, $p(\mathbf{x}) = (0.1, 0.3, 0.4, 0.2)$. By x_0 and x_1 we denote the first respectively the second bit of a sequence. We have just shown that the reverse I-projection of p onto the factorizable distributions $\mathcal{E}_{\mathcal{F}}$, $p^*(\mathbf{x}) = p_{\mathcal{F}}(\mathbf{x})$, is given by the

product of the marginal distributions. In our case,

$$p_{\mathcal{F}}(\mathbf{x}) = p_0(x_0)p_1(x_1) = \begin{cases} \sum_{\mathbf{x}:x_0=0} p(\mathbf{x}) \sum_{\mathbf{x}:x_1=0} p(\mathbf{x}) = 0.5 \times 0.6 = 0.3 & \mathbf{x} = (00), \\ \sum_{\mathbf{x}:x_0=0} p(\mathbf{x}) \sum_{\mathbf{x}:x_1=1} p(\mathbf{x}) = 0.5 \times 0.4 = 0.2 & \mathbf{x} = (01), \\ \sum_{\mathbf{x}:x_0=1} p(\mathbf{x}) \sum_{\mathbf{x}:x_1=0} p(\mathbf{x}) = 0.5 \times 0.6 = 0.3 & \mathbf{x} = (10), \\ \sum_{\mathbf{x}:x_0=1} p(\mathbf{x}) \sum_{\mathbf{x}:x_1=1} p(\mathbf{x}) = 0.5 \times 0.4 = 0.2 & \mathbf{x} = (11) \end{cases}$$

We notice that a maximum likelihood sequence decoder would have picked sequence 10 as the most probable one given the original pmf, but it would be unable to decide between 00 and 10 knowing only the factorizable distribution, which is a consequence of the loss of information due to the marginalization.

2.3 Iterative Algorithms

In this section we will consider iterative algorithms to minimize the KL divergence between two convex sets of distributions.

2.3.1 Alternating Divergence Minimization

In the following we describe an algorithm that finds the minimum KL divergence between two convex sets $\mathcal{P}$ and $\mathcal{Q}$ of probability distributions on a finite alphabet $\mathcal{X}$. The algorithm is actually also applicable to a more general divergence definition and arbitrary sets, and is presented with proofs in [9], but we will stick to this special case.

Let $\mathcal{P}$ and $\mathcal{Q}$ be convex sets of discrete probability distributions on $\mathcal{X}$, and $D(p\|q)$ the KL divergence. To find the two distributions out of $\mathcal{P}$ and $\mathcal{Q}$ respectively that minimize the KL divergence between the two sets,

$$D_{\min} \triangleq \min_{p\in\mathcal{P},\, q\in\mathcal{Q}} D(p\|q),$$

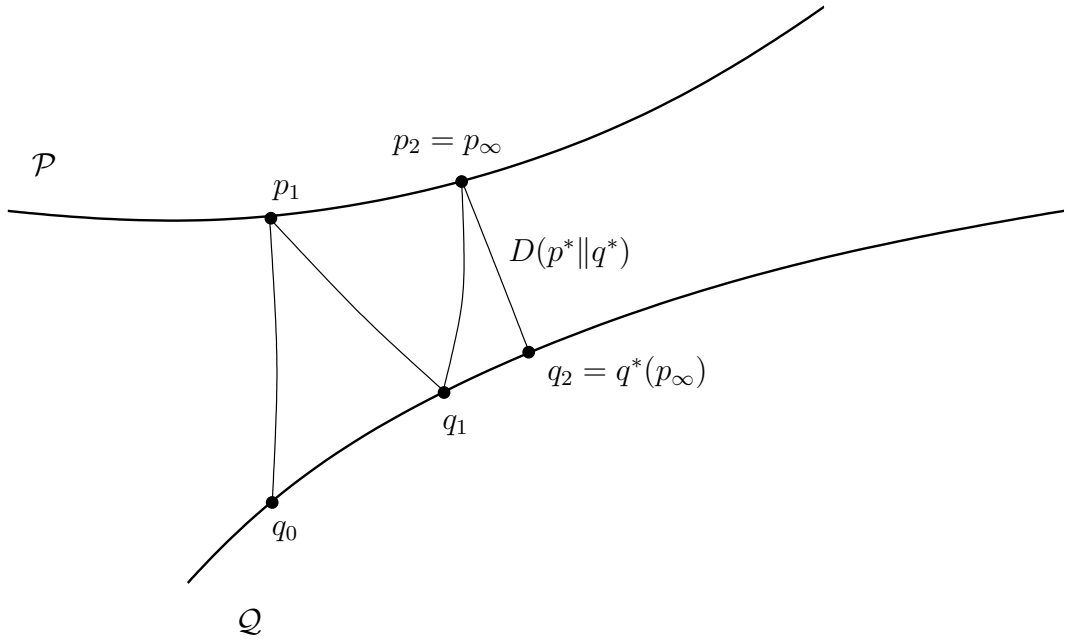

Figure 2.6: Alternating divergence minimization between convex sets $\mathcal{P}$ and $\mathcal{Q}$

there is the following iteration scheme.

Starting from some distribution $q_0 \in \mathcal{Q}$, alternating projections are carried out: $p^*(q_{n-1})$ is the I-projection of q_{n-1} onto $\mathcal{P}$,

$$p_n = p^*(q_{n-1}) = \arg\min_{p\in\mathcal{P}} D(p\|q_{n-1}),$$

and $q_n = q^*(p_n)$ is the reverse I-projection of p_n onto $\mathcal{Q}$,

$$q_n = q^*(p_n) = \arg\min_{q\in\mathcal{Q}} D(p_n\|q).$$

For an illustration of the algorithm, see Fig. 2.6.

According to a general theorem proved in [9], this iteration scheme converges:

$$D(p_n\|q_n) \to D_{\min}, \quad \text{as } n \to \infty$$

If $\mathcal{P}$ is compact, p_n converges to an p_∞, $p \to p_\infty$, such that

$$D(p_\infty \| q^*(p_\infty)) = D_{\min}$$

There is also the inequality

$$D(p_{n+1} \| q_n) - D_{\min} \leq \max_{x \in \mathcal{X}} \log \frac{p_{n+1}(x)}{p_n(x)}, \tag{2.7}$$

which allows to use the right-hand-side of 2.7 as stopping criterion.

2.3.2 The EM Algorithm

The expectation-maximization (EM) algorithm is an often used technique in statistics to estimate a probability distribution, that is supposed to belong to a set of "feasible" distributions $\mathcal{Q}$ on alphabet $\mathcal{X}$, when there is only incomplete sample data available. Instead of the "full" sample $\mathbf{x} = (x_1, x_2, \ldots, x_N) \in \mathcal{X}^N$, only an "incomplete" sample $\mathbf{y} = (y_1, y_2, \ldots, y_N) \in \mathcal{Y}^N$ can be observed. The components y_i are obtained by a known mapping $T : \mathcal{X} \mapsto \mathcal{Y}$, $y_i = Tx_i$. This mapping is normally a many-to-one mapping, so that information that exists in the full sample is lost when only the incomplete sample is observable. One example would be combining several adjacent outcome values to one single value, like in a histogram.

The EM algorithm produces a sequence of distributions $q_k \in \mathcal{Q}$, that are regarded as consecutively improved estimates of the unknown distribution, by iterating the following two steps, starting with an arbitrary pmf q_0.

E-step: Calculate the conditional expectation

$$p_k = \mathbb{E}_{q_{k-1}}\{\hat{p}_N | \mathbf{y}\}$$

Here $\hat{p}_N(a) = \frac{\text{number of occurrences of } a \text{ in sequence } \mathbf{x}}{N}$ is the empirical distribution ("type"[6]) of

[6] An introduction into the concept of types can be found in [11].

the full sample $\mathbf{x}$. This empirical distribution cannot be observed directly, because we cannot observe $\mathbf{x}$. We can observe $\mathbf{y}$ though, which is of course statistically dependent on $\mathbf{x}$, due to the deterministic mapping $y_i = Tx_i$. The expected value is taken with respect to the current estimate of the true distribution, q_{k-1}. The result is a distribution p_k that is constructed by taking its previous estimate and the observed incomplete samples into account.

M-step: In the next step, we calculate the maximum-likelihood estimate of the distribution where the full sample $\mathbf{x}$ is coming from, using the distribution p_k obtained in the last step as the "empirical" distribution. This ML estimate is obtained as the result of the following KL divergence minimization (cf. [9], Lemma 3.1):

$$q_k = \arg\min_{q \in \mathcal{Q}} D(p_k \| q).$$

The EM algorithm is a special case of an alternating divergence minimization algorithm as introduced in the previous section. The M-step is obviously divergence minimization, where we minimize over all $q \in \mathcal{Q}$, $\mathcal{Q}$ being the set of "feasible" distributions where q is supposed to come from.

Now we will show, that the E-step minimizes the divergence $D(p\|q_{k-1})$ subject to $p \in \mathcal{P}$. Here, $\mathcal{P} = \{p : p^T = \hat{p}_n^T\}$ is the set of distributions p, whose image under the mapping T is equal to the image of the empirical distribution of the (unobservable) full sample. The image of a distribution p under a mapping $T : A \mapsto B$ is given by $p(b) = \sum_{a:Ta=b} p(a)$ - the probabilities of outcomes $a \in A$ that are mapped to the same $b \in B$ are added up.

One can show that for any distribution q on A, the conditional expectation $p = \mathbb{E}_q(\hat{p}_n|\mathbf{y})$ attains the minimum of $D(p\|q)$ subject to $p^T = \hat{p}_n^T$: Let $\delta(x, a) = 1$ if $x = a$

and 0 otherwise. Then we can write, for any $a \in A$

$$\begin{aligned} p_k(a) &= \mathbb{E}_{q_{k-1}}\{\hat{p}_n(a)|\mathbf{y}\} = \mathbb{E}_{q_{k-1}}\left\{\frac{1}{n}\sum_{i=1}^{n}\delta(x_i,a)\Big|\mathbf{y}\right\} \\ &= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{q_{k-1}}\{\delta(x_i,a)|y_i\} = \frac{1}{n}\sum_{i=1}^{n}\sum_{x_i} q_{k-1}(x_i|y_i)\delta(x_i,a) \\ &= \frac{|i:y_i = Ta|}{n}\frac{q_{k-1}(a)}{q^T_{k-1}(Ta)} = \hat{p}^T_n(Ta)\frac{q_{k-1}(a)}{q^T_{k-1}(Ta)}. \end{aligned} \tag{2.8}$$

In the first step we wrote the empirical distribution $\hat{p}_n(a)$ in terms of $\delta(x,a)$. Next we used the fact that the y_i are i.i.d. Then the conditional expectation in the sum is written out explicitely, and it turns out that it is actually the probability of $x_i = a$ conditioned on y_i, which is $\frac{q_{k-1}(a)}{q^T_{k-1}(Ta)}$ if $y_i = Ta$ and zero otherwise. The first fraction in the resulting expression can be recognized as the empirical distribution of y_i, called $\hat{p}^T_n$.

It remains to be shown that the $p_k(a)$ obtained in 2.8 has minimum KL divergence to $q_{k-1}(a)$. This can be done by using the "lumping property" (see Lemma 4.1 in [9]) $D(\tilde{p}\|q) \geq D(\tilde{p}^T\|q^T)$ for arbitrary $\tilde{p}$. Setting $\tilde{p} = p_k(a)$ in the lumping property shows that it satisfies the inequality with equality, meaning it minimizes $D(\tilde{p}\|q_{k-1})$ subject to $\tilde{p}^T = \hat{p}^T_n$.

We have thus shown that both in the E-step and the M-step a KL divergence is minimized: the E-step is an I-projection and the M-step an rI-projection.

3

Geometrical Interpretation of BICM Decoding

3.1 Introduction

The optimum decoder (in terms of sequence error probability) is the maximum a posteriori (MAP) decoder, which chooses of all possible transmit sequences the one with the highest probability given the observed symbols. In the case of BICM, an exhaustive search over all code words is generally the only possible implementation of the MAP decoder, with a complexity growing exponentially in codeword length. Algorithms that reduce this complexity, like the Viterbi algorithm, cannot be directly used for BICM in most cases, because they rely on the encoder being a Markov source, observed through a memoryless channel. The interleaving on bit level destroys this structure.

It will be shown that using an information-geometric approach, the decoding problem can be split up into three steps [8]: First, the a posteriori probabilities (APP) of the transmit sequences given the observed receive symbols are calculated, omitting the code structure. By doing so, the complexity of the APP calculation is reduced to a linear one, since only the marginal probabilities of the complex symbols have to be computed instead of the probabilities of the whole codewords. The second step accounts for the

code structure: the distribution resulting from the first step is projected onto the linear family of code-compatible distributions. This projection is shown to be an inverse I-projection. The third and last step is the maximization over all possible sequences.

The procedure described in the last paragraph still cannot be implemented directly, but [8] furthermore describes a practical algorithm justified by information-geometric arguments, which will also be dealt with in the present chapter.

3.2 BICM Decoding

First, let us explain where the problem with BICM decoding lies, which necessitates special precautions at the receiver. For many transmission schemes, a maximum a posteriori (MAP) or maximum likelihood (ML) sequence decoder, which is optimum in that it minimizes sequence error probability, can be implemented using a Viterbi algorithm (VA). For example, consider a transmission scheme consisting of a convolutional encoder followed by a PAM modulator, that maps the encoded bits to transmit symbols, which are sent over a memoryless channel. The receiver could be a soft-input Viterbi decoder that uses the received symbols as input. This is possible because the convolutional coder is a Markov source and the channel is memoryless. Symbol interleaving at the transmitter would still pose no problem, the interleaving could be reversed at the receiver before the deinterleaved symbols are forwarded to the VA.

The problem lies in the bit-level interleaving. Consider for example a 16-QAM constellation, where four coded bits are mapped to one complex transmit symbol. It can be seen that these four bits will in general correspond to different, non-adjacent trellis transitions, because they originate in coded bits belonging to different information bits. Although for some specific interleavers it is possible to construct a new trellis (with far more states than the original one) and still use the VA, it is in general not practical.[1]

[1] For BPSK constellations and QPSK constellations with Gray labeling, this problem does not occur, because only one bit is mapped to one symbol respectively one quadrature component.

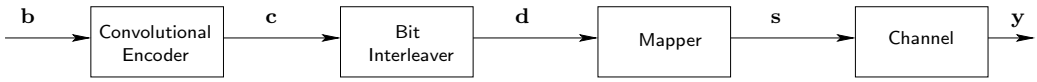

Figure 3.1: BICM transmitter consisting of encoder, interleaver and mapper, with physical channel.

For a BICM system with a 16QAM or higher constellation, the channel "seen" by each code bit, consisting of interleaving, mapping, and the physical channel, has to be considered a channel with memory, if the source is still modeled Markovian. Or alternatively, the channel can be modeled as memoryless, but the source is not Markovian anymore in this case. Either way, Viterbi decoding cannot be applied directly.

One solution to the problem is to demap the received symbols to code bits, so that these can be deinterleaved before decoding. This is not optimum, because the demapper's output are bit probabilities, and due to the channel being non-memoryless, the sequence probabilities cannot be simply recovered as the product of the bit probabilities anymore. But this is exactly the way BICM receiver discussed in the present chapter works, and it uses an iterative procedure to achieve near-optimum decoding.

3.3 System Model and Notation

In this chapter, vectors or sequences are represented by bold-face letters. The transmission scheme is shown in Fig. 3.1.

The length K sequence $\mathbf{b} = (b_0, \ldots, b_{K-1})$ of source bits b_i is first encoded into a length N sequence $\mathbf{c} = (c_0, \ldots, c_{N-1})$ of coded bits c_j. The codeword length $N \geq K$ depends on the encoder used. In the following we will only consider a convolutional encoder with rate $R = \frac{K}{N} = \frac{1}{2}$, although the results apply to other encoders. In this case, the sequence of coded bits has length $N = 2K$ and we can write $\mathbf{c} = (c_0^0, c_0^1, \ldots, c_{K-1}^0, c_{K-1}^1)$. There are of course more sequences of length N, which constitute the set $\mathcal{R} = \{0,1\}^N$, than there are valid codewords $\mathbf{c} \in \mathcal{C}$ of the same length, therefore $\mathcal{C} \subset \mathcal{R}$. Due to the 1:1-correspondence between sequences of information bits

and codewords, the cardinality of the set of codewords $|\mathcal{C}| = 2^K = 2^{NR}$ in contrast to $|\mathcal{R}| = 2^N = 2^{2K}$.

The next stage of the transmitter is the *interleaver* that scrambles the encoded bits to avoid burst errors at the receiver. It is represented by a permutation function $\pi(.)$ operating on the bit indices j.

The resulting interleaved bits $d_j = c_{\pi(j)}$ are then gathered into subsequences of B bits $\mathbf{d}_k = (d_{kB}, d_{kB+1}, \ldots, d_{(k+1)B-1})$ that are mapped to complex transmit symbols s_k from a given signal constellation of size $M = 2^B$. The sequence of interleaved coded bits corresponding to $\mathbf{c}$ is denoted as $\mathbf{d}$ and the corresponding set of interleaved codewords as $\mathcal{D}$. So we can also say that the interleaved code bits sequence $\mathbf{d}$ is mapped to the sequence of $L = N/B$ complex symbols $\mathbf{s}_\mathcal{C} = (s_0, \ldots, s_{L-1})$ which is sent over the channel. The subscript $\mathcal{C}$ is used to emphasize that this is a sequence that corresponds to a valid interleaved code sequence. We write simply $\mathbf{s}$ to denote an arbitrary sequence of transmit symbols. In the following, we denote by $\mathcal{S}_\mathcal{C}$ and $\mathcal{S}$ the set of symbol sequences corresponding to the set of code sequences respectively arbitrary symbol sequences of length L.

An arbitrary sequence of code bits will be denoted as $\mathbf{x}$. The interleaved bit sequence corresponding to the same code bit sequence is written as $\tilde{\mathbf{x}}$. The relationship between a code bit with the corresponding interleaved code bit is $\tilde{x}_j = x_{\pi(j)}$, respectively $x_{j'} = \tilde{x}_{\pi^{-1}(j')}$.

Only the AWGN case is considered here, but the results can easily extended to flat Rayleigh or Ricean fading channels. Therefore the input-output relation of the channel is

$$\mathbf{y} = \mathbf{s}_\mathcal{C} + \mathbf{n}, \quad \mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma_n \mathbf{I}),$$

or $y_k = s_k + n_k$ for the k-th symbol in the received symbols vector $\mathbf{y} = (y_0, \ldots, y_{L-1})$.

The receiver's task is to detect the transmitted information bits given the observed symbol sequence $\mathbf{s}$. The classical optimality criterion is the sequence error probabil-

ity, which is minimized by the MAP-detector (which reduces to the ML detector for uniformly distributed information sequences).

3.4 Interpretation of Soft Sequence Decoding

On the way to a practical implementation, we first introduce an information-geometric interpretation of soft sequence-decoding, which is general and not limited to BICM.

We will start from the MAP criterion and show how its calculation can be split up into three steps. The MAP decoder, which is equal to the maximum likelihood (ML) decoder for uniformly distributed information bits, chooses the information bit sequence that has the highest probability given the received symbol sequence:

$$\hat{\mathbf{b}} = \arg\max_{\mathbf{b}} p(\mathbf{b}|\mathbf{y}). \tag{3.1}$$

It can be shown that it minimizes the sequence error probability $\mathrm{P}(\hat{\mathbf{b}} \neq \mathbf{b}|\mathbf{y})$.

Carrying out this minimization directly has a complexity growing exponentially with information sequence length, since for each sequence the probability has to be calculated. There are algorithms like the Viterbi Algorithm (VA) that take advantage of the hidden markov model (HMM) structure of the convolutional encoder, avoiding the need to do an exhaustive search over all sequences. As already mentioned, this is not an option for BICM in most cases.

In the following, a three-step procedure is shown, as a first step to a practical implementation.

3.4.1 Step One: Calculating the "Observed Distribution"

In this step, the a posteriori pmf $p(\mathbf{s}|\mathbf{y})$, called "observed distribution" of the transmit symbol sequences $\mathbf{s}$ is calculated, ignoring the constraints that are mandated by the code. In other words, we simply calculate the probabilities of all possible transmit

symbol sequences given the received sequence $\mathbf{y} = \mathbf{s} + \mathbf{n}$. The code structure will be "reintroduced" in the next step.

Let's first consider a series of equalities that are due to the various 1:1-mappings of the sequences we have considered so far, starting from the MAP-criterion:

$$\begin{aligned} \max_{\mathbf{b}} p(\mathbf{b}|\mathbf{y}) &= \max_{\mathbf{c}\in\mathcal{C}} p_{\mathcal{C}}(\mathbf{c}|\mathbf{y}) \\ &= \max_{\mathbf{d}\in\mathcal{D}} p_{\mathcal{D}}(\mathbf{d}|\mathbf{y}) = \max_{\mathbf{s}_{\mathcal{C}}\in\mathcal{S}_{\mathcal{C}}} p_{\mathcal{S}_{\mathcal{C}}}(\mathbf{s}_{\mathcal{C}}|\mathbf{y}). \end{aligned} \tag{3.2}$$

The maximizing arguments of each of these expressions also have 1:1 correspondences with each other, via encoding, interleaving and mapping, respectively.

Note that it is very important that for this equalities to hold one has to carry out the maximizations over the right subset of sequences, $\mathcal{C}$, $\mathcal{D}$ or $\mathcal{S}_{\mathcal{C}}$!

Now let's calculate the pmf $p(\mathbf{s}|\mathbf{y})$ for *all possible* sequences $\mathbf{s} \in \mathcal{S}$, including those which do not correspond to valid code sequences. Using the Bayes Criterion and the fact that the channel is memoryless (each "channel use" is independent), we can write

$$p(\mathbf{s}|\mathbf{y}) = \frac{p(\mathbf{s})}{p(\mathbf{y})} p(\mathbf{y}|\mathbf{s}) = \frac{p(\mathbf{s})}{p(\mathbf{y})} \prod_k p(y_k|s_k).$$

$p(\mathbf{y})$ is a positive factor that does not depend on s, so it can be omitted for the maximization. Further assuming uniformly distributed information bits (which actually means using the ML-criterion instead of the MAP-criterion), thus uniformly distributed code and symbol sequences, we can also omit $p(\mathbf{s})$ because it is constant for all $\mathbf{s}$. Thus we finally obtain

$$p(\mathbf{s}|\mathbf{y}) \propto \prod_k p(y_k|s_k).$$

We observe that the probability of a whole sequence (omitting the code structure) is actually the product of its symbol probabilities. The complexity of calculating the sequence probabilities is thus reduced to a linearly growing one.

Using the correspondence between symbols s_k and subsequences of interleaved bits

$\tilde{\mathbf{x}}_k$, we can compute $p(\tilde{\mathbf{x}}|\mathbf{y})$ for every sequence $\tilde{\mathbf{x}} \in \mathcal{R}$:

$$p(\tilde{\mathbf{x}}|\mathbf{y}) = p(\mathbf{s}(\tilde{\mathbf{x}})|\mathbf{y}) \propto \prod_k p(y_k|\tilde{\mathbf{x}}_k). \tag{3.3}$$

Because the de-interleaved bits $\mathbf{x}$ correspond 1:1 to the interleaved bits $\tilde{x}$, this is further equivalent to evaluating the pmf

$$p_{\mathcal{R}}(\mathbf{x}) = p(\mathbf{x}|\mathbf{y}) \propto \prod_k p(y_k|\mathbf{x}_k), \tag{3.4}$$

where $\mathbf{x}_k = (x_{\pi^{-1}(kB)}, \ldots, x_{\pi^{-1}(k+1)B-1)})$. Note: this holds for $\mathbf{x} \in \mathcal{R}$, meaning all sequences of length N. In other words: this probability distribution will in general also have non-zero value for sequences $\mathbf{x}$ that are not in the code. It is intuitive that these sequences should not be considered when doing the maximization, so we need to get rid of them before. This is done in the next step.

3.4.2 Step Two: I-Projection onto the Code Manifold

Omitting the code structure altogether of course won't lead to an optimum decoder. We will account for it in the present step by taking the sequence APPs obtained in the first step and projecting it onto the submanifold of probability distributions that are compatible with the code. Intuitively, this submanifold only may contain distributions that have zero probability for sequences *not* included in the code, leaving all the non-zero probabilities to the valid code sequences.

We are now going to present two approaches to this projection problem.

Approach I

The first approach is taken from [8]. It starts with the following

Definition 3.4.1. Code-compatible distributions

$$\mathcal{E}_{\mathcal{C}} = \{p \,|\, p(\mathbf{x}) = 0 \text{ iff } \mathbf{x} \notin \mathcal{C}\}.$$

This can be shown to be an exponential family but this fact is not actually used in the following.

The observed distribution $p(\mathbf{x}|\mathbf{y})$ is now projected upon this set of code-compatible distributions.

Theorem 3.4.1. Projection onto the code-compatible distributions

For any distribution $p(\mathbf{x})$, the closest distribution to p in $\mathcal{E}_{\mathcal{C}}$,

$$p_{\mathcal{C}} = \arg\min_{q \in \mathcal{E}_{\mathcal{C}}} D(q\|p),$$

is the distribution $p_{\mathcal{C}}$ given for each $\mathbf{x}$ by

$$p_{\mathcal{C}}(\mathbf{x}) = \frac{p(\mathbf{x}) I_{\mathcal{C}}(\mathbf{x})}{\sum_{\mathbf{x} \in \mathcal{R}} p(\mathbf{x}) I_{\mathcal{C}}(\mathbf{x})}, \tag{3.5}$$

where $I_{\mathcal{C}}$ is the "indicator function" of the code, given by

$$I_{\mathcal{C}} = \begin{cases} 1 \text{ if } \mathbf{x} \in \mathcal{C} \\ \\ 0 \text{ if } \mathbf{x} \notin \mathcal{C} \end{cases}.$$

For any distribution p and for any distribution $r \in \mathcal{E}_{\mathcal{C}}$, we have $D(r\|p) = D(r\|p_{\mathcal{C}}) + D(p_{\mathcal{C}}\|p)$.

Proof. Let us consider a code-compatible distribution $r \in \mathcal{E}_{\mathcal{C}}$ and any discrete proba-

bility distribution p. Then

$$\begin{aligned} D(r\|p) &= \sum_{\mathbf{x}} r(\mathbf{x}) \log \frac{r(\mathbf{x})}{p(\mathbf{x})} = \sum_{\mathbf{x}} r(\mathbf{x}) \left(\log \frac{r(\mathbf{x})}{\frac{p(\mathbf{x})I_{\mathcal{C}}(\mathbf{x})}{\sum_{\mathbf{x}'} p(\mathbf{x}')I_{\mathcal{C}}(\mathbf{x}')}} + \log \frac{\frac{p(\mathbf{x})I_{\mathcal{C}}(\mathbf{x})}{\sum_{\mathbf{x}'} p(\mathbf{x}')I_{\mathcal{C}}(\mathbf{x}')}}{p(\mathbf{x})} \right) \\ &= D\Bigg(r \Bigg\| \underbrace{\frac{p(\mathbf{x})I_{\mathcal{C}}(\mathbf{x})}{\sum_{\mathbf{x}'} p(\mathbf{x}')I_{\mathcal{C}}(\mathbf{x}')}}_{p_{\mathcal{C}}} \Bigg) + \sum_{\mathbf{x}} r(\mathbf{x}) \log \frac{I_{\mathcal{C}}(\mathbf{x})}{\sum_{\mathbf{x}'} p(\mathbf{x}')I_{\mathcal{C}}(\mathbf{x}')}. \end{aligned}$$

The term on the left is $D(r\|p_{\mathcal{C}})$ and therefore zero for $r = p_{\mathcal{C}}$. The right hand term does not depend on r and is equal to $D(p_{\mathcal{C}}\|p)$:

$$\begin{aligned} \sum_{\mathbf{x}} r(\mathbf{x}) \log \frac{I_{\mathcal{C}}(\mathbf{x})}{\sum_{\mathbf{x}'} p(\mathbf{x}')I_{\mathcal{C}}(\mathbf{x}')} &= \left(\sum_{\mathbf{x}} r(\mathbf{x}) \right) \log \frac{1}{\sum_{\mathbf{x}'} p(\mathbf{x}')I_{\mathcal{C}}(\mathbf{x}')} \\ &= \log \frac{1}{\sum_{\mathbf{x}'} p(\mathbf{x}')I_{\mathcal{C}}(\mathbf{x}')} \\ &= \sum_{\mathbf{x}} \underbrace{\frac{p(\mathbf{x})I_{\mathcal{C}}(\mathbf{x})}{\sum p(\mathbf{x})I_{\mathcal{C}}(\mathbf{x})}}_{p_{\mathcal{C}}} \log \underbrace{\frac{p(\mathbf{x})I_{\mathcal{C}}(\mathbf{x})}{\sum p(\mathbf{x})I_{\mathcal{C}}(\mathbf{x})}}_{p_{\mathcal{C}}} \frac{1}{p(\mathbf{x})}. \end{aligned}$$

In the first step we use the fact that $r(\mathbf{x})$ is code-compatible and therefore $r(\mathbf{x})I_{\mathcal{C}}(\mathbf{x}) = r(\mathbf{x})$. In the final step, we multiplied with the sum over $p_{\mathcal{C}}$, which is one, and extended the fraction in the logarithm with $I_{\mathcal{C}}(\mathbf{x})\frac{p(\mathbf{x})}{p(\mathbf{x})}$, which does not change the value of the expression, because the summation is actually done only over the valid codewords.

□

We have shown that the distribution that is compatible with the code and with minimum KL divergence to the observed distribution $p(\mathbf{x}|\mathbf{y})$ is given by the expression (3.5). The denominator in that expression is a normalization factor, which ensures that $p_{\mathcal{C}}$ is a valid pmf, i.e. that the sum over all its values is one.

This approach interprets the code-compatible densities as an exponential family, but does not make use of this fact. Furthermore, the projection given in the above theorem is easily recognized to be an I-projection, and the equality given at the end of the theorem is the Pythagorean Theorem for a projection onto a linear family that we

discussed in section 2.2.4.

Approach II

Another approach [12] regards the manifold of code-compatible distributions as a linear family right from the start, and furthermore gives a more general view of our problem.

When inferring the distribution $p(\mathbf{x})$ (in our case $p_{\mathcal{C}}(\mathbf{x})$) of a random variable $\mathbf{x}$ with an *a priori* distribution $q(\mathbf{x})$ (here $p(\mathbf{x}|\mathbf{y})$), the *Principle of minimum cross-entropy (=KL divergence)* selects that distribution $p(\mathbf{x})$ which has the minimum KL divergence $D(p\|q)$ subject to the known constraints on the random variable $\mathbf{x}$. This can be formally written as follows:

Find $p(\mathbf{x})$ such that

$$D(p\|q) = \arg\min_{s} D(s\|q), \tag{3.6}$$

subject to

$$\mathbb{E}_p\{\mathbf{f}(\mathbf{x}) = \mathbf{0}\}. \tag{3.7}$$

For our projection problem, the component functions f_i that constitute $\mathbf{f}$ in (3.7) correspond to the parity-check equations of the code. For a linear binary (N, K)-block code, there will be $F = N - K$ independent check equations. A check equation can be expressed as an equality constraint, e.g.

$$f_i(\mathbf{x}) = x_1 \oplus x_2 \oplus x_6 = 0. \tag{3.8}$$

If the code bits satisfy the constraint, which means that $\mathbf{x}$ is a valid codeword, $f_i(\mathbf{x}) = 0$, else $f_i(\mathbf{x}) = 1$. Since the constraint functions f_i are nonnegative, the parity check equation (3.8) is equivalent to the expected value constraint $\mathbb{E}_p\{f_i(\mathbf{x})\} = 0$. This gives rise to the following

Definition 3.4.2. Family of code-compatible distributions (alternative definition)

$$\mathcal{L}_{\mathcal{C}} = \Big\{ p_{\mathcal{C}}(\mathbf{x}) \colon \mathbb{E}_p\big\{\mathbf{f}(\mathbf{x})\big\} = \mathbf{0} \Big\}. \tag{3.9}$$

The essential property of this family is the following: each distribution of the family has non-zero probability values for valid codewords, and zero probability for non-codewords.

Recalling the information geometric setting established in Chapter 2, we recognize (3.7) as a linear family, and (3.6) is therefore the I-projection of q onto $\mathcal{L}_{\mathcal{C}}$. Thus, $p_{\mathcal{C}}(\mathbf{x})$ is of the form (see also the remark 3.1 in [9])

$$p_{\mathcal{C}}(\mathbf{x}) = c \cdot q(\mathbf{x}) \exp\Big(\sum_{1}^{K} \theta_i f_i(\mathbf{x}) \Big). \tag{3.10}$$

In order to find the values θ_i, one first notes that c, $q(\mathbf{x})$ and $f_i(\mathbf{x})$ are all non-negative. Thus for any $\mathbf{x}$ such that $f_i(\mathbf{x}) > 0$, θ_i has to be $-\infty$ for the restriction (3.9) to hold. With $\theta_i = -\infty$

$$\exp(\theta_i f_i(\mathbf{x})) = \begin{cases} 0 & \text{if } f_i(\mathbf{x}) \neq 0 \\ 1 & \text{if } f_i(\mathbf{x}) = 0 \end{cases} = I_i(\mathbf{x}),$$

where $I_i(\mathbf{x})$ is the indicator function for the sequences $\mathbf{x}$ that satisfy the i-th check equation of the code.

In this manner, the expression (3.10) becomes

$$p(\mathbf{x}) = c q(\mathbf{x}) I_1(\mathbf{x}) I_2(\mathbf{x}) \cdots I_F(\mathbf{x}).$$

Because a valid codeword has to satisfy all the parity check equations simultaneously, the individual indicator functions can be collected into the indicator function for the code, $I_{\mathcal{C}}(\mathbf{x}) = \prod_{i=1}^{F} I_i(\mathbf{x})$. Using $p(\mathbf{x}) = p(\mathbf{x}|\mathbf{y})$ the posterior pmf $p_{\mathcal{C}}(\mathbf{x}) = p(\mathbf{x})$ is finally

written as

$$p_{\mathcal{C}}(\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{y})I_{\mathcal{C}}(\mathbf{x})}{\sum_{\mathbf{x}\in\mathcal{R}} p(\mathbf{x}|\mathbf{y})I_{\mathcal{C}}(\mathbf{x})}.$$

Here the constant c has already been chosen so that

$$\sum_{\mathbf{x}\in\mathcal{X}} p_{\mathcal{C}}(\mathbf{x}) = 1.$$

Comparing the two approaches we conclude that they lead to the same solution: we end up with the I-projection $p_{\mathcal{C}}$ of $p(\mathbf{x}|\mathbf{y})$ onto the code-compatible distributions.

3.4.3 Step Three: Maximization

In the previous step we have shown the distribution in the code-submanifold that has minimum KL divergence to the observed distribution $p(\mathbf{x}|\mathbf{y})$ of all possible sequences to be its I-projection onto the code-submanifold $\mathcal{L}_{\mathcal{C}}$. The expression can be written explicitly as

$$p_{\mathcal{C}}(\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{y})I_{\mathcal{C}}(\mathbf{x})}{\sum_{\mathbf{x}\in\mathcal{R}} p(\mathbf{x}|\mathbf{y})I_{\mathcal{C}}(\mathbf{x})}.$$

The third step is to perform the maximization. It can be easily shown that maximizing $p_{\mathcal{C}}$ obtained in the projection step over all possible bit sequences (3.11d) leads to the same resulting code sequence estimate $\hat{\mathbf{c}}$ as evaluating the MAP-criterion directly:

$$\hat{\mathbf{c}}_{\mathrm{MAP}} = \arg\max_{\mathbf{c}\in\mathcal{C}} p(\mathbf{c}|\mathbf{y}) \tag{3.11a}$$

$$= \arg\max_{\mathbf{c}\in\mathcal{C}} p(\mathbf{c}|\mathbf{y})I_{\mathcal{C}}(\mathbf{c}) \tag{3.11b}$$

$$= \arg\max_{\mathbf{c}\in\mathcal{C}} p_{\mathcal{C}}(\mathbf{c}) \tag{3.11c}$$

$$= \arg\max_{\mathbf{x}\in\mathcal{R}} p_{\mathcal{C}}(\mathbf{x}). \tag{3.11d}$$

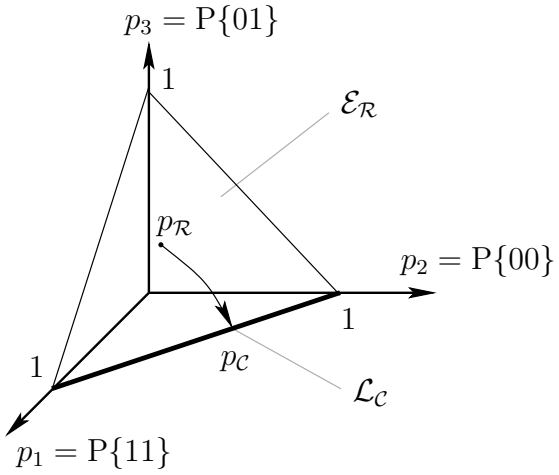

Figure 3.2: Soft sequence decoding.

3.4.4 Example

As an illustration of the information-geometric interpretation of the soft sequence-decoding process, we consider the simple case of length-two bit sequences and a repetition code.

There are four bit sequences $\mathbf{x}$ of length two: $\mathbf{x} \in \mathcal{R} = \{00, 01, 10, 11\}$. The codebook of the repetition code consists only of two codewords, $\mathcal{C} = \{00, 11\}$.

A pmf $p(\mathbf{x}) \in \mathcal{E}_\mathcal{R}$ consists of four probabilities for the four different bit sequences: $p(00) = \mathrm{P}\{00\}$, $p(01) = \mathrm{P}\{01\}$, $p(10) = \mathrm{P}\{10\}$, $p(11) = \mathrm{P}\{11\}$. As $\sum_\mathbf{x} p(\mathbf{x}) = 1$, three probabilities fully determine the fourth one, for example, $p(10) = p(11) - p(01) - p(00)$. The set of valid probability distributions p is therefore a three-dimensional manifold ($\mathcal{E}_\mathcal{R}$ in Fig. 3.2). It consists of the three-dimensional simplex between the corner points with coordinates $(0, 0, 0)$,$(1, 0, 0)$,$(0, 1, 0)$ and $(0, 0, 1)$. The coordinates used are mixture coordinates $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3)$, which for discrete distributions are simply the probabilities of the different outcomes, $\alpha_i = p_i$, i being the decimal value of the sequence.

Note the difference to the visualization we presented in the example 2.2.2, Fig. 2.1. There, the manifold was two-dimensional, but we had chosen to embed it into a three-dimensional coordinate system, to make things easier to understand. Now, we make use of the fact that the fourth probability is totally defined by the other three

probabilities and therefore we can use a three-dimensional visualization although we have four probabilities. We have chosen to omit $p(10)$ from the coordinate system. This means, that in the origin lies the distribution $\alpha = (0, 0, 0)$ where $p(10) = 1$ and the othere probabilities are zero.

Let's give one more example to make the meaning of the coordinate system used more clear. The point $\boldsymbol{\alpha}' = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right)$ represents the uniform distribution for which $p(00) = p(01) = p(10) = p(11) = \frac{1}{4}$.

The repetition code allows only the sequences 00 and 11. The linear family of code-compatible densities is in this case and in the chosen coordinate system a "straight line" (m-geodesic) between the points $(1, 0, 0)$ and $(0, 1, 0)$. It can be easily be verified that all points on this connection line correspond to distributions that have zero probability for the non-codewords 01 and 10. That is, $\mathcal{L}_\mathcal{C}$ is the convex combination

$$\mathcal{L}_\mathcal{C} = \{p_t \,|\, p_t = (1-t)p_0 + tp_1, \quad 0 \leq t \leq 1\}.$$

Now that the setting for the example is laid out, the actual interpretation of soft-decoding in the context of the example is quickly explained. The first step is to calculate the observed distribution $p_\mathcal{R}(\mathbf{x}) = p(\mathbf{x}|\mathbf{y})$. This distribution is represented by a point inside the three-dimensional simplex representing the manifold $\mathcal{E}_\mathcal{R}$. The second step is the I-projection of $p_\mathcal{R}(\mathbf{x})$ onto the code-manifold $\mathcal{L}_\mathcal{C}$, we obtain the distribution $p_\mathcal{C}(\mathbf{x}) = \arg\min_{p \in \mathcal{L}_\mathcal{C}} D(p \| p_\mathcal{R})$. The third step (maximization) is in the present example a decision between the two sequences 00 and 11.

3.5 A Practical Algorithm

The procedure described in the previous section is a reinterpretation of maximum-likelihood soft sequence-decoding. The complexity of soft sequence-decoding generally increases exponentially with sequence length, because the posterior probability of each

code sequence has to be calculated. The information geometric interpretation splits the problem up into to sub-problems. The first step, the calculation of the "observed" distribution $p_{\mathcal{R}}(\mathbf{x}) = p(\mathbf{x}|\mathbf{y})$, only has linear complexity, because it can be split up into the products of individual symbol pmfs. But the second step still has exponential complexity, because the calculation of the projection onto the code manifold $p_{\mathcal{C}}$ requires to go through all the sequences and decide wether they are in the codebook or not.

For many transmission systems, the implementation of optimum or near-optimum sequence detection is still possible with reasonable complexity by using e.g. the Viterbi algorithm or the BCJR-algorithm. But direct application of such an algorithm is not an option for BICM. These algorithms assume a Markovian source that is observed through a memoryless channel. Because of the interleaver, the channel cannot be regarded as memoryless anymore.

However, there are standard BICM receiver algorithms, described e.g. in [4] An information-geometrical interpretation of one such algorithm is proposed in [8], this interpretation will be presented in the following.

The basic idea is to split up the detection problem into two main subblocks. The first subblock, in the following referred to as the "demapper", calculates probabilities for the code bits, assuming that the bits carried by a received symbol are independent. The code structure is not considered - it cannot be, because the demapper carries out symbol-by-symbol processing, it does not consider the whole symbol sequence.

The second main block is a soft-input-soft-output ("SISO"[2]) channel decoder, in the following called "decoder". Its task is to decode the channel code using the bit probabilities provided by the demapper, and to produce information bit probabilities.

The two blocks are linked by a deinterleaver that reverses the bitwise interleaving that has been performed at the transmitter.

The receiver can be improved by using an iterative procedure, which we will describe

[2]Unfortunately, this is the same acronym as the one of the well established term "single-input-single-output".

Figure 3.3: BICM receiver, consisting of demapper, deinterleaver and SISO decoder.

in the final section of the present chapter. For now, we will restrict ourselves to the non-iterative system shown in Fig. 3.3. In the following sections we will describe the different subblocks in detail and show that their function can be interpreted as projections of probability distributions.

3.5.1 The Demapping Sub-Block

In section 3.4, the first step was to calculate the "observed" distribution $p(\mathbf{x}|\mathbf{y})$, where $\mathbf{x} \in \mathcal{R}$ stands for any possible bit sequence with length N. It was shown that this pmf can be factored into individual symbol probabilities.

To be able to use bit-wise processing, we need a factorization of $p(\mathbf{x}|\mathbf{y})$ into individual *bit* probabilities,

$$p_{\text{dem}}(\mathbf{x}|\mathbf{y}) = \prod_j p_j(x_j|y_{k(j)}). \tag{3.12}$$

Remember that j is the index of the coded bits and $k(j)$ denotes the index of the received symbol y_k that carries the coded bit c_j.

To simplify things in the following we assume a rate-$\frac{1}{2}$ convolutional code, which means that every information bit b_i is encoded into two coded bits c_i^0 and c_i^1. Furthermore we will now describe the interleaver with the two functions π_i^0 and π_i^1, that give the indices of the transmit symbols that carry the encoded bits c_0 and c_1, respectively. A code bit can now be linked with the corresponding received symbol the bit is mapped to, for example c_i^0 is mapped to $y_{\pi_i^0}$.

The next assumption is that we have chosen a labeling for which a code bit is mapped either to the in-phase or to the quadrature component. The function $f_{\pi_i^0}$ denotes the function representing either the real part or the imaginary part function, depending on which component the bit c_i^0 is mapped to.

With this notation and assumptions, we obtain for the observed distribution the expression

$$p(\mathbf{x}|\mathbf{y}) = c \prod_i p_i^0(x_i^0|f_{\pi_i^0}(y_{\pi_i^0}))\, p_i^1(x_i^1|f_{\pi_i^1}(y_{\pi_i^1})). \tag{3.13}$$

Note that for this equation to hold, the equivalent channel "seen" by each code bit, consisting of bit interleaver, mapping and transmission channel, has to be memoryless. Only in that case the received code bits are statistically independent[3] and the sequence pmf is the product of the bit pmfs. This is true for simple signal constellations like BPSK or 4QAM with Gray labeling, but not for higher-order constellations, as explained in section 3.2. In other words, using this relation for higher signal constellations is an approximation, and will lead to suboptimum sequence error performance in general.

But there is an information-geometric justification for the approximation that the received code-bits are independent: it is the best approximation in the sense that the factorizable distribution given in (3.12) has minimum KL divergence to the original distribution $p(\mathbf{x}|\mathbf{y})$.

In fact, the factorization can be regarded as an inverse I-projection onto the factorizable distributions $\mathcal{E}_{\mathcal{F}}$. This has been shown in example 2.2.7.

By approximating the observed distribution by the product of individual bit probabilities, the demapper sub-block does an implicit projection of the observed distribution onto the factorizable distributions.

The observed distribution $p(\tilde{\mathbf{x}}|\mathbf{y})$ written in terms of the interleaved encoded bits $\tilde{\mathbf{x}}$ is given by

$$p(\tilde{\mathbf{x}}|\mathbf{y}) = p(\mathbf{s}(\tilde{\mathbf{x}})|\mathbf{y}) = \prod_k p(\tilde{\mathbf{x}}_k|y_k). \tag{3.14}$$

Let us now concentrate on the individual factors of the product. Each one is a posterior probability for the interleaved bit sub-sequence $\tilde{\mathbf{x}}_k$ carried by the observed received symbol y_k. Here, $\tilde{\mathbf{x}}_k = (\tilde{x}_{kB}, \tilde{x}_{kB+1}, \ldots, \tilde{x}_{kB+(B-1)}) =: (\tilde{x}_k^0, \tilde{x}_k^1, \ldots, \tilde{x}_k^{(B-1)})$. To each

[3]The code bits are of course never independent, because they have dependencies due to the code structure. But for the calculation of the observed distribution we don't consider the code structure!

interleaved bit sub-sequence there is the corresponding sequence of non-interleaved code bits $\mathbf{x}_k = (x_{\pi^{-1}(kB)}, x_{\pi^{-1}(kB+1)}, \ldots, x_{\pi^{-1}(kB+(B-1))})$. The bit probabilities can now be calculated by marginalizing:

$$p(\tilde{x}_k^l|y_k) = p(x_{\pi^{-1}(kB+l)}|y_k) = \sum_{s_k:\tilde{x}_k^l} p(s_k|y_k),$$

that is, by summing the symbol APP $p(s_k|y_k)$ over all transmit symbols s_k of the symbol alphabet whose bit at position l has the value $\tilde{x}_k^l$. As there is a one-to-one mapping between transmit symbols and interleaved bits, we can further write

$$p(\tilde{x}_k^l|y_k) = \sum_{\tilde{\mathbf{x}}_k \sim \tilde{x}_k^l} p(s_k(\tilde{\mathbf{x}}_k)|y_k), \tag{3.15}$$

and, using Bayes' theorem and assuming uniformly distributed code bits and thus transmit symbols,

$$p(\tilde{x}_k^l|y_k) \propto \sum_{\tilde{\mathbf{x}} \sim \tilde{x}_k^l} p(y_k|s_k(\tilde{\mathbf{x}}_k). \tag{3.16}$$

For an AWGN channel with noise variance σ^2, $p(y_k|s_k(\tilde{\mathbf{x}}_k))$ is

$$p(y_k|s_k(\tilde{\mathbf{x}}_k)) = e^{-\frac{1}{2\sigma^2}|y_k - s_k|^2}.$$

Using the one-to-one correspondence between s_k and $\tilde{\mathbf{x}}_k$, and normalizing appropriately, the a posteriori probability of the interleaved bit $\tilde{x}_k^l$, is

$$p_{\text{dem}}(\tilde{x}_k^l) := p(\tilde{x}_k^l|y_k) = \frac{\sum_{\tilde{\mathbf{x}}_k:\tilde{x}_k^l} p(y_k|\tilde{\mathbf{x}}_k)}{\sum_{\tilde{\mathbf{x}}_k} p(y_k|\tilde{\mathbf{x}}_k)}. \tag{3.17}$$

This is the "local", that is, *symbol level* description of the demapper sub-block. For one given received symbol y_k it produces $2B$ (the number of bits per symbol of the constellation times two, for values 0 and 1) bit-probabilities. This local behavior is illustrated in Fig. 3.4.

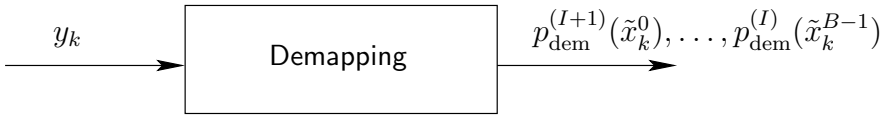

Figure 3.4: Local interpretation of the demapper sub-block.

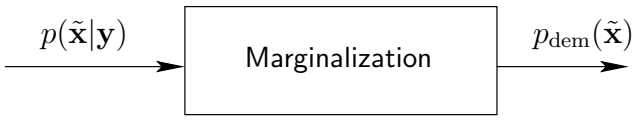

Figure 3.5: Global interpretation of the demapper sub-block.

The "global" description, i.e. the description on the level of sequence probabilities, is given by the projection onto the factorizable distributions, which is the product of the individual bit probabilities. The demapper's output distribution can therefore finally be written as

$$\begin{aligned} p_{\mathrm{dem}}(\tilde{\mathbf{x}}) &= \prod_k \prod_{l=0}^{B-1} \sum_{s_k : x_{\pi^{-1}(kB+l)}} p(\tilde{\mathbf{x}}_k | y_k) \\ &= \prod_{k,l} p(\tilde{x}_k^l | y_k) \\ &= \prod_{k,l} p_{\mathrm{dem}}(\tilde{x}_k^l). \end{aligned} \tag{3.18}$$

This interpretation is illustrated in Fig. 3.5.

3.5.2 BCJR Decoding Sub-Block

The BCJR decoder (also called "forward-backward" or "sum-product" algorithm) takes a sequence of bit-probabilities as input. It produces a sequence of updated probabilities, using the whole sequence and taking the code structure into account, as output. A detailed discussion of the BCJR algorithm can be found for example in the original paper [13] by the inventors of the algorithm. It is applicable to our problem because it is a method of calculating a posteriori *bit* probabilities, instead of sequence probabilities, which the Viterbi Algorithm would produce. This will be essential for iterative (turbo) decoding, as we will see later. For non-iterative BICM decoding, the use of a (soft-input)

VA would be also possible.

Geometrically, the operation of the BJCR-algorithm can be interpreted as two successive projections: The starting point is a factorizable probability distribution. This input probability distribution is first projected onto the manifold of code-compatible densities, which is a linear family, but not necessarily a factorizable distribution. In a second step, the newly found distribution is again projected onto the manifold of factorizable densities. This is what we will show in the following.

The output distribution of the demapper, after having been deinterleaved, serves as the input distribution for the decoder,

$$p_{\mathrm{dem}}(\mathbf{x}) = \prod_{m,n} p_{\mathrm{dem}}(x_m^n),$$

which is a factorizable distribution consisting of the code bit probabilities.

In the following, we will assume a rate-$\frac{1}{2}$ convolutional channel code code, which means that every information bit b_j corresponds to two code bits, (c_j^0, c_j^1). The convolutional code is represented by a time-invariant trellis with M states (the number of states depends on the code). Each trellis transition from one state m to another state m' corresponds to one combination of two output code bits. From one time point k to the next $k+1$, there are only certain allowed trellis transitions - this represents the code constraints. By taking a path through the trellis and concatenating the code bits of all the transitions, a valid codeword is obtained.

The BCJR algorithm operates on the code trellis and recursively computes the following two quantities:

$$\alpha_k(m) = \sum_{\mathbf{c}_1^k \in \mathcal{C}_1^k(m)} \prod_{j=1}^{k} p_{\mathrm{dem}}(c_j^0) p_{\mathrm{dem}}(c_j^1)$$

and

$$\beta_k(m) = \sum_{\mathbf{c}_{k+1}^N \in \mathcal{C}_{k+1}^N(m)} \prod_{j=k+1}^{N} p_{\text{dem}}(c_j^0) p_{\text{dem}}(c_j^1),$$

where $\mathcal{C}_1^k(m)$ stands for the set of $2k$ bits that are allowed by the trellis structure and *end* in state m at time k. Similarly $\mathcal{C}_{k+1}^N(m)$ stands for the set of $2(N-k)$ bits that correspond to allowed valid paths in the trellis and *begin* in state m at time k.

These quantities can be seen as the projection of the probabilities of the length-$2k$ bit-sequences onto the set of partial codewords starting an ending in state m. Using the indicator function $I_{\mathcal{C}_1^k(m)}(.)$ that has value one only for these codewords, zero for the rest of the sequences, we can rewrite $\alpha_k(m)$ as

$$\begin{aligned}\alpha_k(m) &= \sum_{\mathbf{c}_1^k \in \mathcal{C}_1^k(m)} \prod_{j=1}^{k} p_{\text{dem}}(c_j^0) p_{\text{dem}}(c_j^1) \\ &= \sum_{\mathbf{x}_1^k \in \mathcal{R}_1^k(m)} \left(\prod_{j=1}^{k} p_{\text{dem}}(x_j^0) p_{\text{dem}}(x_j^1) \right) I_{\mathcal{C}_1^k(m)}(\mathbf{x}_1^k).\end{aligned}$$

In the same manner, denoting by $I_{\mathcal{C}_1^k}(.)$ the set of all partial code sequences of length $2k$, $\sum_m \alpha_k(m)$ can be seen as the projection of the probabilities of the sequences of $2k$ bits onto the set of partial codewords. Indeed $I_{\mathcal{C}_1^k}(.) = \sum_m I_{\mathcal{C}_1^k(m)}(.)$ and therefore

$$\begin{aligned}\sum_m \alpha_k(m) &= \sum_m \sum_{\mathbf{c}_1^k \in \mathcal{C}_1^k(m)} \prod_{j=1}^{k} p_{\text{dem}}(c_j^0) p_{\text{dem}}(c_j^1) \\ &= \sum_m \sum_{\mathbf{x}_1^k \in \mathcal{R}_1^k(m)} \left(\prod_{j=1}^{k} p_{\text{dem}}(x_j^0) p_{\text{dem}}(x_j^1) \right) I_{\mathcal{C}_1^k(m)}(\mathbf{x}_1^k) \\ &= \sum_{\mathbf{x}_1^k \in \mathcal{R}_1^k(m)} \left(\prod_{j=1}^{k} p_{\text{dem}}(x_j^0) p_{\text{dem}}(x_j^1) \right) \sum_m I_{\mathcal{C}_1^k(m)}(\mathbf{x}_1^k) \\ &= \sum_{\mathbf{x}_1^k \in \mathcal{R}_1^k(m)} \left(\prod_{j=1}^{k} p_{\text{dem}}(x_j^0) p_{\text{dem}}(x_j^1) \right) I_{\mathcal{C}_1^k}(\mathbf{x}_1^k).\end{aligned} \tag{3.19}$$

The β-coefficients can be interpreted in a similar way by considering the sets $\mathcal{C}_k^N(m)$.

Once the α_k's and β_k's have been calculated, the BCJR algorithm evaluates the probability that a transition from state m to state m' occurs at time k of the trellis:

$$\sigma_k(m,m') = \alpha_{k-1}(m) p_{\text{dem}}(x_k^0 = c_{m\to m'}^0) p_{\text{dem}}(x_k^1 = c_{m\to m'}^1) \beta_k(m') I_c(m,m'), \qquad (3.20)$$

where $I_c(m,m')$=1 only if the transition $m \to m'$ is allowed by the trellis, 0 else. Furthermore, $c_{m\to m'}^0$ and $c_{m\to m'}^1$ denote the values of the first respectively the second output bit corresponding to the trellis transition $m \to m'$.

We see that this is the moment where the a priori probabilities of the coded bits $p_{\text{dem}}(x_j^{0,1})$, that are provided by the demapper, are incorporated by the decoder.

In the same way we did for the α and β coefficients, we will now rewrite $\sigma_k(m,m')$ using some indicator functions, in order to reveal the underlying projection onto the code structure. Let us first use the expression (3.5.2) and the analogue for β in (3.20):

$$\begin{aligned}\sigma_k(m,m') = &\left(\sum_{\mathbf{c}_1^k \in \mathcal{C}_1^k(m)} \prod_{j=1}^{k} p_{\text{dem}}(c_j^0) p_{\text{dem}}(c_j^1) \right) \big(p_{\text{dem}}(x_k^0 = c_{m\to m'}^0) \\ &p_{\text{dem}}(x_k^1 = c_{m\to m'}^1) I_c(m,m') \big) \left(\sum_{\mathbf{c}_{k+1}^N \in \mathcal{C}_{k+1}^N(m')} \prod_{j=k+1}^{N} p_{\text{dem}}(c_j^0) p_{\text{dem}}(c_j^1) \right).\end{aligned}$$

Now we introduce the indicator functions

$$\begin{aligned}\sigma_k(m,m') = &\left(\sum_{\mathbf{x}_1^k \in \mathcal{R}_1^k} I_{\mathcal{C}_1^{k-1}(m)}(\mathbf{x}_1^{k-1}) \prod_{j=1}^{k} p_{\text{dem}}(x_j^0) p_{\text{dem}}(x_j^1) \right) \\ &\big(p_{\text{dem}}(x_k^0 = c_{m\to m'}^0) p_{\text{dem}}(x_k^1 = c_{m\to m'}^1) I_c(m,m') \big) \\ &\left(\sum_{\mathbf{x}_{k+1}^N \in \mathcal{R}_{k+1}^N} I_{\mathcal{C}_{k+1}^N(m')}(\mathbf{x}_{k+1}^N) \prod_{j=k+1}^{N} p_{\text{dem}}(x_j^0) p_{\text{dem}}(x_j^1) \right)\end{aligned}$$

and gather the marginal probabilities:

$$\begin{aligned}\sigma_k(m,m') &= \sum_{\mathbf{x}_1^k\in\mathcal{R}_1^k}\sum_{\mathbf{x}_{k+1}^N\in\mathcal{R}_{k+1}^N} I_{\mathcal{C}_1^{k-1}(m)}(\mathbf{x}_1^{k-1})I_c(m,m')I_{\mathcal{C}_{k+1}^N(m')}(\mathbf{x}_{k+1}^N)\\ &\quad p_{\mathrm{dem}}(x_k^0=c_{m\to m'}^0)p_{\mathrm{dem}}(x_k^1=c_{m\to m'}^1)\prod_{j\neq k}p_{\mathrm{dem}}(x_j^0)p_{\mathrm{dem}}(x_j^1)\\ &= \sum_{\mathbf{x}\in\mathcal{R}:(c_{m\to m'}^0,c_{m\to m'}^1)} I_{\mathcal{C}_1^{k-1}(m)}(\mathbf{x}_1^{k-1})I_c(m,m')I_{\mathcal{C}_{k+1}^N(m')}(\mathbf{x}_{k+1}^N)\\ &\quad \left(\prod_{j=1}^N p_{\mathrm{dem}}(x_j^0)p_{\mathrm{dem}}(x_j^1)\right).\end{aligned}$$

By defining $I_{\mathcal{C}:m\to m'}(\mathbf{x}) := I_{\mathcal{C}_1^{k-1}(m)}(\mathbf{x}_1^{k-1})I_c(m,m')I_{\mathcal{C}_{k+1}^N(m')}(\mathbf{x}_{k+1}^N)$, we finally obtain

$$\sigma_k(m,m') = \sum_{\mathbf{x}\in\mathcal{R}} I_{\mathcal{C}:m\to m'}(\mathbf{x})\underbrace{\left(\prod_{j=1}^N p_{\mathrm{dem}}(x_j^0)p_{\mathrm{dem}}(x_j^1)\right)}_{p_{\mathrm{dem}}(\mathbf{x})}.$$

The posterior pmfs $p_{\mathrm{dec}}(x_k^l)$ are then evaluated by summing up the $\sigma_k(m,m')$ over all the trellis transitions (m,m') that yield as output the value of the bit x_k^l we are considering:

$$\begin{aligned}p(x_k^l) &\propto \sum_{(m,m'):c_k^l}\sigma_k(m,m')\\ &= \sum_{(m,m'):c_k^l}\left(\sum_{\mathbf{x}\in\mathcal{R}} I_{\mathcal{C}:m\to m'}(\mathbf{x})p_{\mathrm{dem}}(\mathbf{x})\right)\\ &= \sum_{(m,m')}\left(\sum_{\mathbf{x}\in\mathcal{R}:c_k^l} I_{\mathcal{C}:m\to m'}(\mathbf{x})p_{\mathrm{dem}}(\mathbf{x})\right)\\ &= \sum_{\mathbf{x}\in\mathcal{R}:c_k^l}\left(\sum_{(m,m')} I_{\mathcal{C}:m\to m'}(\mathbf{x})p_{\mathrm{dem}}(\mathbf{x})\right)\\ &= \sum_{\mathbf{x}\in\mathcal{R}:c_k^l}\left(I_{\mathcal{C}}(\mathbf{x})p_{\mathrm{dem}}(\mathbf{x})\right).\end{aligned}$$

Therefore the output of the BCJR decoder can be seen as the result of the projection

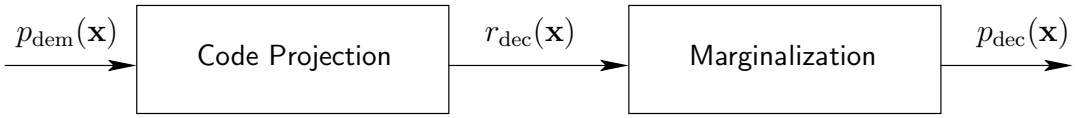

Figure 3.6: Interpretation of the BCJR decoder.

onto the set of factorizable densities $\mathcal{E}_{\mathcal{F}}$ (which is represented by the marginalization) of the projection of $p_{\text{dem}}(\mathbf{x})$ onto the set $\mathcal{L}_{\mathcal{C}}$ of code-compatible distributions (represented by the indicator function that zeros out all the non-codewords' probabilities).

The output distribution, which consists of the a posteriori code bit pmfs, is written as

$$p_{\text{dec}}(\mathbf{x}) = \prod_{k,l} p(x_k^l).$$

The operation of the decoding sub-block is illustrated in Fig. 3.6. The intermediate distribution r_{dec}, which is the projection of p_{dem} onto $\mathcal{L}_{\mathcal{C}}$ is given by

$$r_{\text{dec}}(\mathbf{x}) = \frac{p_{\text{dem}}(\mathbf{x}) I_{\mathcal{C}}(\mathbf{x})}{\sum_{\mathbf{x}} p_{\text{dem}}(\mathbf{x}) I_{\mathcal{C}}(\mathbf{x})}.$$

3.5.3 Example

We will again illustrate the procedure described above by an example. The setup is the same as in the example given in section 3.4.4, that is we consider length-two bit-sequences and a repetition code. The setup is illustrated in Fig. 3.7. It shows the probability simplex, which represents the manifold of all pmfs with four outcomes, and two submanifolds: the code manifold $\mathcal{L}_{\mathcal{C}}$ and the manifold of the factorizable distributions, $\mathcal{E}_{\mathcal{F}}$.

The manifold of factorizable densities is in our case two-dimensional. This can be understood by considering the fact that each pmf of the family is a product of two bit-pmfs, $p_{\mathcal{F}}(\mathbf{x}) = \prod_i p_i(x_i) = p_0(x_0)p_1(x_1)$. Each bit-pmf is actually characterized fully by one parameter (either the probability $P(x_i = 0)$ or $P(x_i = 1)$). Since $p_{\mathcal{F}}(\mathbf{x})$ is the product of two pmfs, each described by one parameter, we need two parameters to fully

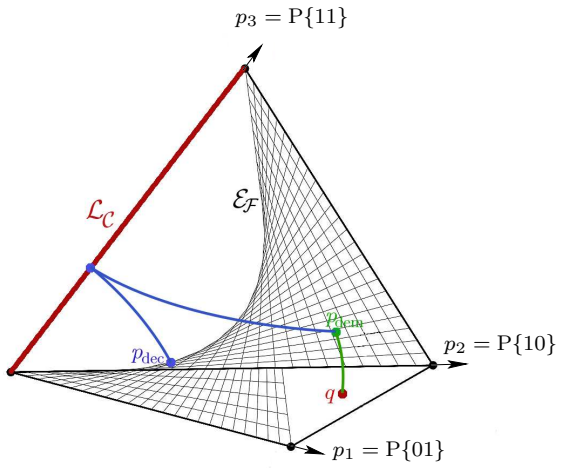

Figure 3.7: The practical BICM-decoding algorithm.

determine it.

The demapper takes as input the observed transmit symbols, calculates the observed distribution $q(\tilde{\mathbf{x}}) = p(\tilde{\mathbf{x}}|\mathbf{y})$ and implicitly projects it onto the family of factorizable distributions, yielding $p_{\mathrm{dem}}(\tilde{\mathbf{x}})$. Then the BCJR-decoder projects $p_{\mathrm{dem}}(\mathbf{x})$ onto the code-manifold $\mathcal{L}_\mathcal{C}$ and again onto the factorizable distributions. The resulting pmf is $p_{\mathrm{dec}}(\mathbf{x})$, the output of the decoder.

3.6 Iterative Demodulation

In order to compensate for the loss of information (and therefore performance) that occurs due to the demapper's assumption that the code bits are independent, [8] describes an iterative algorithm. The idea is to "feed back" the information about the code bits generated by the decoder sub-block to the demapper, which can use it as *a priori* information about the bits to be demapped. In this way, the new pmf generated by the demapper should represent the actual coded bits transmitted in a better way. This new pmf is then again used as input for the BCJR-decoder, and the next iteration begins.

Iterative demodulation is commonly also referred to as "turbo demodulation", be-

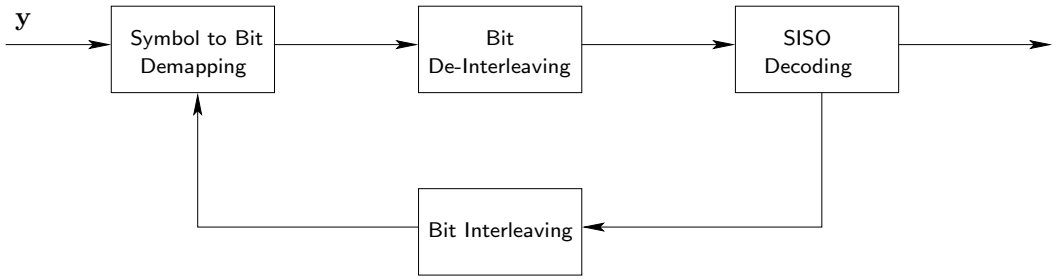

Figure 3.8: Iterative BICM receiver

cause of the analogy to a turbocharger that uses the pressure of the exhaust gases to compress the air in the combustion chamber of an engine, to improve its performance.

We need to make three essential changes to the non-iterative setup described so far: the demapper has to be modified to take bit probabilities as additional input besides the received symbols. Second, instead of directly using the a posteriori bit probabilities generated by one block as the input of the other block, we use so-called "extrinsic probabilities" because this leads to better convergence behavior. And finally, a feed-back branch has to be introduced to supply the demapper with bit probabilities generated by the decoder. The iterative receiver is depicted in Fig. 3.8

The probability distributions in the following will be denoted with a superscript (e.g., "I") that stands for the number of the iteration step in which it is produced.

3.6.1 Extrinsic Probabilities

Let's first define extrinsic probabilities and then discuss the meaning of the definition.

Definition 3.6.1. The *extrinsic probability* of an encoded bit b is the ratio of the a posteriori probability of b given the *constraints* imposed by the considered sub-block and the a priori probabilities over the a priori probability of b. Informally, we can write

$$e(b) = \frac{\text{a posteriori probability (b)}}{\text{a priori probability (b)}}.$$

We will give the explicit expressions for the extrinsic probabilities of the respective sub-block in the corresponding sections.

3.6.2 The Demapping Sub-Block

The demapper described for non-iterative decoding calculated the probabilities for the encoded bits without considering the code structure. The APPs of the code bits are (see equation (3.15))

$$p(\tilde{x}_k^l|y_k) = \sum_{\tilde{\mathbf{x}}_k \sim \tilde{x}_k^l} p(s_k(\tilde{\mathbf{x}}_k)|y_k),$$

and for uniformly distributed code bits (see (3.16)),

$$p(\tilde{x}_k^l|y_k) \propto \sum_{\tilde{\mathbf{x}}_k \sim \tilde{x}_k^l} p(y_k|s_k(\tilde{\mathbf{x}}_k)).$$

For the iterative receiver, the demapper has to incorporate a priori knowledge about the coded bits, therefore we drop the assumption the code bits being uniformly distributed. This means that Bayes' rule applied to (3.15) yields, instead of (3.16), the expression

$$p(\tilde{x}_k^l|y_k) \propto \sum_{\tilde{\mathbf{x}}_k \sim \tilde{x}_k^l} p(y_k|s_k(\tilde{\mathbf{x}}_k)) \prod_n p_{\text{dec}}^{(I)}(\tilde{x}_k^n),$$

where $n \in (1, 2, \ldots, B-1)$, and $p_{\text{dec}}^{(i)}(\tilde{x}_k^n)$ being the a priori probabilities supplied by the decoder sub-block in iteration I.

To obtain the extrinsic probability of $\tilde{x}_k^l$ we need to divide by its a priory probability $p_{\text{dec}}^{(i)}(\tilde{x}_k^l)$, and get

$$e(\tilde{x}_k^l|y_k) \propto \sum_{\tilde{\mathbf{x}}_k \sim \tilde{x}_k^l} p(y_k|s_k(\tilde{\mathbf{x}}_k)) \prod_{n \neq l} p_{\text{dec}}^{(i)}(\tilde{x}_k^n).$$

The a priori information that goes into the previous two equations comes from the decoder. As already mentioned, the extrinsic probabilities are used instead of the a posteriori probabilities. Inserting these extrinsic probabilities and normalizing, the expressions become

$$p_{\text{dem}}^{(I+1)}(\tilde{x}_k^l) = \frac{\sum_{\tilde{\mathbf{x}}_k \sim \tilde{x}_k^l} p(y_k|\tilde{\mathbf{x}}_k) \prod_n e_{\text{dec}}^{(I)}(\tilde{x}_k^n)}{\sum_{\tilde{\mathbf{x}}_k} p(y_k|\tilde{\mathbf{x}}_k) \prod_n e_{\text{dec}}^{(I)}(\tilde{x}_k^n)} \tag{3.21}$$

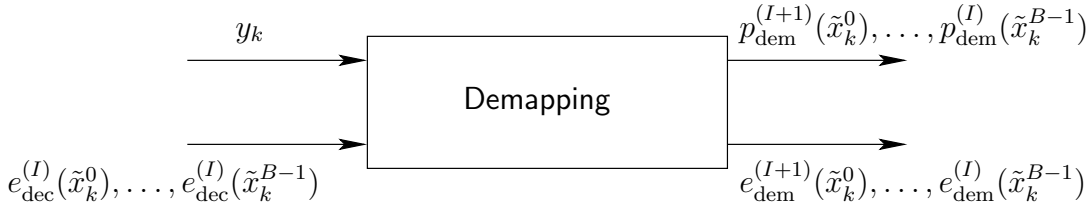

Figure 3.9: Local description of demapping sub-block.

and

$$e_{\text{dem}}^{(I+1)}(\tilde{x}_k^l) = \frac{\sum_{\tilde{\mathbf{x}}_k \sim \tilde{x}_k^l} p(y_k|\tilde{\mathbf{x}}_k) \prod_{n \neq k} e_{\text{dec}}^{(I)}(\tilde{x}_k^n)}{\sum_{\tilde{\mathbf{x}}_k} p(y_k|\tilde{\mathbf{x}}_k) \prod_{n \neq k} e_{\text{dec}}^{(I)}(\tilde{x}_k^n)}. \tag{3.22}$$

The equations (3.21) and (3.22) describe the local behavior of the demapping sub-block for the iterative receiver. For each symbol index k it takes $2B + 1$ inputs: the received symbol y_k and the 2B extrinsic probabilities $e_{\text{dec}}^{(I)}(\tilde{x}_k^l = b)$ for $0 \leq l \leq B - 1$,$b \in 0, 1$, obtained after the Ith iteration. It produces $4B$ outputs, $2B$ a posteriori probabilities and $2B$ extrinsic probabilities of the encoded bits. Fig. 3.9 illustrates this behavior.

The global behavior in the iterative case is slightly more complicated to describe than in the non-iterative case. In the non-iterative case, the demapper's only function was the implicit projection of the observed distribution $p(\mathbf{x}|\mathbf{y})$ onto the factorizable distributions. This projection is the geometric interpretation of the approximation of $p(\mathbf{x}|\mathbf{y})$ by its marginal bit probabilities.

In the iterative case, there is the additional a priori pmf

$$q_{\text{dec}}(\tilde{\mathbf{x}}) = \prod_{m,n} e_{\text{dec}}^{(I)}(x_{\sigma,m}^n)$$

for the sequence of interleaved coded bits coming from the decoder, which is incorporated by the demapper.

We will now define an "intermediate" pmf $r_{\text{dem}}^{(I+1)}$

$$r_{\text{dem}}^{(I+1)}(\tilde{\mathbf{x}}) = \frac{p(\tilde{\mathbf{x}}|\mathbf{y})q_{\text{dec}}^{(I)}(\tilde{\mathbf{x}})}{\sum_{\tilde{\mathbf{x}}'} p(\tilde{\mathbf{x}}'|\mathbf{y})q_{\text{dec}}^{(I)}(\tilde{\mathbf{x}}')}.$$

Then we define $p_{\text{dem}}^{(I+1)}$ as the projection of $r_{\text{dem}}^{(I+1)}$ onto the set of factorizable distributions:

$$p_{\text{dem}}^{(I+1)}(\tilde{x}_m^n) = \frac{\prod_{m,n}\sum_{\tilde{\mathbf{x}}'\sim\tilde{x}_m^n} r_{\text{dem}}^{(I+1)}(\tilde{\mathbf{x}}')}{\sum_{\tilde{\mathbf{x}}'}\prod_{m,n}\sum_{\tilde{\mathbf{x}}''\sim\tilde{x}_m'^n} r_{\text{dem}}^{(I+1)}(\tilde{\mathbf{x}}'')} = \frac{\prod_{m,n}\sum_{\tilde{\mathbf{x}}'\sim\tilde{x}_m^n} p(\tilde{\mathbf{x}}'|\mathbf{y})q_{\text{dec}}^{(I)}(\tilde{\mathbf{x}}')}{\sum_{\tilde{\mathbf{x}}'}\prod_{m,n}\sum_{\tilde{\mathbf{x}}''\sim\tilde{x}_m'^n} p(\tilde{\mathbf{x}}''|\mathbf{y})q_{\text{dec}}^{(I)}(\tilde{\mathbf{x}}'')}.$$

Now let us show that the resulting pmf is the output of the demapper, that is, it is the factorizable distribution consisting of the individual bit pmfs given in equations (3.21) and (3.22): the upper term can be further written as

$$\begin{aligned}\prod_{m,n}\sum_{\tilde{\mathbf{x}}'\sim\tilde{x}_m^n} p(\tilde{\mathbf{x}}'|\mathbf{y})q_{\text{dec}}^{(I)}(\tilde{\mathbf{x}}') &\propto \prod_{m,n}\sum_{\tilde{\mathbf{x}}'\sim\tilde{x}_m^n}\prod_k p(y_k|\tilde{\mathbf{x}}'_k)\prod_l e_{\text{dec}}^{(I)}(\tilde{\mathbf{x}}'^l_k)\\ &= \prod_{m,n}\sum_{\tilde{\mathbf{x}}'_1}\cdots\sum_{\tilde{\mathbf{x}}'\sim\tilde{x}_m^n}\cdots\sum_{\tilde{\mathbf{x}}'_L}\left(\prod_k p(y_k|\tilde{\mathbf{x}}'_k)\prod_l e_{\text{dec}}^{(I)}(\tilde{\mathbf{x}}'^l_k)\right)\\ &= \prod_{m,n}\sum_{\tilde{\mathbf{x}}'\sim\tilde{x}_m^n} p(y_m|\tilde{\mathbf{x}}'_m)\prod_l e_{\text{dec}}^{(I)}(\tilde{\mathbf{x}}'^l_m),\end{aligned}$$

the lower term as

$$\prod_{m,n}\sum_{\tilde{\mathbf{x}}'} p(\tilde{\mathbf{x}}'|\mathbf{y})q_{\text{dec}}^{(I)}(\tilde{\mathbf{x}}') \propto \prod_{m,n}\sum_{\tilde{\mathbf{x}}'_m} p(y_m|\tilde{\mathbf{x}}'_m)\prod_l e_{\text{dec}}^{(I)}(\tilde{\mathbf{x}}'^l_m),$$

with equal proportionality constants. Therefore the marginal probability for a bit $\tilde{x}_m^n$ of $p_{\text{dem}}(\tilde{\mathbf{x}})$ is equal to

$$p_{\text{dem}}^{(I+1)}(\tilde{x}_m^n) = \frac{\sum_{\tilde{\mathbf{x}}'\sim\tilde{x}_m^n} p(y_m|\tilde{\mathbf{x}}'_m)\prod_l e_{\text{dec}}^{(I)}(\tilde{\mathbf{x}}'^l_m)}{\sum_{\tilde{\mathbf{x}}'_m} p(y_m|\tilde{\mathbf{x}}'_m)\prod_l e_{\text{dec}}^{(I)}(\tilde{\mathbf{x}}'^l_m)}.$$

This is the same expression as we derived for the marginal bit probability in (3.21). In this way we have found a probability distribution that describes the output of the

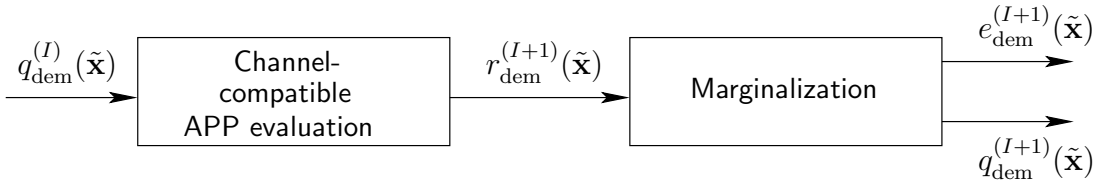

Figure 3.10: Global interpretation of the demapping sub-block.

demapper. The corresponding distribution describing the extrinsic bit probabilities can then be obtained as

$$q_{\text{dem}}^{(I+1)}(\tilde{\mathbf{x}}) = \frac{p_{\text{dem}}^{(I+1)}(\tilde{\mathbf{x}})/q_{\text{dec}}^{(I)}(\tilde{\mathbf{x}})}{\sum_{\tilde{\mathbf{x}}} p_{\text{dem}}^{(I+1)}(\tilde{\mathbf{x}})/q_{\text{dec}}^{(I)}(\tilde{\mathbf{x}})}.$$

So, for the iterative receiver, the global interpretation of the demapping sub-block is that it performs two steps, illustrated in Fig. 3.10. The first step is an evaluation of a posteriori coded bit probabilities that are compatible with the input symbols as well as the a priori probabilities. It could be regarded as a projection of the input distribution onto a manifold of "channel-likelihood-compatible distributions", which will be defined below. The second step is a projection onto the family of factorizable distributions.

Definition 3.6.2. The set of *channel-likelihood-compatible distributions* is given by

$$\mathcal{E}_{\mathcal{Y}} = \{p | p(\tilde{\mathbf{x}}) = c \cdot p(\tilde{\mathbf{x}}|\mathbf{y}) q(\tilde{\mathbf{x}})\}, \quad \text{where } q(\tilde{\mathbf{x}}) \text{ is a factorizable distribution,}$$

with a normalization constant c such that p is a valid pmf.

It can be shown that $\mathcal{E}_{\mathcal{Y}}$ is an exponential family [8].

3.6.3 BCJR Decoding Sub-Block

The interpretation of the BCJR algorithm as two successive projections has already been shown in section 3.5.2 for the non-iterative case. The input distribution was the factorizable distribution consisting of the a posteriori bit probabilities, coming from the demapper.

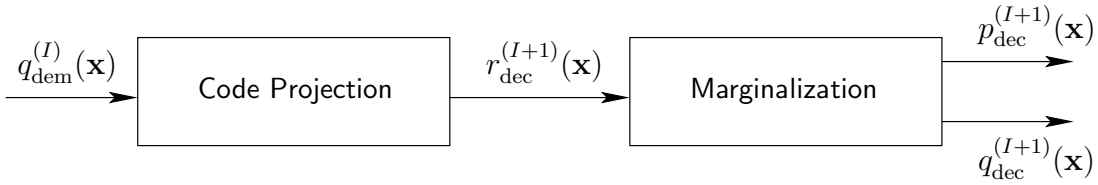

Figure 3.11: Global interpretation of the decoding sub-block.

The only modification for the iterative receiver is to substitute the input distribution $p_{\text{dem}}(\mathbf{x})$ by $q_{\text{dem}}(\mathbf{x})$, the factorizable distribution consisting of the extrinsic bit probabilities $e(x_m^n)$. All the derivations of section 3.5.2 stay valid. The end result stays the same: the decoding sub-block can be interpreted as two projections: first, the projection of the input distribution q_{dem} onto the code-manifold. Second, the projection of the intermediate distribution obtained in the first step onto the factorizable distributions.

The first output distribution, which consists of the a posteriori code bit pmfs is written as

$$p_{\text{dec}}^{(I+1)}(\mathbf{x}) = \prod_{k,l} p(x_k^l).$$

The second output distribution, consisting of the extrinsic code bit probabilities, is given by

$$q_{\text{dec}}^{(I+1)}(\mathbf{x}) = \frac{p_{\text{dec}}^{(I+1)}(\mathbf{x}) / q_{\text{dem}}^{(I+1)}(\mathbf{x})}{\sum_{\mathbf{x}} p_{\text{dec}}^{(I+1)}(\mathbf{x}) / q_{\text{dem}}^{(I+1)}(\mathbf{x})}.$$

The operation of the decoding sub-block is illustrated in Fig. 3.11. The intermediate distribution $r_{\text{dec}}^{(I+1)}$, which is the projection of $q_{\text{dem}}^{(I+1)}$ onto $\mathcal{L}_{\mathcal{C}}$ is given by

$$r_{\text{dec}}^{(I+1)} = \frac{q_{\text{dem}}^{(I+1)}(\mathbf{x}) I_{\mathcal{C}}(\mathbf{x})}{\sum_{\mathbf{x}} q_{\text{dem}}^{(I+1)}(\mathbf{x}) I_{\mathcal{C}}(\mathbf{x})}.$$

4

Log-Likelihood Ratio Clipping

4.1 Introduction

In practical implementations, log-likelihood ratios (LLRs) are often preferred over bit probabilities as the carrier of soft information. As the theoretical range for an LLR is from $-\infty$ to ∞, it is obvious that these values have to be clipped at some threshold. Limiting the LLRs to smaller values can reduce implementation complexity and thus cost, because then a smaller number of bits is required for representing an LLR value.

Furthermore it has been shown, that the MIMO sphere decoder described for example in [5] can benefit from notable complexity reduction if it needs to deliver only clipped LLR values.

In the following chapter, we will show that bit LLR clipping has an information-geometric interpretation as projection onto a clipping-manifold, which is a submanifold of the manifold of factorizable distributions. To this end, we first introduce bit LLRs. Then we show that clipping of a single bit amounts to an I-projection that is obtained by solving an optimization problem, with the constraint that the clipped distribution has a certain maximum KL divergence from the uniform distribution. This "binary" case is generalized to the case of bit-sequences, for which we define "sequence LLRs".

The interesting result here is that clipping of those LLRs does not change the maximum of the sequence pmf, therefore the MAP estimate of the transmit sequence is unaffected by the clipping.

The practical demodulation algorithm described in the previous chapter works with factorizable distributions. We will therefore finally show that under the assumption of a factorizable distributions, the sequence LLR clipping for which we have found an information geometric interpretation, can also be used for the description of bit-wise LLR clipping, where the bits are from a bit sequence.

4.2 Log-Likelihood Ratios

The a posteriori log-likelihood ratio[1] of a bit b, given some observation $\mathbf{y}$ is defined as

$$\xi(b) = \ln \frac{\mathrm{P}\{b = 1|\mathbf{y}\}}{\mathrm{P}\{b = 0|\mathbf{y}\}}.$$

An LLR can take on values ranging from $-\infty$ to ∞. The sign of the LLR contains the information about the value of the bit, i.e. a positive LLR means that the probability of the bit to be "1" is greater than the probability that it is "0". The magnitude of the LLR stands for the reliability of this statement, larger LLR values mean more certainty about the bit value. Values near zero mean great uncertainty, and $\xi = 0$ corresponds to a uniform bit pmf and stands for non-existing knowledge about the bit value.

The representation of soft information by such "bit LLRs" is fully equivalent to using a posteriori bit probabilities. Let's consider the probability $p(1)$ that bit b given $\mathbf{y}$ is equal to 1, $p(1) = \mathrm{P}\{b = 1|\mathbf{y}\}$. In this case, obviously, $p(0) = \mathrm{P}\{b = 0|\mathbf{y}\} = 1 - p(1)$, which means any of $p(0)$ or $p(1)$ is a full representation of the soft information about b.

[1] Despite the name *likelihood*-ratio, we use the term in the broader sense of the ratio of two probabilities instead of likelihood functions

Given one of the probabilities $p(1)$ or $p(0)$ we obtain the LLR ξ as

$$\xi(b) = \ln \frac{\mathrm{P}\{b=1|\mathbf{y}\}}{\mathrm{P}\{b=0|\mathbf{y}\}} = \ln \frac{p(1)}{p(0)} = \ln \frac{p(1)}{1-p(1)} = \ln \frac{1-p(0)}{p(0)}.$$

Going into the other direction, we can calculate the bit probabilities given some LLR-value ξ as

$$p(1) = \frac{e^{\xi}}{1+e^{\xi}} = \frac{1}{1+e^{-\xi}}$$

and

$$p(0) = 1 - \frac{e^{\xi}}{1+e^{\xi}} = \frac{1}{1+e^{\xi}}.$$

4.3 Sequence LLR Clipping

Soft sequence detection is the calculation of the a posteriori pmf $p(\mathbf{x}|\mathbf{y})$ of the bit sequences $\mathbf{x}$ given a received sequence $\mathbf{y}$. One could think of defining sequence LLRs as some function of the a posteriori sequence probabilities, analog to bit LLRs. The problem is that there are many possibilities for such a definition, and it is not clear which of those are meaningful. Given a set of sequences $\mathbf{x}$, over which a probability distribution $p(\mathbf{x}) : \mathbf{p} = (p^0, p^1, \ldots, p^{N-1})$, $p^i = \mathrm{P}\{\mathbf{x} = \mathbf{x}_i\}$ is defined, one possible LLR-definition would be

$$\Lambda^i = \log \frac{\mathrm{P}\{\mathbf{x}=\mathbf{x}_i\}}{\mathrm{P}\{\mathbf{x}=\mathbf{x}_0\}} = \log \frac{p^i}{p^0}, \qquad i = 1, \ldots, N-1 \tag{4.1}$$

which means that one sequence probability (p^0) is arbitrarily picked as a reference. As there is no reason for one sequence to be singled out, this sort of definition is problematic. But we will be using it in the next section about bit LLR clipping where it actually leads to meaningful results.

We use the following approach to circumvent this problem: bit LLRs can be regarded as special case of LLRs of a length-one "sequence". We will show that it is

possible to describe bit LLR clipping as a projection of the corresponding bit probability distribution onto a clipping manifold, which contains all Bernoulli-distributions that have a certain maximum KL divergence from the uniform distribution. Then we generalize this one-dimensional clipping manifold to higher dimensions, thus considering sequences of a length greater than one.

Consider a bit-distribution $p(x)$, $x \in \{0, 1\}$, described by the length-two probability vector $\mathbf{p} = (p(0), p(1))$, with $p(0) + p(1) = 1$. The uniform bit distribution is denoted as $\bar{p}(x)$, with the corresponding probability vector $\bar{\mathbf{p}} = (1/2, 1/2)$.

First we will show that clipping the LLR values for one bit is equivalent to restricting the KL divergence of the corresponding bit distribution to the uniform distribution. Let γ be the clipping level (clipping threshold), i.e we restrict the LLR values to the interval

$$-\gamma \leq \xi_i \leq \gamma.$$

For the probability of the "1"-bit this means

$$-\gamma \leq \ln \frac{p(1)}{1 - p(1)} \leq \gamma,$$

which is equivalent to

$$\frac{1}{1 + e^{\gamma}} \leq p(1) \leq \frac{1}{1 + e^{-\gamma}}.$$

This means that we have found two bounding values in the "probability regime" that correspond to the clipping threshold in the "LLR-regime". Because $p(0) = 1 - p(1)$, we have furthermore found the two distributions that correspond to the clipping threshold,

$$\mathbf{p}^U = \left(\frac{1}{1 + e^{\gamma}}, \frac{1}{1 + e^{-\gamma}} \right), \quad \mathbf{p}^L = \left(\frac{1}{1 + e^{-\gamma}}, \frac{1}{1 + e^{\gamma}} \right)$$

Finally, we explicitly write down the KL divergences of each of the two distributions

to the uniform distribution:

$$D(p^U\|\bar{p}) = \frac{1}{1+e^{\gamma}} \ln \frac{2}{1+e^{\gamma}} + \frac{1}{1+e^{-\gamma}} \ln \frac{2}{1+e^{-\gamma}}$$
$$D(p^L\|\bar{p}) = \frac{1}{1+e^{-\gamma}} \ln \frac{2}{1+e^{-\gamma}} + \frac{1}{1+e^{\gamma}} \ln \frac{2}{1+e^{\gamma}}$$

These divergences are identical. We conclude that for one single bit, LLR clipping is equivalent to restricting the KL divergence of the corresponding bit distribution from the uniform distribution. This is illustrated in Fig. 4.1. The upper part of the figure shows the LLR values and the clipping thresholds $-\gamma$ and γ. The middle part is the one-dimensional manifold of binary distributions p, parametrized by the coordinate p_1. The section of the axis between the probability values p_1^L and p_1^U is the clipping manifold

$$\mathcal{M}_{\text{clip}} = \big\{p(x) : D(p\|\bar{p}) \leq D^*\big\}.$$

The lower part of the figure shows the KL divergence $D(p\|\bar{p})$, which is a convex function. This representation makes it easy to see that restricting $D(p\|\bar{p})$ is equivalent to clipping the LLR value ξ.

Let us now calculate the I-projection of the bit pmf p onto the clipping manifold. This calculation is an optimization problem that can be approached using Lagrangian multipliers with Kuhn-Tucker conditions. The problem is defined as follows: we are looking for the distribution p_{clip} that lies inside the clipping manifold $\mathcal{M}_{\text{clip}}$ and has minimum KL divergence $D(p_{\text{clip}}\|p)$ to the original pmf p. This can be formulated by the following expressions:

$$p_{\text{clip}} = \arg\min_{q} D(q\|p), \tag{4.2a}$$
$$D(q\|\bar{p}) \leq D^*, \tag{4.2b}$$
$$q(0) + q(1) = 1. \tag{4.2c}$$

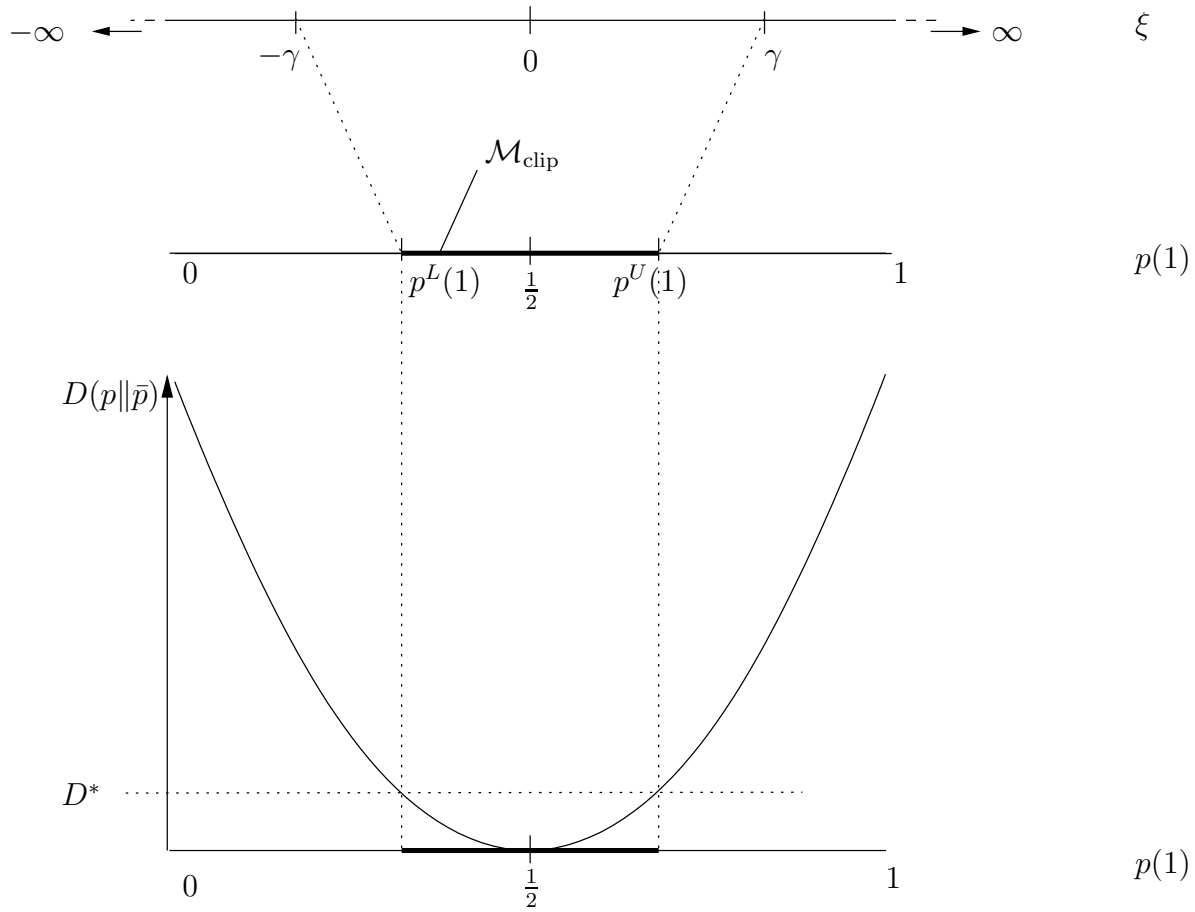

Figure 4.1: Relations between LLR clipping and restricting the KL divergence to the uniform distribution.

The Lagrangian function is therefore

$$L = q(0)\ln\frac{q(0)}{p(0)} + q(1)\ln\frac{q(1)}{p(1)} + \lambda_1\big(q(0)\ln 2q(0) + q(1)\ln 2q(1) - D^*\big) + \lambda_2\big(1 - q(0) - q(1)\big). \quad (4.3)$$

The solution to the optimization problem gives the resulting distribution p_{clip} as

$$\mathbf{p}_{\text{clip}} = c(\alpha)\cdot\big(p(0)^\alpha, p(1)^\alpha\big),$$

with a constant α that depends on the "probability clipping threshold" D^*, and a second constant $c(\alpha)$. The latter is obtained by inserting into (4.2c),

$$c(\alpha) = \frac{1}{p(0)^\alpha + p(1)^\alpha},$$

which means normalizing.

To calculate α, we insert into (4.2b). Due to the convexity of $\mathcal{M}_{\text{clip}}$, p_{clip} lies at the boundary of $\mathcal{M}_{\text{clip}}$, hence we look for the α where (4.2b) attains the bound. This yields

$$\begin{aligned} D(p\|\bar{p}) &= p(0)\ln 2p(0) + p(1)\ln 2p(1) = \\ &= c(\alpha)p(0)^\alpha \ln 2c(\alpha)p(0)^\alpha + c(\alpha)p(1)^\alpha \ln 2c(\alpha)p(1)^\alpha = D^*. \end{aligned}$$

Unfortunately there is no analytical solution, so α has to be calculated numerically. Let us now consider two extreme cases. For $\alpha = 0$, we get

$$\alpha = 0: \quad \mathbf{p}_{\text{clip}} = \big(c(\alpha), c(\alpha)\big) = \left(\frac{1}{2}, \frac{1}{2}\right) = \bar{\mathbf{p}},$$

the uniform distribution. Obviously, this is the case of extreme clipping, that is, setting

the LLR clipping threshold or, equivalently, D^* to zero. Setting $\alpha = 1$ yields

$$\mathbf{p}_{\text{clip}} = \big(p(0), p(1)\big) = \mathbf{p}.$$

This can be seen to be the case of no clipping, because the clipped distribution is equal to the original one.

The binary case discussed above can be straightforwardly generalized to sequences of bits. For a bit sequence of length L there are $N = 2^L$ different possible sequences $\mathbf{x}_i, i = 0, \ldots, N-1$. The a posteriori distribution $p(\mathbf{x}|\mathbf{y})$ has therefore N outcomes and is characterized by the probability vector

$$\mathbf{p} = (p^0, p^1, \ldots p^{N-1}).$$

The uniform distribution is now

$$\bar{\mathbf{p}} = \Big(\underbrace{\frac{1}{N}, \ldots \frac{1}{N}}_{n \text{ times}}\Big)$$

There are $N-1$ corresponding sequence LLRs

$$\Lambda^i = \log \frac{p^i}{p^0}, \qquad i = 1, \ldots, N-1.$$

We clip the sequence LLRs, i.e. restrict them to

$$-\gamma \leq \Lambda^i \leq \gamma,$$

and argue that this is again equivalent to restricting the KL divergence of the sequence pmf to some clipping threshold,

$$D(p\|\bar{p}) \leq D^*.$$

This leads to a slightly modified version the equations (4.2).

$$p_{\text{clip}} = \arg\min_{q} D(q\|p), \tag{4.4a}$$

$$D(q\|\bar{p}) \leq D^*, \tag{4.4b}$$

$$\sum_i q^i = 1. \tag{4.4c}$$

Solving this optimization problem is now more elaborate than in the binary case, the Lagrangian function is

$$L = \sum_{k=0}^{N-1} q^k \ln \frac{q^k}{p^k} + \lambda_1 \left(\sum_{l=0}^{N-1} q^l \ln(N q^l) - D^* \right) + \lambda_2 \left(\sum_{m=0}^{N-1} q^m - 1 \right) \tag{4.5}$$

The solution to the problem is, similar to the binary case,

$$\mathbf{p}_{\text{clip}} = c(\alpha) \cdot \left(\left(p^0\right)^\alpha, \ldots, \left(p^{N-1}\right)^\alpha \right).$$

The most interesting aspect of this solution is that the MAP estimate of the sequence $\mathbf{x}$ given the observation $\mathbf{y}$ is not influenced by the clipping:

$$\mathbf{x}_{\text{MAP}} = \arg\max_{\mathbf{x}} p(\mathbf{x}|\mathbf{y}) = \arg\max_{\mathbf{x}} p_{\text{clip}}(\mathbf{x}|\mathbf{y}).$$

The case $N = 3$ is illustrated in Fig. 4.3. The clipping in a way "smoothes" the corners of the probability simplex, which stands for non-clipped distributions. This can be understood by considering that the restriction of the KL divergence of a distribution from the uniform distribution means prohibiting very high probability values of single sequences.

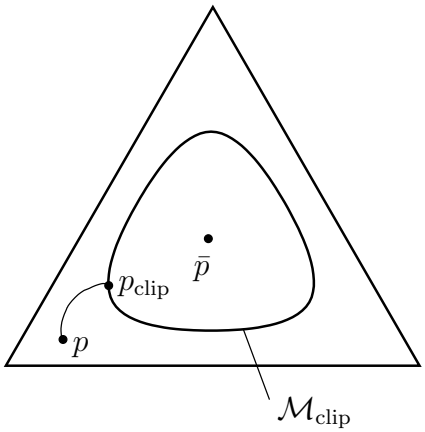

Figure 4.2: Sequence LLR clipping for the case of three different possible sequences.

4.4 Bit LLR Clipping

In the discussion of the practical BICM receiver in the previous chapter, we already stated that in order to handle bit interleaving, the probability distributions manipulated by the receiver have to be factorizable into individual bit probabilities, or approximated by factorizable distributions, respectively.

When a sequence probability can be factorized into bit probabilities, we can also split up sequence LLRs into individual bit LLRs. Using the fact that the sequence probabilities considered are factorizable, the definition for sequence LLRs (4.1) becomes

$$\Lambda^i = \log \frac{p^i}{p^0} = \log \frac{\prod_k p_k^i}{\prod_k p_k^0} = \sum_k \log \frac{p_k^i}{p_k^0} = \sum_k \delta(b_k^i = 1)\xi_k.$$

In this expression and for the rest of this section, the subscripts k denote the position of the bit in the sequence, i.e $p_k^i = \mathrm{P}\{x_k = [\mathbf{x}_i]_k\}$, and $p^i{=}p(\mathbf{x}_i)$. The LLR of the bit at position k is $\xi_k = \log \frac{p_k=1}{p_k=0}$. The reference sequence $\mathbf{x}_0$ has been chosen to be the all zeros codeword. We see that in this case the ith sequence LLR Λ^i is the sum of the individual LLRs of the bits that are "1" in the sequence $\mathbf{x}_i$.

Let us again consider the example from section 3.5.3. We have sequences of length two, and a family of corresponding probability distributions $p_{\boldsymbol{\alpha}}(\mathbf{x})$, $\mathbf{p} = (p^0, p^1, p^2, p^3)$, where p^i is the probability of sequence with decimal value i, which is illustrated by

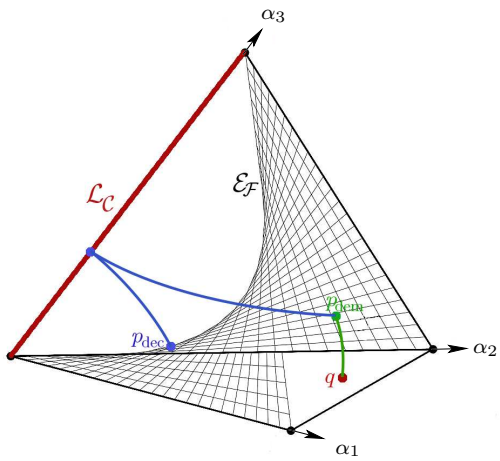

Figure 4.3: Probability simplex with factorizable distributions.

Fig. 4.3. The twisted surface in the figure represents the manifold $\mathcal{E}_\mathcal{F}$ of factorizable distributions, i.e. distributions that can be written as the product of marginal bit-distributions, $p_\mathcal{F}(\mathbf{x}) = \prod_i p_i(x_i)$. The coordinate system in the figure is that of the mixture coordinates, $\alpha_i = p^i$. For the factorizable distributions this means

$$\begin{aligned} p(00) &= p_0(0)p_1(0) \\ \alpha_1 = p(01) &= p_0(0)p_1(1) \\ \alpha_2 = p(10) &= p_0(1)p_1(0) \\ \alpha_3 = p(11) &= p_0(1)p_1(1) \end{aligned}$$

In natural coordinates,

$$\theta_i = \log \frac{p^i}{p^0},$$

which happen to be the sequence LLRs we defined in (4.1), the exponential family of factorizable distributions $\mathcal{E}_\mathcal{F}$ is a plane, see Fig. 4.4. Written explicitly, the natural

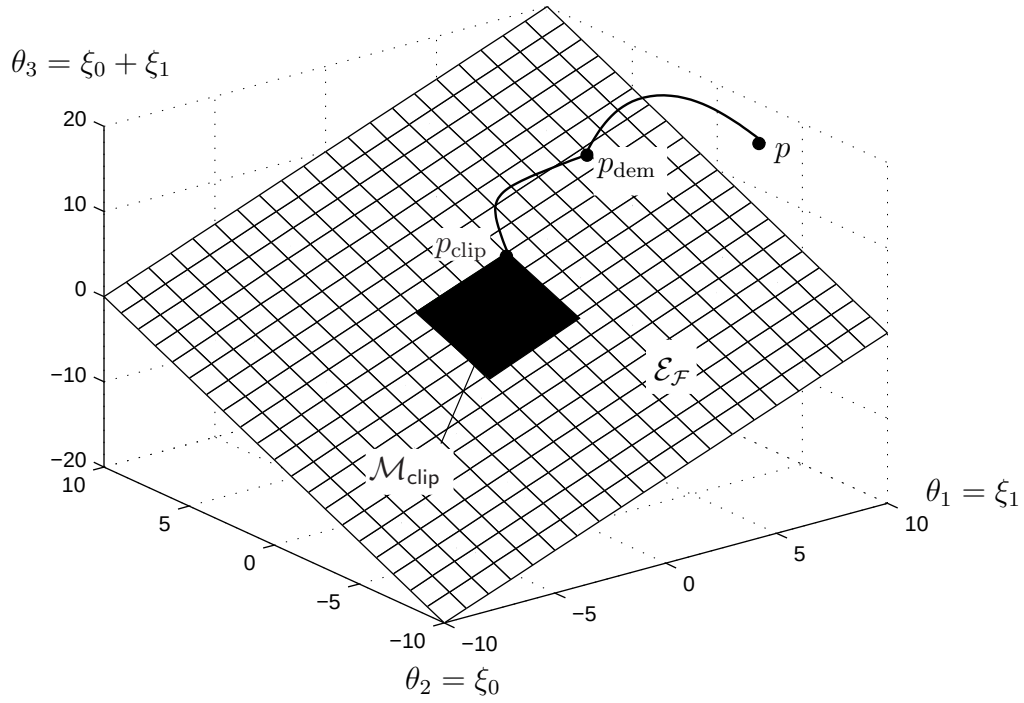

Figure 4.4: Bit LLR clipping: interpretation as projection of p_{dem} onto the clipping manifold $\mathcal{M}_{\text{clip}}$.

coordinates are

$$\begin{aligned}
\theta_1 &= \ln \frac{p^1}{p^0} = \ln \frac{p_1(1)}{p_1(0)} = \xi_1 \\
\theta_2 &= \ln \frac{p^1}{p^0} = \ln \frac{p_0(1)}{p_0(0)} = \xi_0 \\
\theta_3 &= \ln \frac{p^1}{p^0} = \ln \frac{p_0(1)p_1(1)}{p_0(0)p_1(0)} = \xi_0 + \xi_1,
\end{aligned}$$

where ξ_0 and ξ_1 are the LLR values of the first and the second bit respectively. We see that the coordinates of the distributions in $\mathcal{E}_{\mathcal{F}}$ are linear functions of the LLRs of the individual bits. It is therefore easy to calculate the coordinates of the submanifold $\mathcal{M}_{\text{clip}}$ which contains factorizable sequence pmfs that are allowed for a certain clipping value. For the figure, a uniform (for both bits) clipping threshold of $\gamma = 2$ has been chosen. The clipping region is the dark square that is located around the origin, the latter representing the uniform distribution.

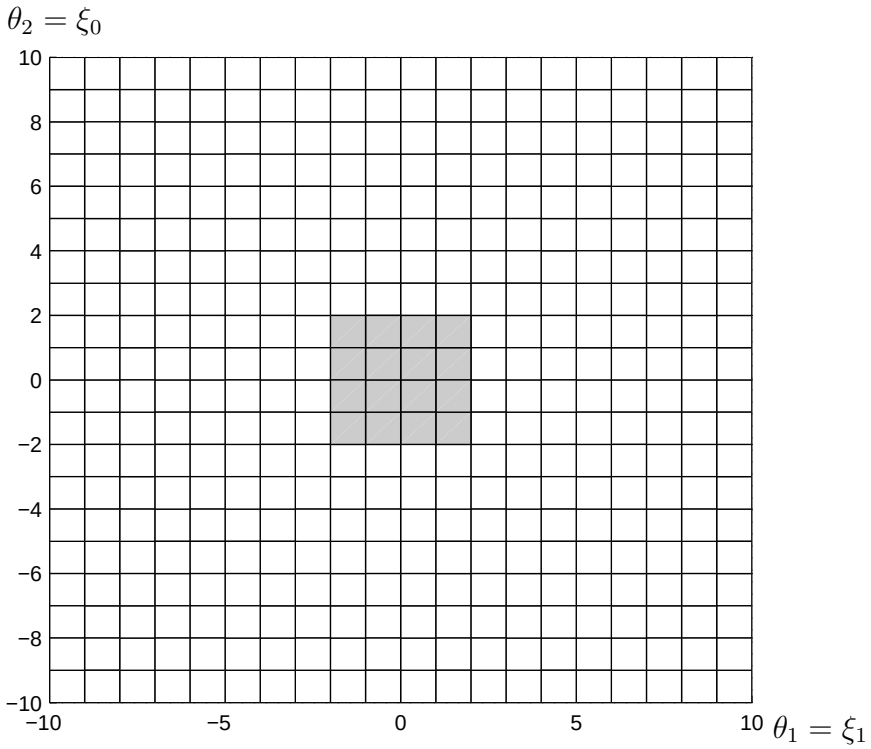

Figure 4.5: Bit LLR clipping (2-dimensional representation)

Because the manifold of factorizable distributions is two-dimensional, the coordinate θ_3 is redundant, therefore we can also illustrate it like in Fig. 4.5.

$\mathcal{M}_{\text{clip}}$ is a convex set. Bit LLR clipping can be interpreted as an I-projection of a distribution from the family of factorizable distributions onto the clipping-manifold $\mathcal{M}_{\text{clip}}$.

Bit LLR clipping as we have explained in the present section can be seen as a special case of sequence-LLR clipping, with which we dealt in the previous section. The expressions (4.4) stay unchanged, we repeat them here:

$$p_{\text{clip}} = \arg\min_{q} D(q\|p), \tag{4.6a}$$

$$D(q\|\bar{p}) \leq D^*, \tag{4.6b}$$

$$\sum_i q^i = 1. \tag{4.6c}$$

We want to find the distribution from the clipping manifold that has minimum

KL divergence to the original distribution p, and the clipping manifold is the set of distributions that has some maximum KL divergence to the uniform distribution. What is different now compared to the sequence LLR case is the assumption that the pmf p as well as the clipped distribution p_{clip} are factorizable distributions.

We again consider sequences $\mathbf{x}$ of length K, which means there are $N = 2^K$ different sequences, and the pmf to be clipped, $p(\mathbf{x}) = \prod_{k=0}^{K-1} p_k(x_k)$. Due to the property of the KL divergence of two factorizable distributions (2.4) that we discussed in example 2.2.6,

$$D(q\|p) = \sum_k D(q_k\|p_k) = \sum_k \left[q_k(0) \log \frac{q_k(0)}{p_k(0)} + q_k(1) \log \frac{q_k(1)}{p_k(1)} \right].$$

The Lagrangian function (4.5) is now

$$\begin{aligned} L &= \sum_k \left((1 - q_k(1)) \ln \frac{1 - q_k(1)}{1 - p_k(1)} + q_k(1) \ln \frac{q_k(1)}{p_k(1)} \right) \\ &+ \lambda_1 \left[\sum_k \Big((1 - q_k(1)) \ln\big(N(1 - p_k(1))\big) + q_k(1) \ln(N p_k(1)) \Big) - D^* \right] =: \sum_k L_k(p_k(1)). \end{aligned}$$

Hence L consists of K functions L_k that can be individually minimized by differentiating with respect to $p_k(1)$,

$$\frac{\partial L_k}{\partial p_k(1)} = 1 + \ln \frac{q_k(1)}{p_k(1)} + \lambda_1 \big(1 + \ln 2p_k(1)\big) \stackrel{!}{=} 0,$$

and the solution to this problem is again

$$p_k(1) = c(\alpha) \cdot p_k(1)^\alpha.$$

Each component minimization problem is therefore the same as in the case of bit LLR clipping, described by (4.3) and illustrated by Fig. 4.1.

In this way we have shown that the factorizable pmf

$$p_{\text{clip}}(\mathbf{x}) = \prod_k p_{\text{clip},k}(x_k),$$

which is the product of the individual clipped bit pmfs - which in turn result from the clipping of the respective bit LLRs - has an information-geometric interpretation: it can be described as an I-projection onto the clipping manifold $\mathcal{M}_{\text{clip}}$. We conjecture that the properties of the I-projection, particularly the Pythagorean theorem, may be useful for assessing the performance of communications systems in which LLR clipping is used.

5

Simulation Results

5.1 Implementation Issues

A BICM-receiver as shown in Fig. 5.1 has been implemented in MATLAB. The major difference between the formulas derived for the operation of the demapper and decoder sub-blocks in the previous chapter and the implementation is that LLRs have been used for the implementation instead of bit probabilities. As we have already shown in the previous chapter, the two descriptions are equivalent.

The operation of the demapping sub-block is described by the expression (3.21). A derivation using LLRs instead of bit-probabilities leads to

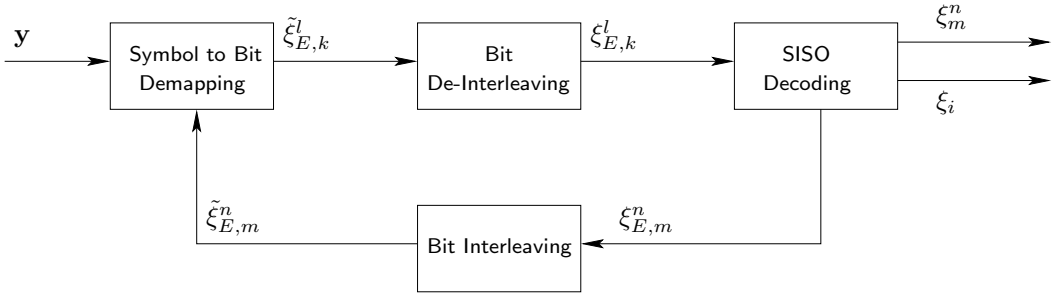

Figure 5.1: Iterative BICM receiver.

$$
\begin{aligned}
\tilde{\xi}_k^l &= \tilde{\xi}_{A,k}^l + \frac{\log \sum_{\tilde{\mathbf{x}}_k:\tilde{x}_k^l=1} \exp\{-\frac{1}{\sigma^2}\|\mathbf{y}_k - \mathbf{H}\mathbf{s}_k(\tilde{\mathbf{x}}_k)\|^2 + \frac{1}{2}\sum_{n\neq l}(2\tilde{x}_k^n - 1)\tilde{\xi}_k^n\}}{\log \sum_{\tilde{\mathbf{x}}_k:\tilde{x}_k^l=0} \exp\{-\frac{1}{\sigma^2}\|\mathbf{y}_k - \mathbf{H}\mathbf{s}_k(\tilde{\mathbf{x}}_k)\|^2 + \frac{1}{2}\sum_{n\neq l}(2\tilde{x}_k^n - 1)\tilde{\xi}_k^n\}} \\
&\approx \tilde{\xi}_{A,k}^l + \max_{\tilde{\mathbf{x}}_k:\tilde{x}_k^l=1}\left\{-\frac{1}{\sigma^2}\|\mathbf{y}_k - \mathbf{H}\mathbf{s}_k(\tilde{\mathbf{x}}_k)\|^2 + \frac{1}{2}\sum_{n\neq l}(2\tilde{x}_k^n - 1)\tilde{\xi}_k^n\right\} \\
&- \max_{\tilde{\mathbf{x}}_k:\tilde{x}_k^l=0}\left\{-\frac{1}{\sigma^2}\|\mathbf{y}_k - \mathbf{H}\mathbf{s}_k(\tilde{\mathbf{x}}_k)\|^2 + \frac{1}{2}\sum_{n\neq l}(2\tilde{x}_k^n - 1)\tilde{\xi}_k^n\right\} \\
&= \tilde{\xi}_{A,k}^l + \min_{\tilde{\mathbf{x}}_k:\tilde{x}_k^l=0}\left\{\frac{1}{\sigma^2}\|\mathbf{y}_k - \mathbf{H}\mathbf{s}_k(\tilde{\mathbf{x}}_k)\|^2 - \frac{1}{2}\sum_{n\neq l}(2\tilde{x}_k^n - 1)\tilde{\xi}_k^n\right\} \\
&- \min_{\tilde{\mathbf{x}}_k:\tilde{x}_k^l=1}\left\{\frac{1}{\sigma^2}\|\mathbf{y}_k - \mathbf{H}\mathbf{s}_k(\tilde{\mathbf{x}}_k)\|^2 - \frac{1}{2}\sum_{n\neq l}(2\tilde{x}_k^n - 1)\tilde{\xi}_k^n\right\}.
\end{aligned}
$$

Here we used the max-log-approximation $\log(e^a + e^b) \approx \max(a, b)$. It is notable that the LLR is the sum of the a-priori information $\tilde{\xi}_{A,k}^l$ and the extrinsic information.

The expression is for a $N \times M$ MIMO-system with a channel matrix $\mathbf{H}$, and MIMO-symbols $\mathbf{s}$. The SISO-system is included as the special case $N = M = 1$.

For the decoding sub-block, an existing C-implementation of the BCJR-algorithm has been used.

5.2 Performance of Iterative Demodulation

The general simulation parameters are: a block length of 256 information bits, a rate-1/2 convolutional code (generator polynomials 13,15), a random interleaver, a 16QAM signal constellation with set partition mapping, and an i.i.d. fast Rayleigh-fading channel. These parameter have been used unless stated otherwise.

Fig. 5.2 shows the performance improvement that is achieved by using the iterative procedure. We see a huge gain already in the first iteration. The subsequent iterations improve the performance even more in the lower SNR domain.

For comparison, in Fig. 5.3 a 4QAM signal constellation has been used. It seems

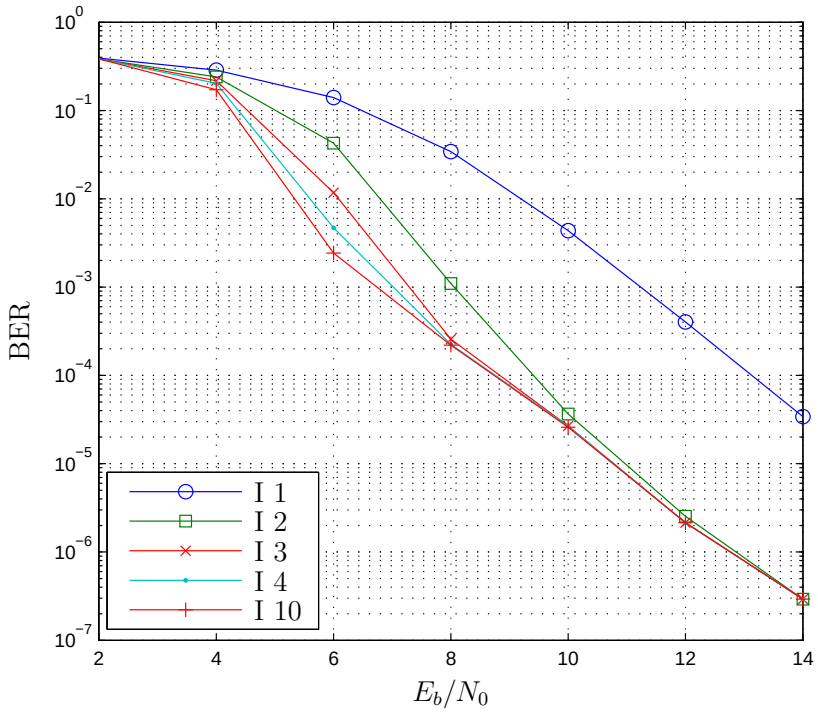

Figure 5.2: Performance improvement over iterations (16QAM).

that the 4QAM constellation does not benefit from the iterative decoding as much the 16QAM constellation. Fig.5.4 shows the results for a 2x2 MIMO system. The deviation of the expected behaviour in the high SNR regime is most certainly due to numerical problems. Qualitatively there is no striking difference between the three results.

The next figures (5.5-5.7) show histograms of the LLR values that correspond to transmitted "1"-bits after the demapper, for an SNR of 12 dB and for the iterations 1, 2 and 10. We can see that the number of negative LLRs, which correspond to a wrong decision for bit "0", decreases over the iterations. Furthermore, the mean travels to the right and the variance increases from iteration to iteration. The variance normalized by the squared mean decreases from 2.6 in the first to 0.2 in the 10th iteration, however.

5.3 Influence of LLR Clipping

The following figures show the influence of LLR clipping onto the receiver performance, for a non-iterative receiver. Fig. 5.8 shows the bit error rate over the clipping value,

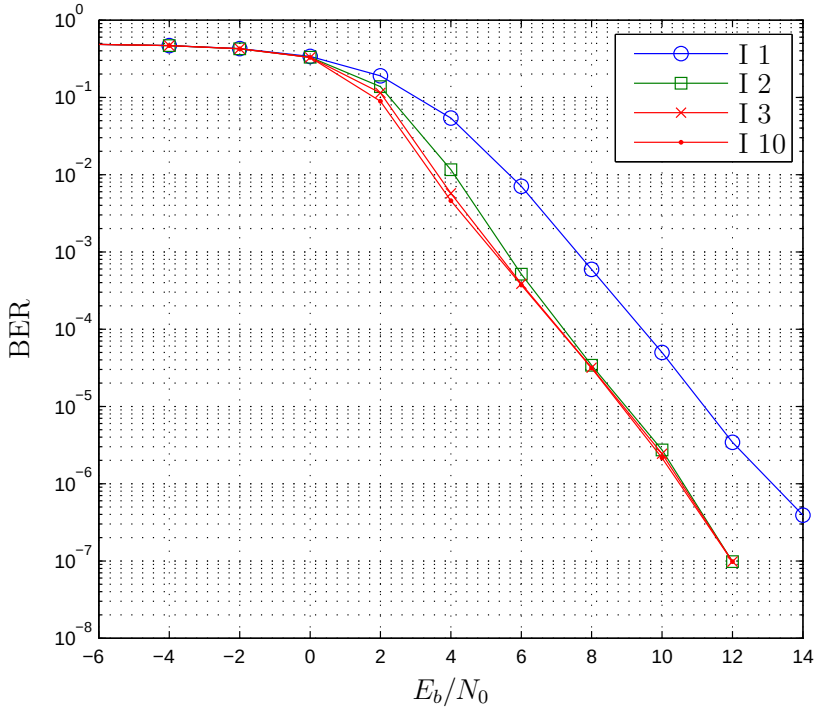

Figure 5.3: Performance improvement over iterations (4QAM).

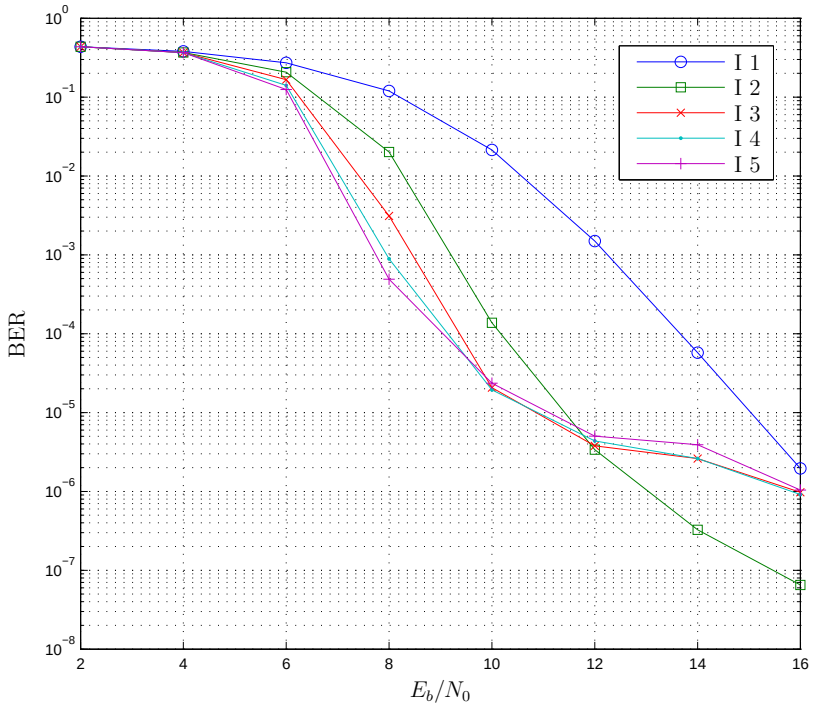

Figure 5.4: Performance improvement over iterations (2x2 MIMO).

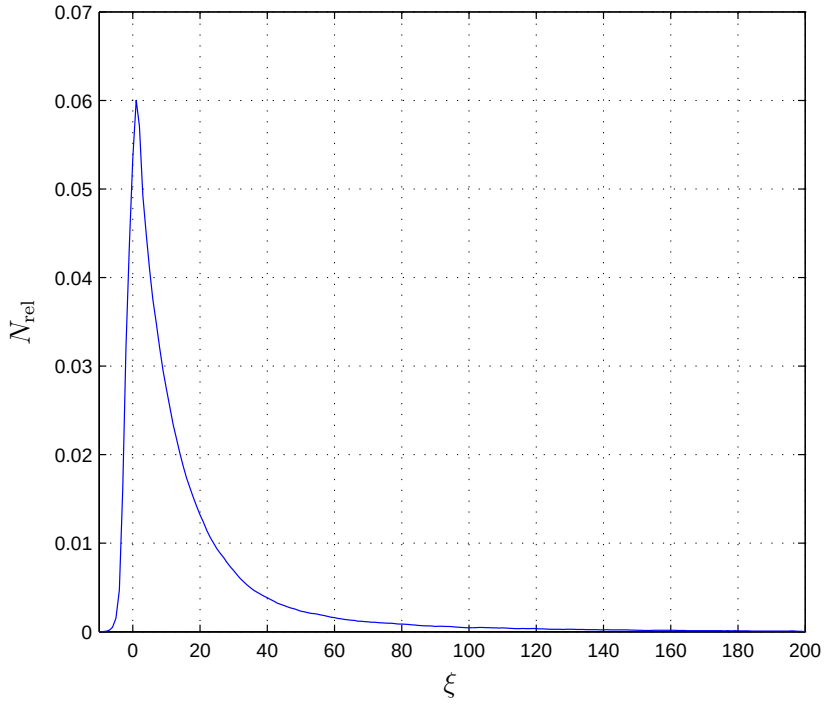

Figure 5.5: LLR histogram, $E_b/N_0 = 12$ dB, Iteration 1.

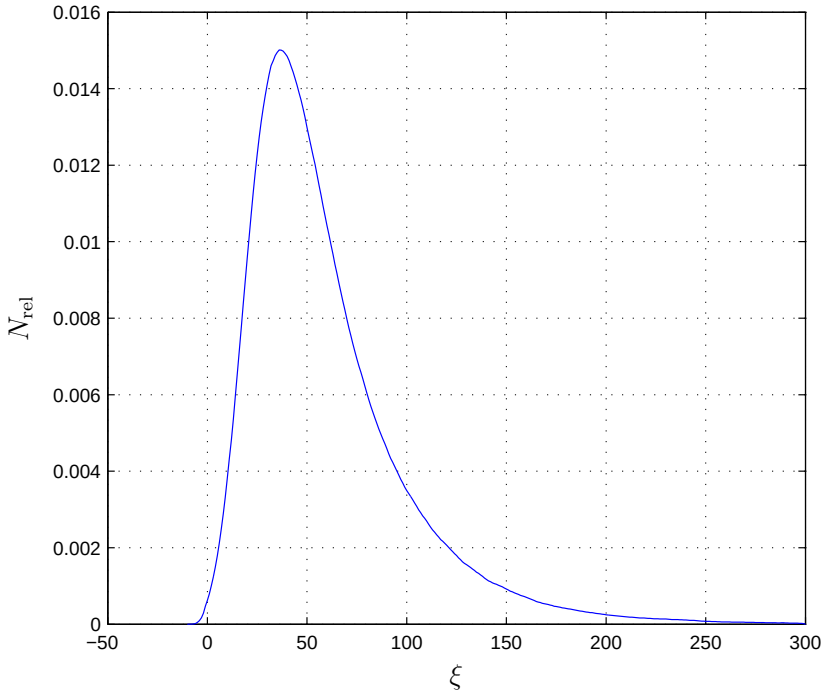

Figure 5.6: LLR histogram, $E_b/N_0 = 12$ dB, Iteration 2.

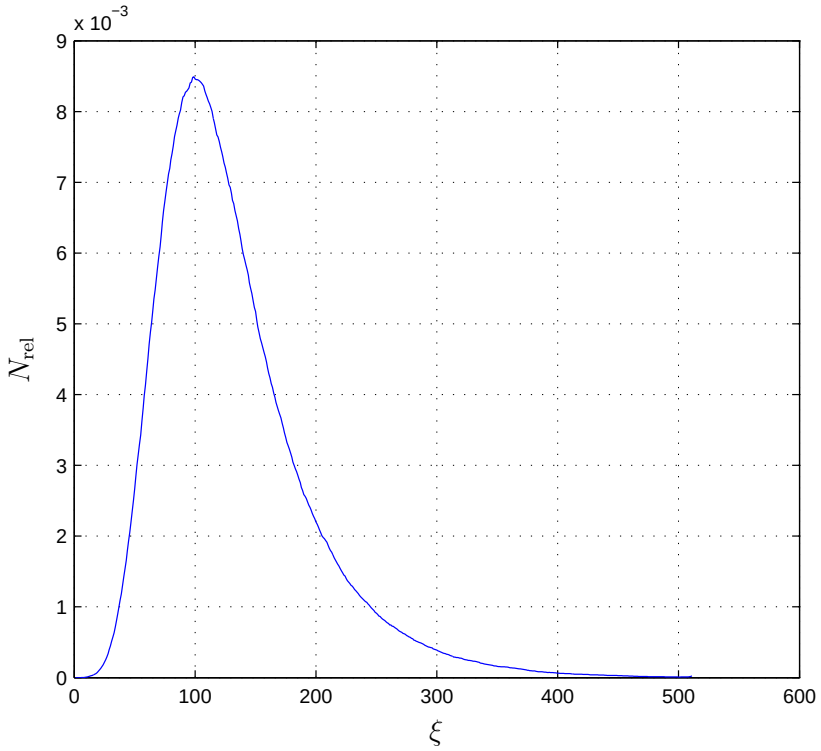

Figure 5.7: LLR histogram, $E_b/N_0 = 12$ dB, Iteration 10.

for different values of E_b/N_0. Coming from the right edge of the figure, where we have virtually no clipping at all, the clipping values become smaller, going to the left. The trace for $E_b/N_0 = 14dB$ is the first one that is affected by the clipping, the others follow. We conclude that for each SNR-value there is one clipping threshold, above which there is no performance penalty. The higher SNR levels are affected more than the lower ones, which means more aggressive clipping is possible at low SNR values without losing performance. Interestingly, by normalizing the clipping values to the square root of the SNR, the saturation regions of the traces are vertically aligned, see Fig. 5.9, with an optimum clipping threshold $\gamma_{\text{opt}} \approx 3 \cdot \sqrt{E_b/N_0}$. This means that by scaling the clipping threshold with the SNR, the performance degradation due to the clipping stays constant over the SNR. The optimum clipping threshold depends on the code, but only in a weak manner, see Fig. 5.10 and Fig. 5.11.

The next results show the relation between the clipping value and the number of LLRs relative to the total block length that get clipped. First, we see the relative

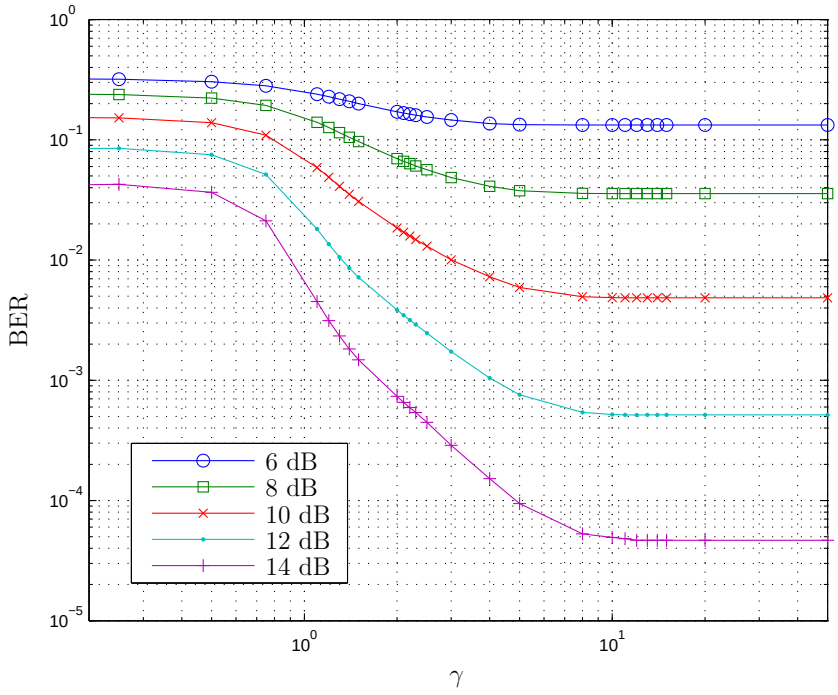

Figure 5.8: BER over clipping value.

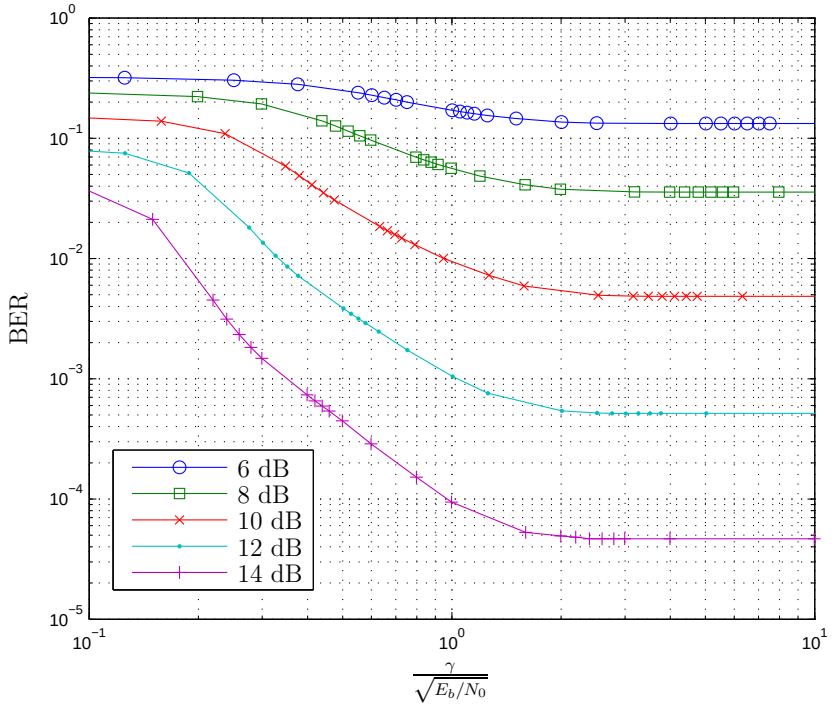

Figure 5.9: BER over normalized clipping value.

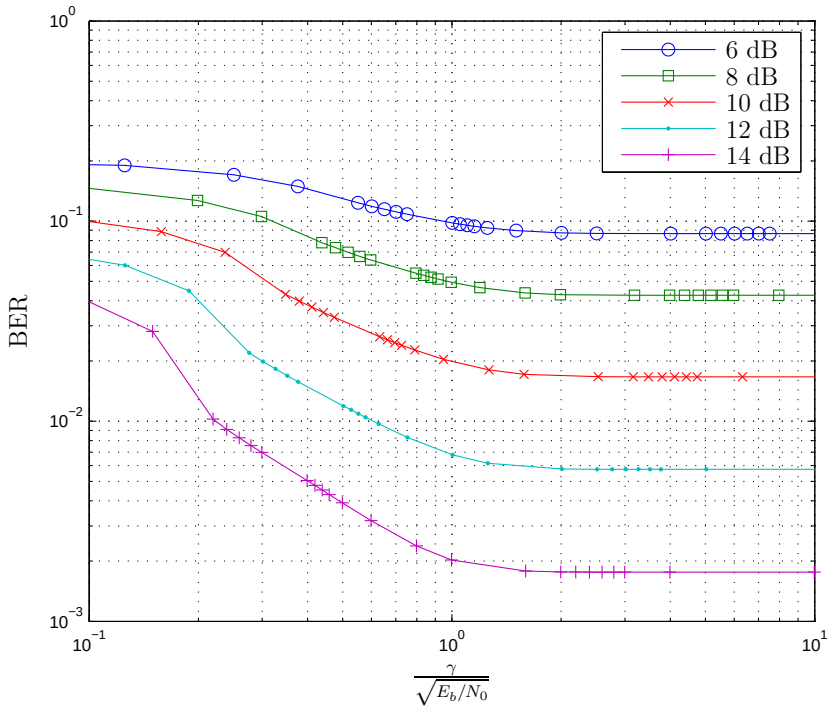

Figure 5.10: BER over normalized clipping value (Code 2,3).

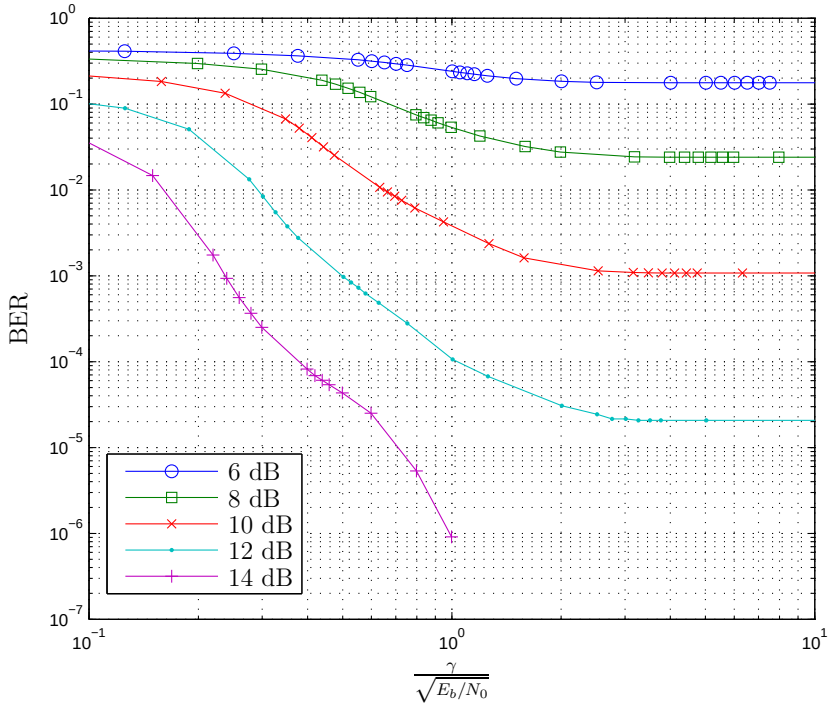

Figure 5.11: BER over normalized clipping value (Code 133,171).

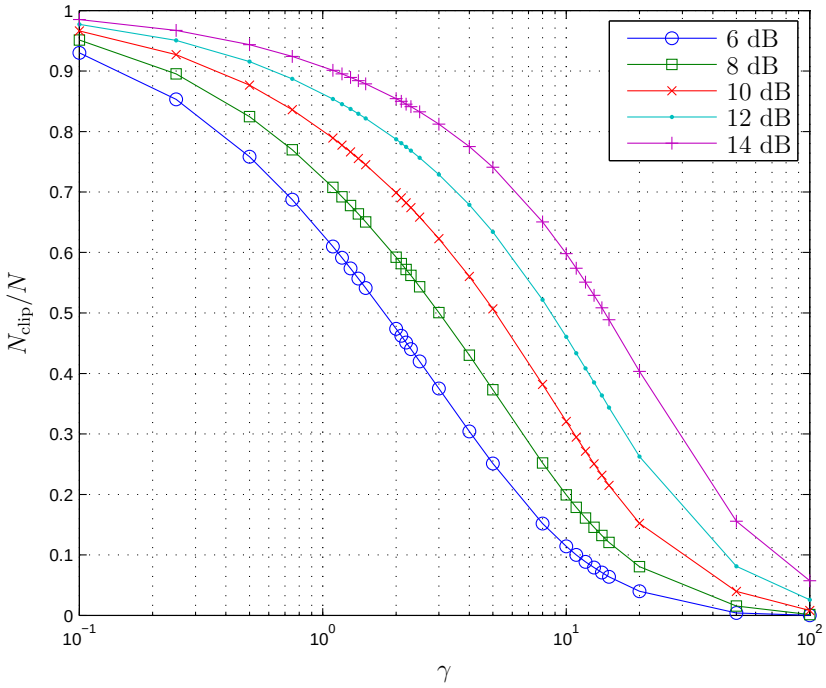

Figure 5.12: Relative number of clipped LLRs over clipping value.

number of clipped LLRs over the clipping level, in Fig. 5.12. The next figure, Fig. 5.13 shows that by normalizing the clipping value to E_b/N_0, we achieve that the traces for the different SNR-values nearly align. We conclude that the number of clipped LLRs stays constant over the SNR if we scale the clipping value accordingly with the SNR.

Fig. 5.14 shows the BER over the relative count of clipped LLRs. As expected the BER increases with the number of LLRs that get clipped. It is interesting to see that for high SNR levels, it is apparently possible to clip more LLRs without losing performance, compared to low SNR levels. This could be due to the fact that for high SNR values, most LLR values are quite large and the sign is in many cases correct, even before the channel decoding. In the low SNR regime, there are more wrong LLR signs, hence the absolute value which contains the reliability of a bit is more important. Due to clipping, the absolute values become more and more uniform, and the information that they contain is lost.

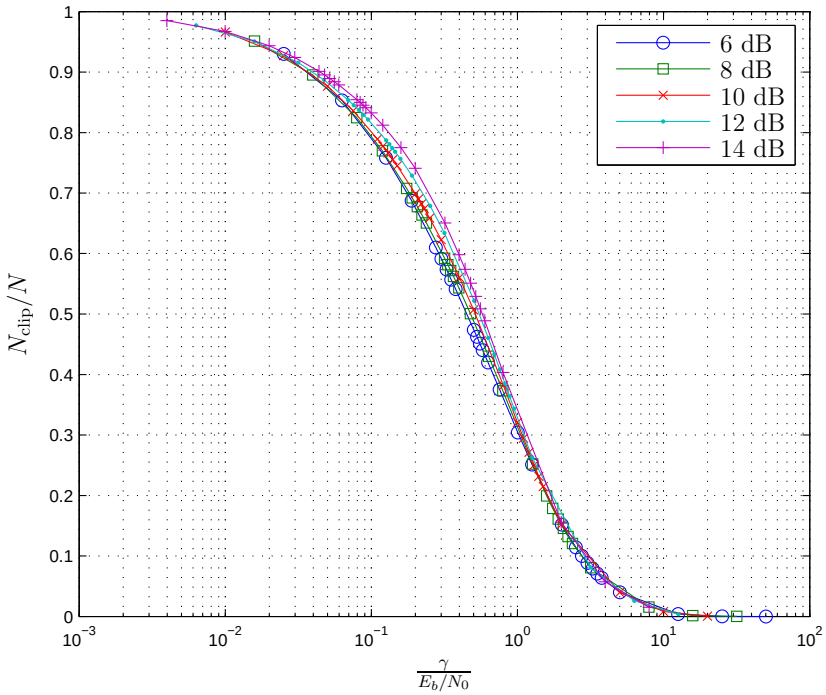

Figure 5.13: Relative number of clipped LLRs over normalized clipping value.

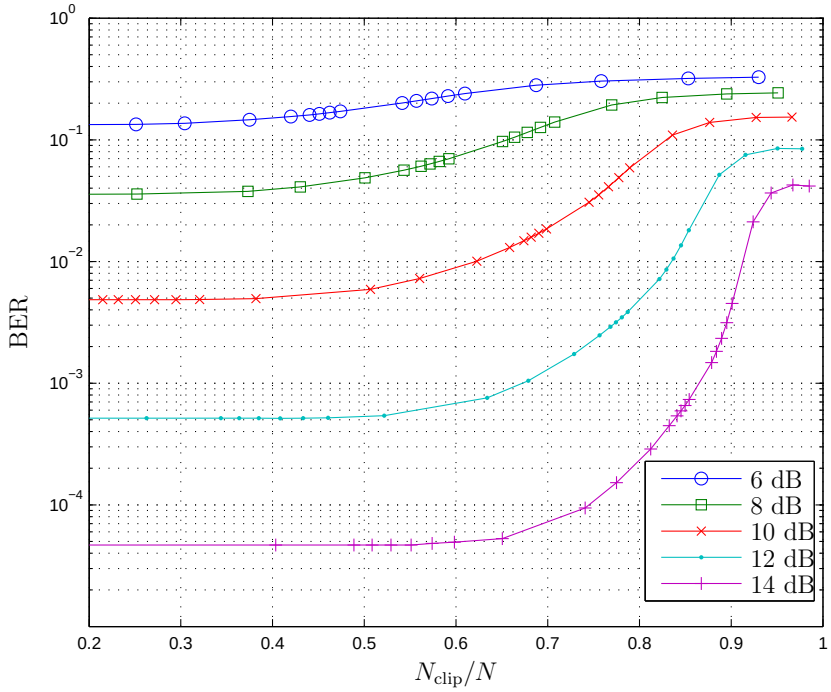

Figure 5.14: BER over the relative count of clipped LLRs.

6

Summary and Outlook

In this thesis, we have presented concepts of information geometry that are applicable to the description of iterative receivers. The main part is dedicated to the geometric interpretation of an iterative BICM receiver that consists of a symbol-to-bit demapper and a BJCR decoder. The operation of these sub-blocks has been recognized as information geometric projections of an input distribution onto certain manifolds of probability distributions. The use of bit-wise processing has lead to sequence probability distributions that are factorizable into marginal bit probabilities.

Then we have applied the information geometric concepts to the description of LLR clipping. Sequence LLRs have been introduced and analysis has shown that clipping these LLRs does not change the sequence MAP estimate. We have further shown that bit-wise LLR clipping can be interpreted as a projection of the original sequence probability distribution onto a clipping manifold.

Finally we have provided simulation results for the BICM system that had been treated in the previous chapters. We have shown the performance gain that can be achieved by soft and iterative decoding and the effects of LLR clipping on the performance. It has been recognized that very strong clipping is possible without noticeable

performance loss and that an optimum normalized clipping threshold that is valid for a range of SNR values can be found.

There is a number of things left that seem natural to explore further: Using the Pythagorean theorem for I-projections it should be possible to assess the performance loss due to clipping, by combining the projections onto the code manifold and on the factorizable distributions.

The analysis and simulations of LLR clipping could be extended to an iterative system.

The empirical distributions of LLR values could be approximated by analytically tractable distributions to assess the number of LLR values that get clipped for a certain clipping threshold.

Bibliography

[1] J. Hagenauer. The turbo principle: Tutorial introduction and state of the art. In *Proc. 1st Int. Symp. on Turbo Codes and Related Topics*, pages 1–11, Brest, France, Sept. 1997.

[2] E. Zehavi. 8-PSK trellis codes for a Rayleigh channel. *IEEE Trans. Comm.*, 40:873–884, May 1992.

[3] G. Caire, G. Taricco, and E. Biglieri. Bit-interleaved coded modulation. *IEEE Trans. Inf. Theory*, 44:927–945, May 1998.

[4] X. Li and J.A. Ritcey. Bit-interleaved coded modulation with iterative decoding. *Communications, 1999. ICC '99. 1999 IEEE International Conference on*, 2:858–863 vol.2, 1999.

[5] C. Studer, M. Wenk, A. Burg, and H. Bölcskei. Soft-output sphere decoding: Performance and implementation aspects. In *Proc. 40th Asilomar Conf. Signals, Systems, and Computers*, pages 2071–2076, Nov. 2006.

[6] I. Csiszár. Information theory and statistics. ENEE 728F Lecture Notes, Univ. Maryland, 1990. Available online at http://www.math.hu.dk/ma/kurser/informationsteori/csiszar.pdf.

[7] S. Amari. Information geometry of the EM and em algorithms for neural networks. *Neural Networks*, 8(9):1379–1408, 1996.

[8] B. Muquet. *Novel receiver and decoding schemes for wireless OFDM systems with cyclic prefix or zero padding*. PhD thesis, ENST, Paris (France), June 2001.

[9] I. Csiszár and P. Shields. Information theory and statistics: A tutorial. *Foundations and Trends in Communications and Information Theory*, 1(4):420–527, 2004.

[10] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge Univ. Press, Cambridge (UK), Dec. 2004.

[11] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, New York, 1991.

[12] M. Moher and T. A. Gulliver. Cross-entropy and iterative detection. *IEEE Trans. Inf. Theory*, 44:3097–3104, Nov. 1998.

[13] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Trans. Inf. Theory*, 20:284–287, March 1974.