**TECHNISCHE UNIVERSITÄT MÜNCHEN**

**Lehrstuhl für Nachrichtentechnik**

# Finite-Precision and Multi-Stream Distribution Matching

Marcin J. Pikus

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor–Ingenieurs

genehmigten Dissertation.

| | | |
|---|---|---|
| Vorsitzender: | | Prof. Dr.-Ing. Bernhard U. Seeber |
| Prüfer der Dissertation: | 1. | Prof. Dr. sc. tech. Gerhard Kramer |
| | 2. | Prof. Giuseppe Caire, Ph.D. |

Die Dissertation wurde am 28.05.2019 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 06.09.2019 angenommen.

# Preface

I'm grateful to my wife, family, friends, colleagues, and supervisors for their on-going support. Especially, I would like to thank Gerhard Kramer for his time and invaluable guidance. Secondly, I have to thank my colleagues from Huawei Technologies for creating and maintaining the atmosphere where one could develop and verify ideas. Every coffee break we took inspired me further. Last but not least, I would like to show my appreciation for colleagues at the Technical University of Munich, where I had this great opportunity to participate in developing a machine learning course, which also inspired my work.

Munich, May 2019 Marcin Pikus

# Contents

# Zusammenfassung

Um die Kapazität eines Kommunikationskanals zu erreichen, müssen in der Regel ungleich verteilte Symbole, sogenannte geshapte Symbole, übertragen werden. Die jüngste Forschungstätigkeit zu Shaping Methoden ist auf das Interesse aus der Industrie an der Verbesserung der spektralen Effizienz von optischen und drahtlosen Kanälen zurückzuführen. Der Begriff Shaping Methoden bezeichnet Methoden, um Kommunikationssysteme zu realisieren, die ungleich verteilte Symbole übertragen. Distribution-Matching ist eine Shaping Methode, die reversibel eine Sequenz unabhängiger und gleichverteilter Bits in eine Sequenz von Symbolen kodiert, die eine bestimmte Wahrscheinlichkeitsverteilung aufweisen. Diese Arbeit analysiert Distribution-Matching Methoden. Ein Algorithmus und Entwurfsregeln für ein auf arithmetischer Kodierung basierendes Distribution-Matching werden abgeleitet. Insbesondere wird die Abhängigkeit zwischen der Leistung eines solchen Distribution-Matchings und der Genauigkeit der Rechenoperationen bei der arithmetischen Kodierung quantifiziert. Außerdem wird gezeigt, wie man das Wahrscheinlichkeitsmodell für das auf der arithmetischen Kodierung basierende Distribution-Matching wählt, um die Leistung für kurze Blocklängen zu verbessern. Außerdem wird eine parallele Architektur für das Distribution-Matching entwickelt, die die Ausgabeverteilung durch eine Verteilung eines Produkts von Zufallsvariablen approximiert, und damit Flexibilität und Durchsatz erhöht. Die parallele Architektur führt viele Distribution-Matchings mit kleineren Ausgabealphabeten durch, um die resultierenden Sequenzen zusammen auf eine Sequenz aus dem primären Ausgabealphabet abzubilden. Die Architektur wird für verschiedene Distribution-Matching Algorithmen und Szenarien analysiert und optimiert. Schließlich werden die Vorteile von Shaping Methoden für mehrere Kommunikationskanäle, einschließlich nicht kohärenter Fading-Kanäle, untersucht.

# Abstract

Achieving the capacity of a communication channel usually involves transmitting non-uniformly distributed, or shaped, symbols. Recent research activity on shaping schemes is driven by industrial interest in improving the spectral efficiency of optical and wireless channels. Distribution matching is a shaping scheme that reversibly encodes a sequence of independent and uniformly distributed bits into a sequence of symbols that have a certain probability distribution. This work analyzes distribution matching techniques. We derive an algorithm and design rules for arithmetic coding distribution matching. In particular, we quantify the dependence between the performance of such distribution matching and the precision used for arithmetic operations by the arithmetic coding algorithm. Moreover, we show how to choose the probability model governing an arithmetic coding based distribution matcher to improve the performance for short blocklengths. We also develop a parallel architecture for distribution matching that approximates the output distribution by a product distribution, and thereby increases flexibility and throughput. The parallel architecture performs many distribution matchings with smaller output alphabets to jointly map the resulting sequences to a sequence from the primary output alphabet. We optimize and analyze the architecture for different distribution matching algorithms and scenarios. Finally, we examine the benefits of shaping schemes for several communication channels including non-coherent fading channels.

# 1

# Preliminaries

## 1.1. Probability and Information Theory

### 1.1.1. Probability Theory

#### Vectors and Sequences

A finite sequence (or a row vector) is denoted by by a bold symbol:

$$\boldsymbol{x} = [x_1, \ldots, x_n] .$$

The $i$-th entry of $\boldsymbol{x}$ is denoted by $x_i$ or $[\boldsymbol{x}]_i$. A subsequence of $\boldsymbol{x}$ starting at the $i$-th entry and ending at the $j$-th entry of $\boldsymbol{x}$ is denoted by

$$\boldsymbol{x}_i^j = [\boldsymbol{x}]_i^j = [x_i, \ldots, x_j] .$$

A concatenation of two sequences $\boldsymbol{a}$ and $\boldsymbol{b}$ is denoted by $[\boldsymbol{a}, \boldsymbol{b}]$. For example, we write

$$\left[[\boldsymbol{x}]_1^2, [\boldsymbol{x}]_3^4\right] = [x_1, x_2, x_3, x_4].$$

The number of entries of a finite sequence (or a dimension of a row vector) is denoted by $l(\boldsymbol{x})$, i.e., we have $l(\boldsymbol{x}) = n$ and $l(\boldsymbol{x}_i^j) = j - i + 1$.

### Random Variables

Random variables (RVs) are denoted by uppercase letters, e.g., $X$, and realizations of RVs by lowercase letters, e.g., $x$. A discrete RV $X$ takes values in a finite or countably infinite set $\mathcal{X}$. We describe $X$ with a probability mass function (PMF) $P_X$ on $\mathcal{X}$, which gives the probability of $X$ taking on the value $x$, i.e., $P_X(x) = \Pr[X = x], \forall x \in \mathcal{X}$. We denote a PMF with an uppercase $P$ with the corresponding RV as a subscript.

A continuous RV $X$ takes on an uncountable number of values from a set $\mathcal{X}$. Furthermore, if $X$ is absolutely continuous, there exists a non-negative function $p_X(x)$ such that $\forall_{\mathcal{S}\subseteq\mathcal{X}} \Pr[X \in \mathcal{S}] = \int_{\mathcal{S}} p_X(x)dx$. The function $p_X(x)$ is called a probability density function (PDF) of the RV $X$ and is denoted by lowercase $p$.

We write

$$X \sim P_X \text{ (resp. } X \sim p_X) \tag{1.1}$$

if the RV $X$ has the PMF $P_X$ (resp. PDF $p_X$).

The set of arguments for which a PMF (resp. PDF) of a discrete (resp. continuous) RV takes on non-zero values is called the support of the RV and is denoted by:

$$\text{supp}(P_X) = \{x \in \mathcal{X}, P_X(x) > 0 \text{ (resp. } \text{supp}(p_X) = \{x \in \mathcal{X}, p_X(x) > 0\}). \tag{1.2}$$

### Expectation

Suppose $X$ is a RV defined on a set $\mathcal{X}$ and $f$ is some real-valued function. Then $f(X)$ is also a RV and the expectation of $f(X)$ is denoted by $\mathbb{E}_X[f(X)]$. If $X$ is a discrete RV, then we have

$$\mathbb{E}_X\left[f(X)\right] = \sum_{a\in\text{supp}(P_X)} P_X(a)f(a). \tag{1.3}$$

If $X$ is a continuous RV with a density, then we have

$$\mathbb{E}_X\left[f(X)\right] = \int_{\text{supp}(p_X)} p_X(a)f(a)\mathrm{d}a. \tag{1.4}$$

assuming the integral exists.

### 1.1.2. Information Theory

#### Entropy and Differential Entropy

The entropy of a discrete RV $X$ is is defined as

$$\mathbb{H}(P_X) = \mathbb{H}(X) = \mathbb{E}_X[-\log_2(P_X(X))]. \tag{1.5}$$

Some important properties of the entropy are [1, Appendix A]:

1. Non-negativity: $\mathbb{H}(X) \geq 0$.
2. Maximum Entropy:
$$\mathbb{H}(X) \leq \log|\mathcal{X}| \tag{1.6}$$
with equality if and only if $X$ is uniformly distributed on $\mathcal{X}$.
3. Maximum Entropy with Linear Constraint: Consider a cost function $f\colon \mathcal{X} \to \mathbb{R}$ and the linear constraint $\mathbb{E}[f(X)] = P$. The maximum entropy is attained by the Maxwell-Boltzmann distribution:
$$P_X(x) \propto e^{-\alpha f(x)} \text{ for } x \in \mathcal{X}.$$
The proof follows, e.g., by using Lagrange multipliers.

For a continuous RV, we use differential entropy defined as

$$h(X) = \mathbb{E}_X[-\log_2(p_X(X))]. \tag{1.7}$$

If $X$ is a discrete RV, then we say that $h(X) = -\infty$.

#### Kullback–Leibler Divergence

The Kullback–Leibler(KL) divergence, also referred to as informational divergence or relative entropy, between two PMFs $P_X$ and $Q_X$ is defined as

$$\mathbb{D}\left(P_X \| Q_X\right) = \mathbb{E}_X\left[\log \frac{P_X(X)}{Q_X(X)}\right] \tag{1.8}$$

for $X \sim P_X$. The divergence can be defined in an analogous way for continuous RVs with PDFs $p_X, q_X$ instead of the PMFs $P_X, Q_X$. Some important properties of the divergence are [1, Appendix A]:

1. Non-negativity: $\mathbb{D}(P_X \| Q_X) \geq 0$, with equality for $P_X = Q_X$.

2. Non-symmetry: $\mathbb{D}(P_X \| Q_X) \neq \mathbb{D}(Q_X \| P_X)$.

3. Data Processing Inequality: Consider two RV $X$, $X'$ with PMFs $P_X$, $Q_X$ defined on the set $\mathcal{X}$, respectively. The two variables $X$, $X'$ are processed by the channel $P_{Y|X}$ to obtain the the variables $Y$, $Y'$ with PMFs $P_Y$, $Q_Y$ defined on the set $\mathcal{Y}$, respectively. That is, we have

$$P_Y(y) = \mathbb{E}_X\left[P_{Y|X}(y|X)\right] \ \forall y \in \mathcal{Y}$$
$$Q_Y(y) = \mathbb{E}_X\left[P_{Y|X}(y|X)\right] \ \forall y \in \mathcal{Y}.$$

   Then we have

$$\mathbb{D}(P_Y \| Q_Y) \leq \mathbb{D}(P_X \| Q_X). \tag{1.9}$$

   The proof follows, e.g., by manipulations and convexity properties (Jensen inequality). One can understand the result as follows: processing the observation makes it more difficult to determine whether it comes from $P_X$ or $Q_X$. The data processing inequality also holds for continuous/mixed RVs and channels (e.g., $X$, $X'$ may be discrete and $Y$, $Y'$ may be continuous) although the formulation above uses PMFs.

### Mutual Information

The mutual information between two RV $X, Y$ is defined as

$$\mathbb{I}(X;Y) = \mathbb{D}(P_{XY} \| P_X P_Y).$$

The mutual information can be defined in an analogous way for continuous RVs. Some important properties of the mutual information are [1, Appendix A]:

1. Non-negativity: $\mathbb{I}(X;Y) \geq 0$.

2. Independence indicator: $\mathbb{I}(X;Y) = 0 \iff X$ and $Y$ are independent.

3. Symmetry: $\mathbb{I}(X;Y) = \mathbb{I}(Y;X)$.

4. Data Processing Inequality: Consider three RVs $X, Y, Z$ forming the Markov chain $X \Leftrightarrow Y \Leftrightarrow Z$. We have

$$\mathbb{I}(X;Z) \leq \mathbb{I}(X;Y). \tag{1.10}$$

The data processing inequality for mutual information is a consequence of the data processing inequality (1.9) for informational divergence. One can understand the result as follows: processing of RVs can not increase the information that one of them carries about the other one. The data processing inequality holds for continuous/mixed RVs as well.

## 1.2. Communication Systems

Shannon [2] identified the purpose of communication as follows

*The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point.*

The communication can take place in space, e.g., transmitting data to a remote destination, or in time, e.g., storing data for future retrieval. We focus now on communication systems that transmit digital data to remote destinations. A simple model of such a system is presented in Figure 1.1. Digital data (data quantized in time and values) enter the transmitter. The transmitter processes the data to obtain a sequence of (digital) signal points. The transmitter processing may involve changing the representation of the data or channel coding. The actual communication often takes place over a physical channel (for example by means of electro-magnetic waves, light, or acoustic waves) which is continuous in nature (continuous in time and values). Thus, in the next step the signal points are mapped to physical quantities that can propagate over the physical channel. The channel modifies the signal, e.g., by adding noise. At the receiver, the physical quantities are translated into estimates of the transmitted signal points, and the estimated signal points are processed to obtain an estimate of the data.

### 1.2.1. Additive White Gaussian Noise Channel

Consider a wireless additive white Gaussian noise (AWGN) channel, where the continuous transmitted signal is affected by three main phenomena: propagation delay, attenuation, and noise. Propagation delay shifts the time axes at the receiver and transmitter. That is, let $x(t) \in \mathbb{R}, t \in \mathbb{R}$, be the continuous transmitted signal (the value of

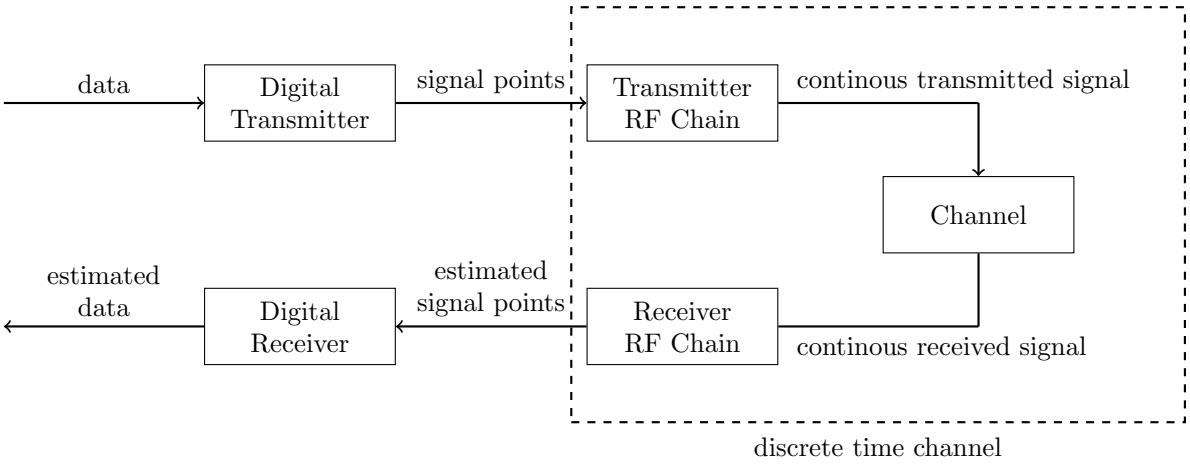



Figure 1.1.: Digital communication system.

the electric field at the output of the transmitter radio frequency (RF) chain). Then the continuous received signal (we ignore the other phenomena affecting the signal) is $x(t-\tau)$ where $\tau$ is the propagation delay between the transmitter and the receiver. One can thus shift the clock at the receiver RF chain by $\tau$ to have the continuous received signal equal to $x(t)$. In the AWGN channel model the attenuation is constant and mainly caused by the propagation loss. On can normalize (amplify) the continuous received signal to have the same amplitude as the transmitted signal. This results in the received signal $y(t)$ being

$$y(t) = x(t) + n(t)$$

where $n(t)$ is noise, e.g., thermal noise due to the receiver RF chain. The noise is modeled as a white Gaussian random process. That is, for any $t_1 \neq t_2$, $n(t_1)$ and $n(t_2)$ are independent zero-mean Gaussian RVs. Suppose that a symbol is transmitted every $T$ seconds, and that a base waveform $v(t)$ of the transmitter RF chain is chosen such that it is orthonormal to its time-shifts by multiples of $T$. The transmitted signal is

$$x(t) = \sum_{i \geq 0} X_i v(t - iT)$$

where the real-valued $X_i$ are digital signal points produced by the digital transmitter. In the absence of noise, each digital signal point can be recovered by filtering

$$X_i = \int_{-\infty}^{\infty} x(t) v(t - iT) \, \mathrm{d}t$$

which follows because $v(t)$ is orthonormal to the other shifts. With the noise, the filtering operation yields

$$\begin{aligned} Y_i &= \int_{-\infty}^{\infty} y(t)v(t-iT)\,\mathrm{d}t \\ &= \int_{-\infty}^{\infty} x(t)v(t-iT)\,\mathrm{d}t + \int_{-\infty}^{\infty} n(t)v(t-iT)\,\mathrm{d}t \\ &= X_i + N_i \end{aligned} \tag{1.11}$$

where $N_i, i \geq 0$ is a filtered white Gaussian noise process, i.e., the sequence $N_i, i \geq 0$, is a sequence of IID zero-mean Gaussian RVs [3, Chapter 8]. Equation (1.11) shows that by filtering one can transform the continuous-time channel into a discrete-time channel, see Figure 1.1. In practice, communication systems often constrain the transmitter output $X$ to be from the amplitude shift keying (ASK) alphabet

$$\mathcal{X} = \{-2^m + 1, \ldots, -5, -3, -1, 1, 3, 5, \ldots, 2^m - 1\}. \tag{1.12}$$

Finally, the discrete-time channel between the transmitter and receiver becomes

$$Y = \Delta X + N \tag{1.13}$$

where $N \sim \mathcal{N}(0,1)$, $X \in \mathcal{X}$, and we introduced the scaling $\Delta > 0$ to allow more flexibility with the fixed alphabet $\mathcal{X}$. Usually the transmitter has a limited power at its disposal. The transmitter power is proportional to the square of the magnitude of the generated electric field, which results in the average power constraint

$$\mathbb{E}\left[|\Delta X|^2\right] \leq P. \tag{1.14}$$

### Efficient Communication over the AWGN Channel

Consider the AWGN channel (1.13) used $n$ times:

$$[Y_1, \ldots, Y_n] = \Delta\,[X_1, \ldots, X_n] + [N_1, \ldots, N_n] \tag{1.15}$$

$$\boldsymbol{Y} = \boldsymbol{X} + \boldsymbol{N} \tag{1.16}$$

where $\Delta$ was merged into the vector $\boldsymbol{X}$, and $\boldsymbol{N}$ is multivariate Gaussian with zero-mean and identity-covariance, i.e., $\boldsymbol{N} \sim \mathcal{N}(\boldsymbol{0}, \mathbb{I})$. Standard digital transmitters generate signal points with a uniform distribution on the alphabet $\mathcal{X}$. This results in $\boldsymbol{X}$ being

| | Shaping gain | |
|---|---|---|
| $n$ | linear | dB |
| 2 | 1.047 | 0.20 |
| 4 | 1.111 | 0.46 |
| 8 | 1.183 | 0.73 |
| 16 | 1.252 | 0.98 |
| 32 | 1.309 | 1.17 |
| 64 | 1.350 | 1.31 |
| $\infty$ | $\frac{1}{6}\pi e$ | 1.53 |

Table 1.1.: Shaping gain when using the $n$-sphere bounded multidimensional constellation in comparison with the $n$-cube bounded constellation.

uniformly distributed on an $n$-dimensional cubic lattice enclosed within an $n$-cube. Let $\mathcal{X}_n$ be an $n$-dimensional constellation, e.g., for the standard digital transmitters we have $\mathcal{X}_n = \underbrace{\mathcal{X} \times \mathcal{X} \times \ldots \times \mathcal{X}}_{n \text{ times}} = \mathcal{X}^n$.

Consider a digital receiver that implements the maximum likelihood (ML) decoder, which for each received vector $\boldsymbol{y}$ outputs $\boldsymbol{x}_{\text{MAP}}$ such that

$$\boldsymbol{x}_{\text{MAP}} = \underset{\boldsymbol{x} \in \mathcal{X}^n}{\operatorname{argmax}}\, p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{x}) \tag{1.17}$$

$$= \underset{\boldsymbol{x} \in \mathcal{X}^n}{\operatorname{argmin}}\, \|\boldsymbol{y} - \boldsymbol{x}\|^2. \tag{1.18}$$

The ML decoder thus chooses the vector $\boldsymbol{x}$ closest to the vector $\boldsymbol{y}$ with respect to Euclidean distance. The probability of a decoding error, i.e., the probability that the vector $\boldsymbol{x}_{\text{MAP}}$ returned by the decoder is different that the transmitted vector, is upper-bounded and closely approximated by the probability of the magnitude of the noise vector $\boldsymbol{N}$ exceeding $\Delta$ (to see this observe that the noise vector $[\Delta + \epsilon, 0, \ldots, 0]$ with $\epsilon > 0$ suffices to cause a decoding error if, e.g., $X_1 = -1$ is transmitted). We would like the error probability to be below some chosen value $p_e$ for all transmit vectors. That is, no two vectors in the constellation $\mathcal{X}_n$ should be closer than some distance $d_e$. For $\boldsymbol{X}$ uniform, the most energy efficient constellation is enclosed (approximately) by an $n$-sphere [4]. By comparing the average energy of a point uniformly distributed in an $n$-cube and $n$-sphere of the same volume (this ensures the both constellations have the same transmission rate and $p_e$), one can compute an approximate power saving resulting

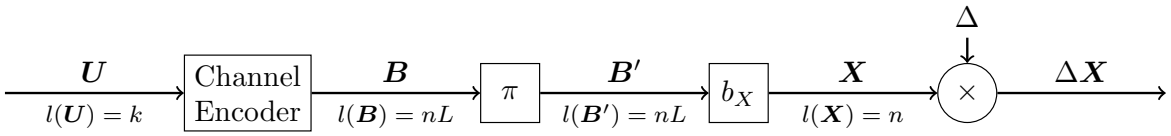



Figure 1.2.: BICM transmitter.

from using the optimal $n$-sphere bounded constellation, as compared to using an $n$-cube. The power savings computed in [5] are presented in Table 1.1 (the power saving is defined as the reduction in average transmission power per two dimensions). For $n \to \infty$ the probability distribution of the marginals $X_i,\ i = 1, \ldots, n$, from the vector $\boldsymbol{X}$ approach a zero-mean IID Gaussian distribution. The same result can be obtained from Shannon's coding theorem where the Gaussian PDF maximizes

$$\max_{p_x\colon \mathbb{E}[|X|^2] \leq P} \mathbb{I}\left(X; \Delta X + N\right). \tag{1.19}$$

## 1.2.2. Bit-interleaved Coded Modulation

Bit-interleaved coded modulation (BICM) [6] is a particular communication scheme that uses a binary error-correcting code with non-binary modulation. A BICM transmitter is presented in Figure 1.2. The binary data sequence $\boldsymbol{U}$ of dimension $k$ is encoded into a binary codeword $\boldsymbol{B}$ of dimension $nL \geq k$. Next, the codeword's bit are interleaved (permuted) to obtain a binary sequence $\boldsymbol{B}'$. The interleaver effectively makes the consecutive bits for each bit-level independent. Next, the interleaved sequence $\boldsymbol{B}'$ is broken into subsequences of length $L$, which are mapped by $b_X\colon \{0,1\}^L \to \mathcal{X}$ to a sequence $\boldsymbol{X}$ of symbols from the ASK alphabet $\mathcal{X}$. Finally, the symbols are scaled by $\Delta$ to match the power constraint. BICM achieves very good performance when combined with the mapping $b_X$ based on *Gray labeling* [6].

An optimal decoder uses symbol-metric decoding (SMD) [7] and achieves the rate $\mathbb{I}(X;Y)$. A BICM decoder uses bit-metric decoding (BMD). Let $\mathcal{X}_{l,b}$ be a subset of symbols from $\mathcal{X}$ whose label has value $b$ in the $l$-th position:

$$\mathcal{X}_{l,b} = \left\{x \in \mathcal{X}\colon\ [b_X(\boldsymbol{x})]_l = b\right\}.$$

The metric for BMD is

$$P_{Y|B_l}(y|b) \propto \sum_{x \in \mathcal{X}_{l,b}} p_{Y|X}(y|x) P_X(x) \tag{1.20}$$

where $l$ is the position of the bit in the symbol label, and $B_l$, $l = 1, \ldots, L$, is a RV corresponding to $l$-th bit-level ($l$-th bit in the label) of the symbol. In this case, the achievable rate is [7]

$$\sum_{l=1}^{L} \mathbb{I}(B_l; Y). \tag{1.21}$$

which is less than $\mathbb{I}(X;Y)$ in general. BMD simplifies the receiver (one binary channel code, independent processing of the bits) at the cost of decreased achievable rate. The rate difference between SMD and BMD is sometimes referred to as *the BICM loss*. We remark that the SMD performance could be achieved by a system with binary channel codes by using multi-level coding, see [8, 9].

### 1.2.3. Probabilistic Amplitude Shaping

Many communication systems, e.g., LTE, DVB-APSK, and WiMAX, use BICM and BMD. Observe in Figure 1.2 that a uniform data sequence $\boldsymbol{U}$ will result in a uniform distribution of the transmit symbols. Using uniformly distributed transmit symbols results in a shaping loss of up to 1.53dB, as explained in Section 1.2.1.

Two solutions are currently proposed for 5G mobile systems to overcome the shaping loss: *geometric shaping* (GS) (also referred to as *non-uniform constellations* [10]) and *probabilistically shaped coded modulation* (PSCM).

GS transmits uniformly distributed symbols from a non-uniformly spaced constellation, where the positions of the constellation points are altered to achieve a distribution beneficial for a certain channel. For example, the Gaussian distribution for the AWGN channel can be imitated by a constellation where the density of the points is higher close to the origin. GS has been studied extensively, e.g., in [11, 12], and is part of the ATSC 3.0 [13] and DVB-NGH [14] standards. The disadvantages of GS systems are an inferior performance as compared to PSCM systems [15], that the optimal constellation changes with Signal-to-Noise Ratio (SNR) which requires the receiver to implement multiple symbol demappers, and a reduced effective number of bits (ENOB) for uniform quantizers.

PSCM, on the other hand, transmits non-uniformly distributed symbols from a uniformly-

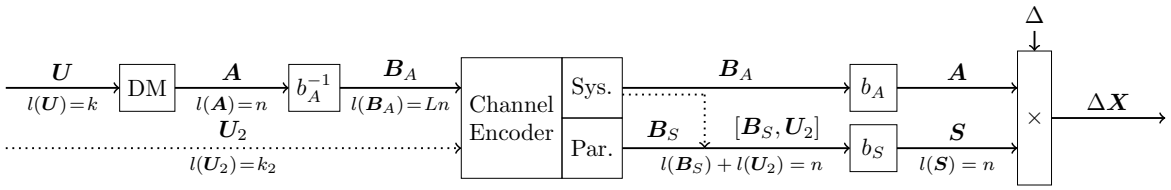



Figure 1.3.: PAS transmitter.

spaced ASK (or QAM) constellation. The essential part of PSCM is to generate non-uniformly distributed symbols from uniformly distributed input data bits. The problem has been studied and different practical implementations of PSCM have been proposed in the literature. A short review on the topic is available in [16, Sec. II].

*Probabilistic Amplitude Shaping* (PAS) [16] is a PSCM transmission scheme that has been used for optical systems [17] and has been proposed for the 5G mobile system [18]. PAS can be seen as a joint source-channel coding scheme where a source decoder, called a distribution matcher (DM), is introduced before a *systematic* channel encoder at the transmitter. Figure 1.3 presents a PAS transmitter. The receiver performs the inverse operations by introducing a source encoder, called an inverse DM, after a channel decoder. This brings a number of advantages. First, the channel input symbols can approach the capacity-achieving distribution because the DM can transform uniformly distributed bits of the input message into an arbitrary non-uniform distribution. Second, by appropriate parametrization of the DM, the transmitter can adjust the transmission rate without changing the parameters of the channel code [16]. Both of these aspects are different from conventional coded modulation schemes, such as BICM, where the rate matching is achieved by adjusting the modulation order or the parameters of the channel code.

Consider the PAS transmitter in Figure 1.3. The transmitter operates as follows (we ignore the dotted arrows for now):

1. A sequence $\boldsymbol{U}$ of $k$ data bits enters the DM.

2. The DM outputs a sequence $\boldsymbol{A}$ of $n$ amplitudes from the alphabet:

$$\mathcal{A} = \{1, 3, \ldots, 2^{L+1} - 1\}.$$

   The sequence $\boldsymbol{A}$ should mimic a sequence of IID symbols distributed according to some target distribution $P_A$ on $\mathcal{A}$.

3. Each amplitude is mapped by a fixed mapping $b_A^{-1}$ to a binary label of dimension $L$. This process produces a binary sequence $\boldsymbol{B}_A$ of dimension $Ln$.

4. The sequence $\boldsymbol{B}_A$ is encoded by a systematic channel encoder with rate $R = \frac{L}{L+1}$, i.e., for each amplitude symbol, one parity bit is produced. The systematic bits $\boldsymbol{B}_A$ leave the channel encoder via the output denoted by Sys, and the parity bits $\boldsymbol{B}_S$ via the output Par. The parity bits $\boldsymbol{B}_S$ are approximately IID Bernoulli-0.5 RVs [16].

5. The binary sequence $\boldsymbol{B}_A$ is mapped back to the sequence $\boldsymbol{A}$ of amplitudes by the mapping $b_A$. The binary sequence $\boldsymbol{B}_S$ of $n$ parity bits is mapped to a sequence $\boldsymbol{S}$ of $n$ sign symbols via:
$$b_S(z) = \begin{cases} -1, & z = 1 \\ +1, & z = 0. \end{cases} \tag{1.22}$$

6. The sequences $\boldsymbol{A}$ and $\boldsymbol{S}$ of $n$ amplitudes and signs, respectively, are multiplied element-wise and scaled by $\Delta$ to obtain transmit symbols. Given the distribution of the bits $\boldsymbol{B}_S$, the transmitted symbols will have a distribution symmetric around zero.

The described approach requires channel codes of rates $R = \frac{L}{L+1}$. The scheme can be modified to use channel codes with $R \geq \frac{L}{L+1}$. The modification is presented with dotted arrows in Figure 1.3. First, observe that for coding rates $R > \frac{L}{L+1}$ the number of parity bits in the sequence $\boldsymbol{B}_S$ will be less than the number of transmitted symbols $n$. The missing bits are generated from the data sequence $\boldsymbol{U}_2$, see the dotted arrows in Figure 1.3. After encoding, the bits $\boldsymbol{U}_2$ are appended to the parity bits $\boldsymbol{B}_S$. The length $k_2$ of the sequence $\boldsymbol{U}_2$ is selected such that $k_2 + l(\boldsymbol{B}_S) = n$. With this modification, the number of generated sign symbols $\boldsymbol{S}$ matches the number of amplitudes $\boldsymbol{A}$. In addition, the bits in the sequence $\boldsymbol{U}_2$ are IID Bernoulli-0.5, like the parity bits $\boldsymbol{B}_S$, resulting in a symmetric probability distribution of the transmit symbols.

## 1.3. Block-to-Block Distribution Matching

We consider a one-to-one block-to-block (b2b) distribution matcher (DM) which is a bijective mapping $f_{\mathrm{DM}}$ from binary data sequences $\boldsymbol{u} \in \{0,1\}^k$ to output sequences

(codewords) $\boldsymbol{c} \in \mathcal{C} \subseteq \mathcal{A}^n$, i.e., we have

$$f_{\mathrm{DM}} : \{0,1\}^k \to \mathcal{C} \subseteq \mathcal{A}^n. \tag{1.23}$$

The set $\mathcal{C}$ of codewords, i.e., the image of $f_{\mathrm{DM}}$, constitutes a codebook of a DM. The ratio $R = \frac{k}{n}$ is called the *matching rate*. We assume that the binary input sequences are uniformly distributed, i.e., the input to the DM is a random variable (RV) $\boldsymbol{U}$ with probability mass function (PMF) $P_{\boldsymbol{U}}(\boldsymbol{u}) = 2^{-k}$ for $\boldsymbol{u} \in \{0,1\}^k$. The output $\tilde{\boldsymbol{A}}$ of a DM is thus a RV which is uniformly distributed on $\mathcal{C}$, that is, $\tilde{\boldsymbol{A}} = f_{\mathrm{DM}}(\boldsymbol{U})$ with PMF $P_{\tilde{\boldsymbol{A}}}(\boldsymbol{a}) = 2^{-k}$ for $\boldsymbol{a} \in \mathcal{C}$.

We want the output of the DM to approximate a sequence of independent and identically distributed (IID) symbols, each distributed according to some target probability distribution $P_A$ on $\mathcal{A}$. The accuracy of the approximation is usually measured by the normalized Kullback–Leibler (KL) divergence[1] of the probability distribution of the DM's output and the probability distribution of the IID sequence $\boldsymbol{A} \sim P_A^n$:

$$\frac{1}{n}\mathbb{D}(P_{\tilde{\boldsymbol{A}}} \| P_A^n) = \frac{1}{n}\sum_{\boldsymbol{c}\in\mathcal{C}} P_{\tilde{\boldsymbol{A}}}(\boldsymbol{c}) \log \frac{P_{\tilde{\boldsymbol{A}}}(\boldsymbol{c})}{P_A^n(\boldsymbol{c})} = \frac{1}{n}\sum_{\boldsymbol{c}\in\mathcal{C}} \frac{1}{|\mathcal{C}|} \log \frac{\frac{1}{|\mathcal{C}|}}{P_A^n(\boldsymbol{c})}. \tag{1.24}$$

The divergence can be minimized by judiciously choosing a codebook $\mathcal{C}$.

**Definition 1.1: Empirical output distribution of a DM**

The empirical probability of a single symbol output by a DM is a PMF on $\mathcal{A}$ equal to the ratio of the number of specific symbols in the codebook $\mathcal{C}$ and the total size of the codebook $\mathcal{C}$ expressed in symbols, i.e., we have

$$P_{A,\mathcal{C}}(a) = \frac{1}{n|\mathcal{C}|}\sum_{\boldsymbol{c}\in\mathcal{C}} n_a(\boldsymbol{c}) \tag{1.25}$$

where $n_a(\boldsymbol{c}) := |\{i \colon c_i = a\}|$ is the number of occurrences of a symbol $a$ in the codeword $\boldsymbol{c}$.

Using Definition 1.1 the divergence can be rewritten as

$$\frac{1}{n}\mathbb{D}(P_{\tilde{\boldsymbol{A}}} \| P_A^n) = \sum_{\boldsymbol{c}\in\mathcal{C}} P_{\tilde{\boldsymbol{A}}}(\boldsymbol{c}) \log \frac{P_{\tilde{\boldsymbol{A}}}(\boldsymbol{c})}{P_{A,\mathcal{C}}^n(\boldsymbol{c})} \frac{P_{A,\mathcal{C}}^n(\boldsymbol{c})}{P_A^n(\boldsymbol{c})} \tag{1.26}$$

[1] Other metrics such as variational distance are sometimes used. The KL-divergence is interesting because it gives a lower-bound for the transmission rates [19, Eq. 5].

$$= \frac{1}{n}\mathbb{D}(P_{\tilde{\boldsymbol{A}}}\|P^n_{A,\mathcal{C}}) + \frac{1}{n}\sum_{\boldsymbol{c}\in\mathcal{C}}\frac{1}{|\mathcal{C}|}\sum_{a\in\mathcal{A}}\log\left(\frac{P_{A,\mathcal{C}}(a)}{P_A(a)}\right)^{n_a(\boldsymbol{c})} \tag{1.27}$$

$$= \frac{1}{n}\mathbb{D}(P_{\tilde{\boldsymbol{A}}}\|P^n_{A,\mathcal{C}}) + \sum_{a\in\mathcal{A}}\frac{\sum_{\boldsymbol{c}\in\mathcal{C}} n_a(\boldsymbol{c})}{n|\mathcal{C}|}\log\frac{P_{A,\mathcal{C}}(a)}{P_A(a)} \tag{1.28}$$

$$= \frac{1}{n}\mathbb{D}(P_{\tilde{\boldsymbol{A}}}\|P^n_{A,\mathcal{C}}) + \mathbb{D}(P_{A,\mathcal{C}}\|P_A). \tag{1.29}$$

By the non-negativity of divergence, a necessary condition for a DM to achieve low normalized divergence is $P_{A,\mathcal{C}} \to P_A$, i.e., the empirical output distribution of a DM must be close to the target PMF $P_A$. Now, consider the first term in (1.29):

$$\begin{aligned}\frac{1}{n}\mathbb{D}(P_{\tilde{\boldsymbol{A}}}\|P^n_{A,\mathcal{C}}) &= \frac{1}{n}\sum_{\boldsymbol{c}\in\mathcal{C}}\frac{1}{|\mathcal{C}|}\log\frac{\frac{1}{|\mathcal{C}|}}{P^n_{A,\mathcal{C}}(\boldsymbol{c})}\\ &= -\frac{1}{n}\log|\mathcal{C}| + \sum_{\boldsymbol{c}\in\mathcal{C}}\frac{1}{|\mathcal{C}|}\sum_{i=1}^{n}\log\frac{1}{P_{A,\mathcal{C}}(c_i)}\\ &= -\frac{1}{n}\log|\mathcal{C}| + \sum_{a\in\mathcal{A}}\frac{\sum_{\boldsymbol{c}\in\mathcal{C}} n_a(\boldsymbol{c})}{n|\mathcal{C}|}\log\frac{1}{P_{A,\mathcal{C}}(a)}\\ &= \mathbb{H}\left(P_{A,\mathcal{C}}\right) - \frac{\log|\mathcal{C}|}{n}\end{aligned} \tag{1.30}$$

which gives

$$\frac{1}{n}\mathbb{D}(P_{\tilde{\boldsymbol{A}}}\|P^n_A) = \mathbb{H}(P_{A,\mathcal{C}}) - \frac{\log|\mathcal{C}|}{n} + \mathbb{D}(P_{A,\mathcal{C}}\|P_A). \tag{1.31}$$

Equation (1.30) suggests that for a given $P_{A,\mathcal{C}}$, larger codebooks, i.e., with greater $|\mathcal{C}|$, are preferred due to lower divergence. The problem of designing a b2b DM corresponds to finding a function (1.23) for a given output length $n$ and target distribution $P_A$ such that the divergence (1.24) is minimized. In practice, we are interested in functions which have low-complexity implementations.

### 1.3.1. Optimal Block-to-Block DM

Consider blocklength $n$ and a target PMF $P_A$. The optimal codebook minimizes the divergence (1.24), which gives the following optimization problem [20]:

$$\min_k\left\{\min_{\mathcal{C}\subseteq\mathcal{A}^n\colon|\mathcal{C}|=2^k}\frac{1}{n}\sum_{\boldsymbol{c}\in\mathcal{C}}\frac{1}{|\mathcal{C}|}\log\frac{\frac{1}{|\mathcal{C}|}}{P^n_A(\boldsymbol{c})}\right\} = \min_k\left\{\min_{\mathcal{C}\subseteq\mathcal{A}^n\colon|\mathcal{C}|=2^k}\sum_{\boldsymbol{c}\in\mathcal{C}}\log\frac{1}{P^n_A(\boldsymbol{c})}\right\}. \tag{1.32}$$

The inner problem is solved by a codebook consisting of the $2^k$ most likely codewords according to $P_A$. The outer minimization can be performed by a line search over $k$, e.g., around $n\mathbb{H}(P_A)$.

# 2

# Arithmetic Coding in Distribution Matching

Arithmetic coding was invented as a lossless source coding scheme around 1960[1]. The idea is based on assigning intervals to source sequences according to a probability distribution of the source, and then transmitting an identifier of the interval (which can be a binary representation of a fraction from the interval). The decoder determines the interval from the identifier, and reconstructs the corresponding source sequence using the probability model of the source. Arithmetic coding can be seen as a recursive application of Shannon-Fano-Elias coding to a sequence of source symbols, and the idea is commonly contributed to Elias [22–25]. According to [26, Sec. 1.2] however, Elias denied inventing the method. The encoding and decoding algorithms are simple to describe, yet require infinite arithmetic precision. Various ideas and improvements for a fixed precision implementation were developed in [22, 27–29]. A work of Witten et al. [30] in 1987, which presents a low-complexity, integer-arithmetic implementation of a multi-symbol arithmetic coder, contributed to the present popularity of the scheme. Nowadays, arithmetic coding is used in many compression standards and algorithms, e.g., JPEG [31], WebP [32], H.264 [33], H.265 [34].

In this chapter, we use arithmetic coding to build a block-to-block (b2b) distribution matcher (DM). The idea of applying arithmetic coding for distribution matching appeared in [35]. The b2b setup was studied in [23] (a binary alphabet implementation),

[1]The idea is first mentioned in a book on information theory by Abramson [21].

and [36] (as a more general concept). For distribution matching, arithmetic coding is applied in reverse order, i.e., for encoding, an arithmetic decoder decompresses a uniformly distributed data sequence into a sequence with a given probability distribution. For decoding, an arithmetic encoder compresses the shaped sequence to the data sequence. Standard arithmetic coding can be decomposed into two main processes: estimation of the source probability distribution, and coding the input sequence using the estimated probability distribution. In the reverse setup, we also have separate processes: choosing a probability model for the shaped sequences, and coding with the selected probability model. As we shall see, choosing a suitable probability model is crucial for DM performance in terms of the code rate, divergence, and complexity.

## 2.1. Infinite Precision Implementation

We present an infinite precision implementation of an arithmetic coding based distribution matcher (ACDM). ACDM implements an invertible mapping from binary[2] input sequences $\boldsymbol{u}$ of length $k$ to non-binary sequences (codewords) $\boldsymbol{c}$ of length $n$ from the alphabet $\mathcal{A} = \{a_1, \ldots, a_m\}$, i.e., $\boldsymbol{c} \in \mathcal{A}^n$. We first describe how the mapping is performed and later focus on making the mapping act as a 'good' DM.

Each input data sequence $\boldsymbol{u}_i, i = 1, \ldots, 2^k$ corresponds to a distinct point $d(\boldsymbol{u}_i), i = 1, \ldots, 2^k$ from the interval $[0, 1)$. On the other hand, each codeword $\boldsymbol{c} \in \mathcal{A}^n$ corresponds to a distinct subinterval $I(\boldsymbol{c})$ (possibly of length 0) of the interval $[0, 1)$. The subintervals $I(\boldsymbol{c}), \boldsymbol{c} \in \mathcal{A}^n$ are chosen such that they *partition* the interval $[0, 1)$, i.e., they are pairwise disjoint and $\bigcup_{\boldsymbol{c}\in\mathcal{A}^n} I(\boldsymbol{c}) = [0, 1)$. At the encoder an input data sequence $\boldsymbol{u}$ is mapped to a codeword $\boldsymbol{c}$ if the corresponding point $d(\boldsymbol{u})$ lines inside the corresponding interval $I(\boldsymbol{c})$. At the decoder first an interval $I(\boldsymbol{c})$ is determined based on the received codeword $\boldsymbol{c}$. Then, a point $d(\boldsymbol{u}) \in I(\boldsymbol{c})$ is found and decoded to the sequence $\boldsymbol{u}$.

[2]Extensions to different input alphabet sizes are straight-forward.

**Definition 2.1: Natural $m$-ary code number**
Consider the alphabet $\mathcal{A} = \{a_1, \ldots, a_m\}$ and a sequence $\boldsymbol{x} \in \mathcal{A}^n$. The function $\mathrm{NC}_m(\cdot)$ returns a natural $m$-ary code number corresponding to the sequence $\boldsymbol{x}$, i.e.,

$$\mathrm{NC}_m(\boldsymbol{x}) = \sum_{j=1}^{n} (\mathrm{id}(x_j) - 1)\, m^{n-j} \tag{2.1}$$

where the function $\mathrm{id}(\cdot)$ returns the alphabet index of the symbol, i.e., $\mathrm{id}(a_i) = i$ for all $i$.

A binary input sequence $\boldsymbol{u} \in \{0,1\}^k$ is mapped to a point $d(\boldsymbol{u}) \in [0,1)$ via

$$d(\boldsymbol{u}) = \frac{\mathrm{NC}_2(\boldsymbol{u})}{2^k}. \tag{2.2}$$

The codeword's intervals are ordered lexicographically in the $[0,1)$ interval, with the first codeword's symbol $c_1$ being the most significant symbol. We consider the lexicographical ordering of the output alphabet symbols $a_1 < a_2 < \ldots < a_m$. That is, for two codewords $\boldsymbol{c}_1 \in \mathcal{A}^n$ and $\boldsymbol{c}_2 \in \mathcal{A}^n$, if $\mathrm{NC}_m(\boldsymbol{c}_1) < \mathrm{NC}_m(\boldsymbol{c}_2)$, the $I(\boldsymbol{c}_1)$ will be placed in the interval $[0,1)$ below the $I(\boldsymbol{c}_2)$. We describe each codeword interval $I(\boldsymbol{c})$ by it's beginning $x(\boldsymbol{c})$ and it's width $y(\boldsymbol{c})$, i.e., $I(\boldsymbol{c}) = [x(\boldsymbol{c}), x(\boldsymbol{c}) + y(\boldsymbol{c}))$[3]. An interval $I(\boldsymbol{c})$ for a codeword $\boldsymbol{c} = [c_1, \ldots, c_n]$ can be computed recursively using a chosen probability model $P_{\boldsymbol{C}}$ on the codeword's symbols. The model $P_{\boldsymbol{C}}$ is usually specified in terms of the conditional probabilities (also referred to as branching probabilities) of the next symbol given the previous symbols, i.e., $P_{C_{i+1}|\boldsymbol{C}_1^i}(\cdot|\boldsymbol{s})$, where $\boldsymbol{s}$ is a sequence denoting a prefix of the codeword. The conditional cumulative probability of a letter $c \in \mathcal{A}$ is defined as

$$F_{C_{i+1}|\boldsymbol{C}_1^i}(c|\boldsymbol{s}) = \sum_{a \le c} P_{C_{i+1}|\boldsymbol{C}_1^i}(a|\boldsymbol{s}) \tag{2.3}$$

where $a \le c$ refers to lexicographical ordering of the alphabet's symbols. Clearly, we have $F_{C_{i+1}|\boldsymbol{C}_1^i}(a_m|\boldsymbol{s}) = 1$ for any $\boldsymbol{s}$. For notational convenience we also use $F_{C_{i+1}|\boldsymbol{C}_1^i}(a_0|\boldsymbol{s}) = 0$ since $a_0 \notin \mathcal{A}$. The recursive computation of the codewords' interval for all $\boldsymbol{c} \in \mathcal{A}^n$ can be performed by iteratively applying equations (2.5) and (2.6) for $i = 0, \ldots, n-1$ :

$$x(\lambda) = 0,\ y(\lambda) = 1 \tag{2.4}$$

$$x(\boldsymbol{s}a_j) = x(\boldsymbol{s}) + y(\boldsymbol{s}) F_{C_{i+1}|\boldsymbol{C}_1^i}(a_{j-1}|\boldsymbol{s}) \text{ for } j = 1, \ldots, m, \tag{2.5}$$

[3] An interval of length 0, i.e., $I(\boldsymbol{c}) = [x(\boldsymbol{c}), x(\boldsymbol{c}))$, contains no points.

$$y(\boldsymbol{s}a_j) = y(\boldsymbol{s})P_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s}), \text{ for } j = 1, \ldots, m \tag{2.6}$$

where $\lambda$ denotes an empty sequence, and $\boldsymbol{s}a_j$ denotes a concatenation of $\boldsymbol{s}$ and $a_j$. Note that, to compute $x(\boldsymbol{c})$ and $y(\boldsymbol{c})$ for a specific codeword $\boldsymbol{c}$, one must compute $x(\boldsymbol{s}), y(\boldsymbol{s})$ for all $\boldsymbol{s} \in \{\boldsymbol{c}_1^1, \boldsymbol{c}_1^2, \ldots, \boldsymbol{c}_1^n\}$, i.e., for all prefixes of the codeword $\boldsymbol{c}$. This way, very long codewords can be generated, since there is no need to construct and store the encoding/decoding trees for the whole codebook. See Figure 2.1 or Algorithm 2.1 and 2.2 below for more details on the encoder and decoder implementations. The recursive procedure (2.4)–(2.6) gives

$$x(\boldsymbol{c}) = \sum_{\boldsymbol{c}' \in \mathcal{A}^n : \boldsymbol{c}' < \boldsymbol{c}} P_{\boldsymbol{C}}(\boldsymbol{c}') \tag{2.7}$$

$$y(\boldsymbol{c}) = \prod_{i=0}^{n-1} P_{C_{i+1}|\boldsymbol{C}_1^i}(c_{i+1}|\boldsymbol{c}_1^i) = P_{\boldsymbol{C}}(\boldsymbol{c}) \tag{2.8}$$

where $\boldsymbol{c}' < \boldsymbol{c}$ refers to the lexicographical ordering of the codewords (with the first symbol $c_1$ being the most significant symbol).

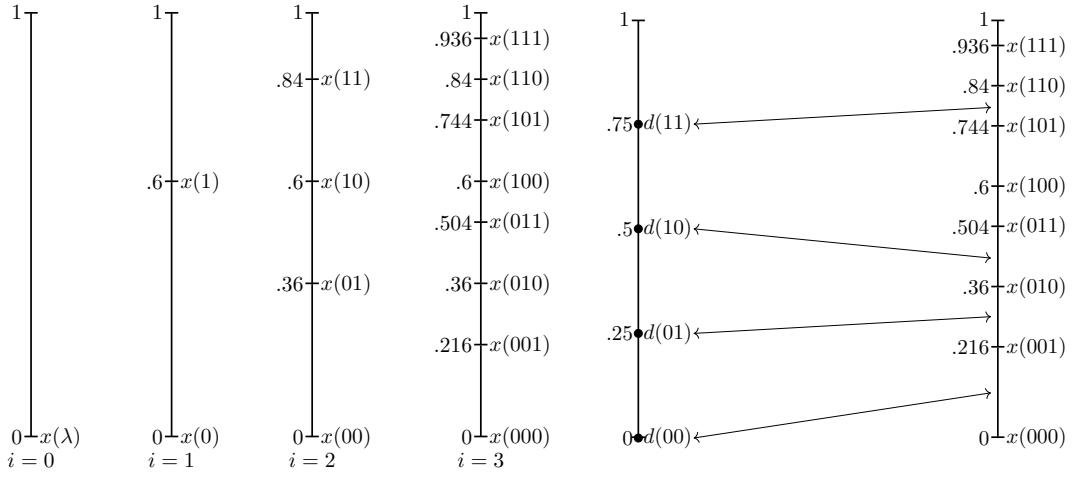
A one-to-one mapping between data sequences and codewords can be established if each datapoint $d(\boldsymbol{u}_i), i = 1, \ldots, 2^k$, belongs to some interval and if each interval $I(\boldsymbol{c}), \boldsymbol{c} \in \mathcal{A}^n$, contains at most one point $d(\boldsymbol{u})$. The first condition follows because the intervals $I(\boldsymbol{c}), \boldsymbol{c} \in \mathcal{A}^n$, partition the unit interval. The second condition can be guaranteed by letting the distance between two adjacent points be greater than the largest interval, i.e.,

$$\frac{1}{2^k} \geq \max_{\boldsymbol{c} \in \mathcal{A}^n} |I(\boldsymbol{c})| = \max_{\boldsymbol{c} \in \mathcal{A}^n} |y(\boldsymbol{c})| = \max_{\boldsymbol{c} \in \mathcal{A}^n} |P_{\boldsymbol{C}}(\boldsymbol{c})|. \tag{2.9}$$

We are interested in maximizing $k$ (since larger codebooks lead to lower divergence as in (1.31)), and thus we often choose

$$k = -\lfloor \log_2(\max_{\boldsymbol{c} \in \mathcal{A}^n} |I(\boldsymbol{c})|) \rfloor. \tag{2.10}$$

See Figure 2.1(b) for an example.




(a) Codewords' intervals. (b) Resulting mapping.

Figure 2.1.: Computing the codeword intervals for $\mathcal{A} = \{0, 1\}$, $n = 3$, and the model $P_{C_{i+1}|\boldsymbol{C}_1^i}(0|\boldsymbol{s}) = 0.6$ for any $\boldsymbol{s}$, and the resulting mapping between input sequences and codewords. The length $k$ of the input sequence satisfies $k \leq -\log_2(\max_{\boldsymbol{c}\in\mathcal{A}^n} |y(\boldsymbol{c})|) = -\log_2 0.216$.

**Definition 2.2: Model Codebook and Implementation Codebook**

For arithmetic coding based DMs only a subset of the codewords with non-zero probability (according to $P_{\boldsymbol{C}}$) will be used. Thus, we define the *model codebook* as the support of the model $P_{\boldsymbol{C}}$ used by a DM, i.e,

$$\mathcal{C}^M = \{\boldsymbol{a} \in \mathcal{A}^n \colon P_{\boldsymbol{C}}(\boldsymbol{a}) > 0\}, \tag{2.11}$$

and the *implementation codebook* as the set of codewords that are used by the DM, i.e.,

$$\mathcal{C}^I = \{\boldsymbol{a} \in \mathcal{A}^n \colon \exists_{\boldsymbol{u}\in\{0,1\}^k}\ \boldsymbol{a} = f_{\mathrm{DM}}(\boldsymbol{u})\}. \tag{2.12}$$

It follows that $\mathcal{C}^I \subseteq \mathcal{C}^M$ since the codewords outside $\mathcal{C}^M$ have intervals of zero length. The output $\tilde{\boldsymbol{A}}$ of a DM is a RV uniformly distributed on $\mathcal{C}^I$, thus the discussion in Chapter 1 implicitly considers the implementation codebook.

**Algorithm 2.1** ACDM Encoding. The encoder operates by recursively dividing the current interval and locating the subinterval which contains the point $d(\boldsymbol{u})$. Lines 3–6 find the subinterval of the current interval $I(\boldsymbol{s}) = [x(\boldsymbol{s}), x(\boldsymbol{s}) + y(\boldsymbol{s}))$ where the point $d(\boldsymbol{u})$ lies. The subinterval is associated with the alphabet symbol $a_j$, which is later appended to $\boldsymbol{s}$.

**Input:** binary sequence $\boldsymbol{u} = [u_1, \cdots, u_k]$
**Output:** codeword $\boldsymbol{c} = [c_1, \cdots, c_n]$ such that $d(\boldsymbol{u}) \in I(\boldsymbol{c})$
1: $\boldsymbol{s} \leftarrow \lambda$, $x(\lambda) \leftarrow 0$, $y(\lambda) \leftarrow 1$
2: **for** $i = 0$ **to** $n - 1$ **do**
3: $\quad j \leftarrow 1$
4: $\quad$ **while** $d(\boldsymbol{u}) < x(\boldsymbol{s}) + y(\boldsymbol{s}) F_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s})$ **do**
5: $\quad\quad j \leftarrow j + 1$
6: $\quad$ **end while**
7: $\quad x(\boldsymbol{s}a_j) \leftarrow x(\boldsymbol{s}) + y(\boldsymbol{s}) F_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s})$
8: $\quad y(\boldsymbol{s}a_j) \leftarrow y(\boldsymbol{s}) P_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s})$
9: $\quad \boldsymbol{s} \leftarrow \boldsymbol{s}a_j$
10: **end for**
11: **return** $\boldsymbol{s}$

**Algorithm 2.2** ACDM Decoding. The decoder operates by recursively dividing the current interval and selecting the next interval according to the received codewords' symbols. Finally, a fraction of the form $d = \frac{n_d}{2^k}$ needs to be located in the final interval. For properly chosen $k$, $d \geq x(\boldsymbol{s}) > d - \frac{1}{2^k} \implies n_d = \lceil x(\boldsymbol{s}) 2^k \rceil$. Next, the inverse mapping $\mathrm{NC}_2^{-1}$ is used to convert the fraction numerator $n_d$ to a binary sequence $\boldsymbol{u}$.

**Input:** codeword $\boldsymbol{c} = [c_1, \cdots, c_n]$
**Output:** binary sequence $\boldsymbol{u} = [u_1, \cdots, u_k]$ such that $d(\boldsymbol{u}) \in I(\boldsymbol{c})$
1: $\boldsymbol{s} \leftarrow \lambda$, $x(\lambda) \leftarrow 0$, $y(\lambda) \leftarrow 1$
2: **for** $i = 0$ **to** $n - 1$ **do**
3: $\quad x(\boldsymbol{s}c_{i+1}) \leftarrow x(\boldsymbol{s}) + y(\boldsymbol{s}) \left( F_{C_{i+1}|\boldsymbol{C}_1^i}(c_{i+1}|\boldsymbol{s}) - P_{C_{i+1}|\boldsymbol{C}_1^i}(c_{i+1}|\boldsymbol{s}) \right)$
4: $\quad y(\boldsymbol{s}c_i) \leftarrow y(\boldsymbol{s}) P_{C_{i+1}|\boldsymbol{C}_1^i}(c_{i+1}|\boldsymbol{s})$
5: $\quad \boldsymbol{s} \leftarrow \boldsymbol{s}c_{i+1}$
6: **end for**
7: $n_d \leftarrow \lceil x(\boldsymbol{s}) 2^k \rceil$
8: $\boldsymbol{u} \leftarrow \mathrm{NC}_2^{-1}(n_d)$
9: **return** $\boldsymbol{u}$

## 2.2. Finite Precision Implementation

The ideas presented in Section 2.1 must be adapted for practical applications. For instance, $y(\boldsymbol{s})$ in (2.6) decreases with each processed symbol which can lead to numerical underflow with a fixed precision. There has been extensive research on efficient implementations of arithmetic coding. Here we briefly present the low-complexity integer-arithmetic implementation from [23,37]. The implementation has time complexity $\mathcal{O}(n)$ and can process very long sequences using fixed precision arithmetic, e.g., using 32-bit integers. The details of the implementation show that ACDM implements a one-to-one mapping (including possible rounding errors).

We start by introducing an integer representation $\hat{F}_{\boldsymbol{C}}$ for the cumulative model $F_{\boldsymbol{C}}$ from (2.3):

$$\hat{F}_{C_{i+1}|\boldsymbol{C}_1^i}(c|\boldsymbol{s}) = \left\lfloor \Theta F_{C_{i+1}|\boldsymbol{C}_1^i}(c|\boldsymbol{s}) + \frac{1}{2} \right\rfloor \tag{2.13}$$

with $\hat{F}_{C_{i+1}|\boldsymbol{C}_1^i}(a_0|\boldsymbol{s}) = 0$, since $a_0 \notin \mathcal{A}$. The model $\hat{P}_{\boldsymbol{C}}$ is defined as

$$\hat{P}_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s}) = \hat{F}_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s}) - \hat{F}_{C_{i+1}|\boldsymbol{C}_1^i}(a_{j-1}|\boldsymbol{s}). \tag{2.14}$$

$\Theta$ is a scaling factor used for the integer representation. It effectively converts the probability models $P_{\boldsymbol{C}}, F_{\boldsymbol{C}}$ into frequency-counts models $\hat{P}_{\boldsymbol{C}}, \hat{F}_{\boldsymbol{C}}$ per $\Theta$ symbols. Next, we represent the subsequent intervals appearing in (2.4)–(2.6) by three integer numbers $\hat{x}(\boldsymbol{s}), \hat{y}(\boldsymbol{s})$, and $L(\boldsymbol{s})$. The start $x(\boldsymbol{s})$ and width $y(\boldsymbol{s})$ of the interval will be represented as binary fractions with $L(\boldsymbol{s}) + w$ bits ($w$ is a fixed parameter described below):

$$x(\boldsymbol{s}) = \frac{\hat{x}(\boldsymbol{s})}{2^{L(\boldsymbol{s})+w}}, \tag{2.15}$$

$$y(\boldsymbol{s}) = \frac{\hat{y}(\boldsymbol{s})}{2^{L(\boldsymbol{s})+w}}. \tag{2.16}$$

The recursive formulas for computing the values $\hat{x}(\boldsymbol{s}), \hat{y}(\boldsymbol{s})$, and $L(\boldsymbol{s})$ are

$$\hat{x}(\lambda) = 0,\ \hat{y}(\lambda) = 2^w,\ L(\lambda) = 0, \tag{2.17}$$

$$\hat{x}(\boldsymbol{s}a_j) = \left( \hat{x}(\boldsymbol{s}) + \left\lfloor \frac{\hat{y}(\boldsymbol{s})\hat{F}_{C_{i+1}|\boldsymbol{C}_1^i}(a_{j-1}|\boldsymbol{s})}{\Theta} + \frac{1}{2} \right\rfloor \right) 2^v \tag{2.18}$$

$$\hat{y}(\boldsymbol{s}a_j) = \left( \left\lfloor \frac{\hat{y}(\boldsymbol{s})\hat{F}_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s})}{\Theta} + \frac{1}{2} \right\rfloor - \left\lfloor \frac{\hat{y}(\boldsymbol{s})\hat{F}_{C_{i+1}|\boldsymbol{C}_1^i}(a_{j-1}|\boldsymbol{s})}{\Theta} + \frac{1}{2} \right\rfloor \right) 2^v \tag{2.19}$$

$$L(\boldsymbol{s}a_j) = L(\boldsymbol{s}) + v, \tag{2.20}$$

where $v$ is chosen such that[4]

$$2^w \leq \hat{y}(\boldsymbol{s}a_j) < 2^{w+1}. \tag{2.21}$$

The parameter $w+1$ represents the number of bits used to represent the mantissa $\hat{y}(\boldsymbol{s}a_j)$ of the current interval width $y(\boldsymbol{s}a_j)$. The scaling by $2^v$ in (2.18) and (2.19) guarantees that the mantissa $\hat{y}(\boldsymbol{s}a_j)$ is at least $2^w$. This provides sufficient precision for further subdivisions in (2.18) and (2.19) for the next symbols. Note that (2.17)–(2.19) round the interval boundaries rather than the width. The interval width (2.19) is simply a difference between two boundaries. This ensures that the original interval is partitioned during each step. Thus, the codewords intervals $I(\boldsymbol{c}), \boldsymbol{c} \in \mathcal{A}^n$ partition the starting unit interval. We want to avoid having the intervals disappear due to the rounding operations, i.e., we require

$$\hat{P}_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s}) > 0 \implies \hat{y}(\boldsymbol{s}a_j) > 0$$

which will be the case if $\frac{\hat{y}(\boldsymbol{s}a_j)}{\Theta} \geq 1$, which in turn can be guaranteed by choosing

$$2^w \geq \Theta. \tag{2.22}$$

As in the infinite precision case (2.9), to guarantee error-free decoding, we require

$$y(\boldsymbol{c}) = \frac{\hat{y}(\boldsymbol{c})}{2^{L(\boldsymbol{c})+w}} \leq \frac{1}{2^k}, \ \forall \boldsymbol{c} \in \mathcal{A}^n. \tag{2.23}$$

Jones [37] suggests using the following mapping $d : \{0,1\}^k \to [0,1)$

$$d(\boldsymbol{u}) = \frac{2^w \sum_{i=1}^{k} 2^{k-i} u_i + \sum_{i=0}^{w-1} 2^i}{2^{k+w}} \tag{2.24}$$

which corresponds to appending $w$ implicit ones to the fraction $\frac{\mathrm{NC}_2(\boldsymbol{u})}{2^k}$. This way the first $k$ digits of $\hat{x}(\boldsymbol{c})$ are the same as the digits of $\boldsymbol{u}$, if[5] $d(\boldsymbol{u}) \in I(\boldsymbol{c})$. Accordingly, the decoding corresponds to finding the first $k$ digits of $\hat{x}(\boldsymbol{c})$ given the codeword $\boldsymbol{c}$. The encoding corresponds to finding an interval that contains the point $d(\boldsymbol{u})$ and for which $\hat{x}(\boldsymbol{c})$ starts with $\boldsymbol{u}$. Both processes can thus be performed in steps, e.g., bit by bit, without processing a large number of bits at once; see below for more details. The

[4] If $\hat{P}_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s}) = 0$ for certain $\boldsymbol{s}, a_j$, then we may have $\hat{y}(\boldsymbol{s}a_j) = 0$ and a $v$ satisfying (2.21) can not be found. This does not lead to problems as the encoder will not produce codewords with the prefix $\boldsymbol{s}a_j$, so further subdivisions are not needed.

[5] By contradiction, we have $(d(\boldsymbol{u}) \in I(\boldsymbol{c}) \wedge [\hat{x}(\boldsymbol{c})]_1^k \neq \boldsymbol{u}) \implies y(\boldsymbol{c}) > \frac{1}{2^k}$.

spacing between the points (2.24) is $2^{-k}$ which, together with (2.23) and the fact that the intervals $I(\boldsymbol{c}), \boldsymbol{c} \in \mathcal{A}^n$, partition the unit interval, guarantees that ACDM implements a one-to-one[6] mapping between data sequences $\boldsymbol{u} \in \{0,1\}^k$ and codewords $\boldsymbol{c} \in \mathcal{C}^I \subseteq \mathcal{A}^n$.

The described scheme can be applied to encode and decode arbitrarily long sequences because of two properties. The first property is that one computes $\hat{x}(\boldsymbol{s}), \hat{y}(\boldsymbol{s}), L(\boldsymbol{s})$ only for $\boldsymbol{s} \in \{\boldsymbol{c}_1^1, \boldsymbol{c}_1^2, \ldots, \boldsymbol{c}_1^n\}$ when encoding/decoding a codeword $\boldsymbol{c}$. The second property is a smart arrangement of the computations in (2.17)–(2.19), where $\hat{x}(\boldsymbol{s}), \hat{y}(\boldsymbol{s}), L(\boldsymbol{s})$ can be evaluated using fixed precision arithmetic. The first property originates from the arithmetic coding approach. Below we focus on the second property.

First, to represent the value of $y(\boldsymbol{s})$ we must store the value of $\hat{y}(\boldsymbol{s})$, which requires $(w+1)$ bits, and the value of $L(\boldsymbol{s})$. The value of $y(\boldsymbol{s})$ decreases as consecutive symbols are processed since $\hat{y}(\boldsymbol{s})$ is bounded and the value of $L(\boldsymbol{s})$ increases, see (2.19)–(2.21). Thus, the binary representation of $y(\boldsymbol{s})$ will consist of $(L(\boldsymbol{s})-1)$ zeros followed by $(w+1)$ bits from $\hat{y}(\boldsymbol{s})$, see (2.25). On the other hand, the value $x(\boldsymbol{s})$ increases as consecutive symbols are processed and requires $(L(\boldsymbol{s})+w)$ bits to be represented in general. However, the representation can be arranged such that only the $w+1$ least significant bits enter the calculations. The binary representation of $x(\boldsymbol{s})$ can be divided into four parts, see (2.26). Starting from right to left we have:

1. *the working end* — the $w+1$ least significant bits which are the only bits effectively participating in the computation (2.18);

2. *a run of* 1*'s* of length $r$, possibly $r = 0$;

3. *the next bit* equal to 0, but it may be changed when there is a carry from the working which will propagate over the run of 1's;

4. *the decided bits* which can not change in the encoding process.

$$y(\boldsymbol{s}) = 0.\overbrace{000\ldots0000}^{L(\boldsymbol{s})-1 \text{ leading 0's}}\overbrace{1XX\ldots X}^{\hat{y}(\boldsymbol{s})} \tag{2.25}$$

$$x(\boldsymbol{s}) = 0.\underbrace{XX\ldots X}_{\text{decided bits}}\underbrace{Z}_{\substack{\uparrow \\ \text{next} \\ \text{bit}}}\underbrace{11\ldots1}_{\text{runs of 1's}}\underbrace{XXX\ldots X}_{\substack{\text{working end} \\ (w+1 \text{ bits})}} \tag{2.26}$$

[6]The partitioning guarantees that each point is in some interval. The spacing and (2.23) guarantee that there is at most one point in an interval.

Such an arrangement of the bits in $x(\boldsymbol{s})$ is possible since according to (2.15) and (2.17)–(2.21), $x(\boldsymbol{s})$ can only increase, and at most by $y(\boldsymbol{s})$. Such increase can result in a carry from the working end, which can propagate forward until the first occurrence of a 0 bit. The remaining bits are immune to changes. Thus, when implementing the line (2.18) we must keep track of the four parts of $x(\boldsymbol{s})$, perform the addition with the working end of $x(\boldsymbol{s})$ and the working end of $y(\boldsymbol{s})$, and update the other parts of $x(\boldsymbol{s})$ according to the addition result.

The storage of $x(\boldsymbol{s})$ thus requires $w+1$ bits[7] for the working end, $\lceil \log_2 r \rceil$ bits for the length $r$, and $\lceil \log_2 L(\boldsymbol{s}) \rceil$ bits for $L(\boldsymbol{s})$. The values of $r$ and $L(\boldsymbol{s})$ can grow without bound, however, this does not lead to limitations in practice.

### 2.2.1. Bounding the Discrepancy

The described finite-precision scheme implements a one-to-one mapping given that (2.23) is satisfied and the codeword intervals partition the unit interval. The latter condition is inherently satisfied by the rounding of the interval boundaries in (2.18) rather than rounding the intervals' lengths. The finite-precision condition (2.23) does not follow from the infinite-precision condition (2.9). This is because the intervals computed by the finite precision implementation are approximations of the infinite-precision intervals. The discrepancy is due to the model rounding (2.13)–(2.14) and the rounding operations during the computation of (2.17)–(2.19). To assess the finite-precision condition (2.23) we must bound the rounding error. From (2.19) we have

$$\frac{\hat{y}(\boldsymbol{s}a_j)}{2^v} = \left\lfloor \frac{\hat{y}(\boldsymbol{s})\hat{F}_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s})}{\Theta} + \frac{1}{2} \right\rfloor - \left\lfloor \frac{\hat{y}(\boldsymbol{s})\hat{F}_{C_{i+1}|\boldsymbol{C}_1^i}(a_{j-1}|\boldsymbol{s})}{\Theta} + \frac{1}{2} \right\rfloor \tag{2.27}$$

$$\leq \frac{\hat{y}(\boldsymbol{s})\left(\hat{F}_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s}) - \hat{F}_{C_{i+1}|\boldsymbol{C}_1^i}(a_{j-1}|\boldsymbol{s})\right)}{\Theta} + 1 \tag{2.28}$$

$$= \frac{\hat{y}(\boldsymbol{s})\hat{P}_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s})}{\Theta} + 1 \tag{2.29}$$

where the second line follows by $x - 1 < \lfloor x \rfloor \leq x$. Dividing both sides by $2^{L(\boldsymbol{s})+w}$ and using (2.16) we get

$$y(\boldsymbol{s}a_j) \leq y(\boldsymbol{s})\frac{\hat{P}_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s})}{\Theta} + 2^{-L(\boldsymbol{s})-w} \tag{2.30}$$

[7] For implementation it is convenient to add one extra bit to store the possible carry. E.g., for a 32 bit machine it is convenient to choose $w = 30$.

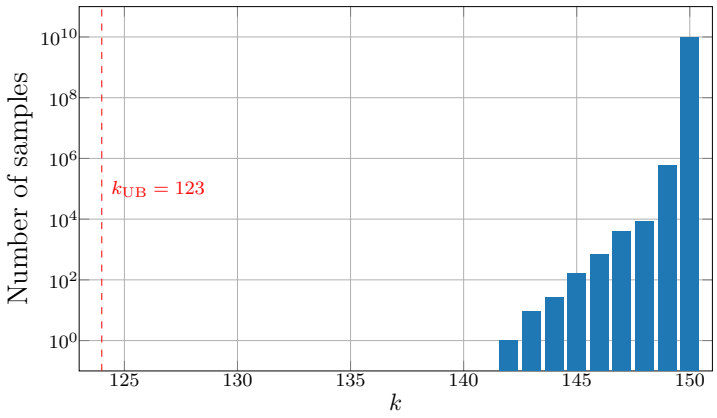



Figure 2.2.: Histogram of the Monte-Carlo samples of the right-hand-side of (2.37) for a DM from Section 2.4.1 with $w = 14$.

$$\leq y(\boldsymbol{s}) \left( P_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s}) + \epsilon \right) + 2^{-L(\boldsymbol{s})-w} \tag{2.31}$$

$$\leq y(\boldsymbol{s}) \left( P_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s}) + \epsilon + 2^{-w} \right) \tag{2.32}$$

$$= y(\boldsymbol{s}) P_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s}) \left( 1 + \frac{\epsilon + 2^{-w}}{P_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s})} \right) \tag{2.33}$$

where in the second line we introduced $\epsilon$ to denote the maximal absolute error between the infinite precision model $P_{C_{i+1}|\boldsymbol{C}_1^i}$ and the rounded model $\frac{1}{\Theta}\hat{P}_{C_{i+1}|\boldsymbol{C}_1^i}$, e.g., $\epsilon = \frac{1}{2}\Theta$ if rounding is used. The third line follows because $y(\boldsymbol{s}) \geq 2^{-L(\boldsymbol{s})}$, see (2.25). Finally, the length of the interval can be bounded by applying (2.33) for all codeword's symbols consecutively

$$y(\boldsymbol{c}) \leq P_{\boldsymbol{C}}(\boldsymbol{c}) \prod_{i=0}^{n-1} \left( 1 + \frac{\epsilon + 2^{-w}}{P_{C_{i+1}|\boldsymbol{C}_1^i}(c_{i+1}|\boldsymbol{c}_1^i)} \right) \tag{2.34}$$

which results in the bound (2.23) on the input length becoming

$$k \leq -\log_2 \left( \max_{c \in \mathcal{A}^n} P_{\boldsymbol{C}}(\boldsymbol{c}) \prod_{i=0}^{n-1} \left( 1 + \frac{\epsilon + 2^{-w}}{P_{C_{i+1}|\boldsymbol{C}_1^i}(c_{i+1}|\boldsymbol{c}_1^i)} \right) \right). \tag{2.35}$$

Inequality (2.35) connects the $k$ obtained with the fixed-precision and the infinite-precision implementations. Inequality (2.34) shows that the base intervals can dilate due to the model approximation and rounding operations. This will effectively reduce the input sequence length $k$. Using higher precision arithmetic, i.e., larger $w$ and $\Theta$, results in lower dilatation. We emphasize the importance of (2.35), since checking directly if an ACDM algorithm is one-to-one is not feasible for large values of $k$.

**Remark 2.1: Almost one-to-one ACDM for practical applications**

The bound (2.35) guarantees that the ACDM is one-to-one. The resulting choice of $k$ may be too strict since the bound assumes worst-case rounding on each encoding step, see (2.29). In practice, the rounding operations performed on each encoding/decoding step may expand or compress the interval. To obtain higher values of $k$ which guarantee correct decoding with high-probability one may resort to a Monte-Carlo-based choice of $k$. That is, the condition (2.23) can be written as

$$k \leq \log_2\left(2^{L(\boldsymbol{c})+w}\right) - \log_2\left(\hat{y}(\boldsymbol{c})\right) \quad \forall \boldsymbol{c} \in \mathcal{A}^n \tag{2.36}$$

which is satisfied if

$$k \leq L(\boldsymbol{c}) - 1 \quad \forall \boldsymbol{c} \in \mathcal{A}^n \tag{2.37}$$

since we have $\hat{y}(\boldsymbol{c}) < 2^{w+1}$ from (2.21). To approximate $k$ satisfying the condition (2.37) we can apply the encoder for $N$ randomly generated input sequences and keep track of the lowest obtained value $L(\boldsymbol{c})$. Note that using too long input sequences will not cause the encoder to fail. This approach results in larger values of $k$ than the upper bound approach (2.35), but does not guarantee a one-to-one DM. However, our results confirm that the decoding error is negligibly small for large values of $N$, see Example 2.1. The Monte-Carlo approach is also useful for probability models for which it is difficult to compute the right-hand-side of (2.35).

**Example 2.1: Almost one-to-one ACDM for practical applications**
For a DM from Section 2.4.1 we compare the input sequence length $k_{\mathrm{UB}}$ obtained from the right-hand-side of (2.35) versus the values obtained via Monte-Carlo simulation as explained in Remark 2.1. We encode $N = 10^{10}$ randomly generated input sequences $\{\boldsymbol{u}_i\}_{i=1}^N$ and keep track of the obtained values

$$k_i = L(\boldsymbol{c}_i) - 1 = L(f_{\mathrm{DM}}(\boldsymbol{u}_i)) - 1 \text{ for } i = 1, \ldots, N.$$

The histogram of the values $\{k_i\}_{i=1}^N$ is presented in Figure 2.2. Next, we choose

$$k_{\mathrm{MC}} = \min \ \{k_i\}_{i=1}^N = 142.$$

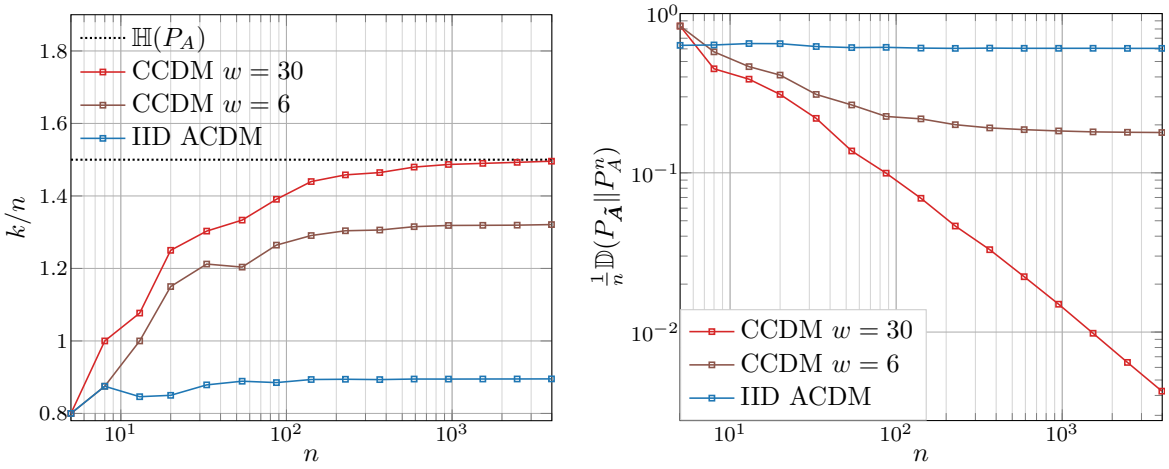
Note that this $k_{\mathrm{MC}}$ guarantees that all sequences $\{\boldsymbol{u}_i\}_{i=1}^N$ can be properly encoded and decoded, since each interval $I(\boldsymbol{c}_i)$ is shorter than $2^{-k_{\mathrm{MC}}}$, i.e., we have

$$y(\boldsymbol{c}_i) < \frac{2^{w+1}}{2^{L(\boldsymbol{c}_i)+w}} = 2^{-k_i} \leq 2^{-k_{\mathrm{MC}}}$$

where the first inequality follows by (2.16) and (2.21). We can also expect low error probability for the input sequences not included in the set $\{\boldsymbol{u}_i\}_{i=1}^N$. In Figure 2.2, we also mark the value $k_{\mathrm{UB}}$ obtained by finding the maximizer of the bound (2.35). The value $k_{\mathrm{MC}}$ is much larger than the conservative choice $k_{\mathrm{UB}}$ for the tested DM and works well in practice, i.e., has low-error probability.

## 2.3. Constant-Composition Distribution Matcher

We have seen in Section 2.1 that choosing the probability model on the output sequences $P_{\boldsymbol{C}}$ determines the ACDM. Indeed $P_{\boldsymbol{C}}$ implies all conditional (branching) probabilities used by the encoder and decoder and the maximal input length $k$. See Algorithms 2.1 and 2.2. Specifying $P_{\boldsymbol{C}}$ for all $\boldsymbol{c} \in \mathcal{A}^n$ is prohibitively complex, and therefore some simplifications are needed. It turns out that for certain probability models $P_{\boldsymbol{C}}$, the branching probabilities admit simple formulas which can be computed on-the-fly. This is the idea behind the Constant-Composition Distribution Matcher (CCDM) [36].




(a) Matching rate.

(b) Normalized divergence.

Figure 2.3.: Matching rate and normalized divergence for CCDM with the target PMF $P_A = [0.538, 0.322, 0.115, 0.025]$ and precision parameter $w \in \{6, 30\}$. IID ACDM corresponds to the ACDM from Example 2.2 and using an IID model for the branching probabilites.

The empirical distribution of a codeword $\boldsymbol{c} \in \mathcal{A}^n$ is defined as

$$P_{A,\boldsymbol{c}}(a) = \frac{n_a(\boldsymbol{c})}{n} \tag{2.38}$$

where $n_a(\boldsymbol{c}) = |\{i : c_i = a\}|$ is the number of occurrences of the symbol $a$ in the codeword $\boldsymbol{c}$, see Definition 1.1 The distribution $P_{A,\boldsymbol{c}}$ is also called the *type* of $\boldsymbol{c}$. For any sequence $\boldsymbol{c}$ we define a vector containing the numbers of occurrences in $\boldsymbol{c}$ of each of the symbols from the alphabet $\mathcal{A}$:

$$\boldsymbol{\gamma}(\boldsymbol{c}) = [n_{a_1}(\boldsymbol{c}), \ldots, n_{a_m}(\boldsymbol{c})]. \tag{2.39}$$

The vector $\boldsymbol{\gamma}(\boldsymbol{c})$ is called the *composition* of $\boldsymbol{c}$. Assume some $n$-type[8] $Q_A$ and the corresponding composition $\boldsymbol{\gamma}_Q$, i.e., $\boldsymbol{\gamma}_Q = [nQ_A(a_1), \ldots, nQ_A(a_m)]$. The set of length $n$ sequences of type $Q_A$ is denoted by $\mathcal{T}^n_{Q_A}$, and the set of the sequences of composition $\boldsymbol{\gamma}_Q$ is denoted by $\mathcal{T}_{\boldsymbol{\gamma}_Q}$. We have

$$\mathcal{T}^n_{Q_A} = \{\boldsymbol{c} \in \mathcal{A}^n \colon P_{\bar{A},\boldsymbol{c}} = Q_A\} = \{\boldsymbol{c} \in \mathcal{A}^n \colon \boldsymbol{\gamma}(\boldsymbol{c}) = \boldsymbol{\gamma}_Q\} = \mathcal{T}_{\boldsymbol{\gamma}_Q}. \tag{2.40}$$

[8] A PMF with probabilities that are multiples of $\frac{1}{n}$.

The size of the set $\mathcal{T}_{\boldsymbol{\gamma}_Q}$ is

$$|\mathcal{T}_{\boldsymbol{\gamma}_Q}| = \frac{n!}{\prod_{j=1}^{m} [\boldsymbol{\gamma}_Q]_j!} \tag{2.41}$$

where the brackets emphasize that $[\boldsymbol{\gamma}_Q]_j$ is the $j$-th entry of the vector $\boldsymbol{\gamma}_Q$.

The CCDM chooses the following model for arithmetic coding

$$P_{\boldsymbol{C}}(\boldsymbol{c}) = \begin{cases} \frac{1}{|\mathcal{T}^n_{Q_A}|}, & \text{if } \boldsymbol{c} \in \mathcal{T}^n_{Q_A} \\ 0, & \text{otherwise} \end{cases} \tag{2.42}$$

for some chosen $n$-type $Q_A$. The $n$-type $Q_A$ (and the model codebook $\mathcal{C}^M = \mathcal{T}^n_{Q_A}$) is generally chosen to minimize the divergence (1.24). The equisized intervals, i.e., equiprobable codewords, are selected to maximize the input length $k$ according to (2.9). Let the corresponding composition be $\boldsymbol{\gamma}_Q = [n_{a_1}, \ldots, n_{a_m}]$. It is easy to check that the branching probabilities are

$$P_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s}) = \frac{P_{\boldsymbol{C}_1^{i+1}}(\boldsymbol{s}a_j)}{P_{\boldsymbol{C}_1^i}(\boldsymbol{s})} = \frac{\sum_{\boldsymbol{c}\in\mathcal{A}^n\,:\,\boldsymbol{c}_1^{i+1}=\boldsymbol{s}a_j} P_{\boldsymbol{C}}(\boldsymbol{c})}{\sum_{\boldsymbol{c}\in\mathcal{A}^n\,:\,\boldsymbol{c}_1^{i}=\boldsymbol{s}} P_{\boldsymbol{C}}(\boldsymbol{c})} = \frac{\sum_{\boldsymbol{c}\in\mathcal{T}_{\boldsymbol{\gamma}_Q}\,:\,\boldsymbol{c}_1^{i+1}=\boldsymbol{s}a_j} 1}{\sum_{\boldsymbol{c}\in\mathcal{T}_{\boldsymbol{\gamma}_Q}\,:\,\boldsymbol{c}_1^{i}=\boldsymbol{s}} 1} \tag{2.43}$$

$$= \frac{|\mathcal{T}_{\boldsymbol{\gamma}_Q-\boldsymbol{\gamma}(\boldsymbol{s}a_j)}|}{|\mathcal{T}_{\boldsymbol{\gamma}_Q-\boldsymbol{\gamma}(\boldsymbol{s})}|} = \frac{\frac{(n-(i+1))!}{\prod_{k=1,\ldots,m}[\boldsymbol{\gamma}_Q-\boldsymbol{\gamma}(\boldsymbol{s}a_j)]_k!}}{\frac{(n-i)!}{\prod_{k=1,\ldots,m}[\boldsymbol{\gamma}_Q-\boldsymbol{\gamma}(\boldsymbol{s})]_k!}} = \frac{\frac{[\boldsymbol{\gamma}_Q-\boldsymbol{\gamma}(\boldsymbol{s})]_j!}{[\boldsymbol{\gamma}_Q-\boldsymbol{\gamma}(\boldsymbol{s}a_j)]_j!}}{n-i} \tag{2.44}$$

$$= \frac{[\boldsymbol{\gamma}_Q - \boldsymbol{\gamma}(\boldsymbol{s})]_j}{n-i} = \frac{n_{a_j} - n_{a_j}(\boldsymbol{s})}{n-i}, \text{ for } a_j \in \mathcal{A}. \tag{2.45}$$

One can interpret the second line as follows. Starting from the denominator: since the prefix $\boldsymbol{s}$ of the codeword was already generated, the remaining suffix must have a composition $(\boldsymbol{\gamma}_Q - \boldsymbol{\gamma}(\boldsymbol{s}))$. Thus, there are $|\mathcal{T}_{\boldsymbol{\gamma}_Q-\boldsymbol{\gamma}(\boldsymbol{s})}|$ equiprobable suffixes. For the numerator, if the first symbol of the suffix is $a_j$, then the remaining part of the suffix has the composition $(\boldsymbol{\gamma}_Q - \boldsymbol{\gamma}(\boldsymbol{s}a_j))$. There are $|\mathcal{T}_{\boldsymbol{\gamma}_Q-\boldsymbol{\gamma}(\boldsymbol{s}a_j)}|$ such remaining parts of the suffix. The probability of the next symbol being $a_j$ is thus a ratio of the mentioned terms. The probabilities (2.45) depend only on the composition of the current prefix $\boldsymbol{s}$ and can be easily computed in each encoding/decoding step. This allows for efficient implementation of the CCDM.

Note that the model (2.42) is approximated by the CCDM. By applying the algorithm from Section 2.1 or 2.2, at most $2^k, k = \lfloor \log_2 |\mathcal{T}^n_{Q_A}| \rfloor$, codewords from $\mathcal{T}^n_{Q_A}$ will be used by the CCDM. The selection is done implicitly by the arithmetic coding algorithm.

The CCDM is asymptotically optimal [36], as shown below.

**Remark 2.2: Properties of the CCDM [36]**

1. Suppose CCDM uses the $n$-type $Q_A$. The matching rate $R = \frac{k}{n}$ of the CCDM satisfies
$$\lim_{n\to\infty} R = \mathbb{H}(Q_A). \tag{2.46}$$

2. Consider an arbitrary target distribution $P_A$ on the output alphabet $\mathcal{A}$, and a CCDM that chooses the $n$-type $Q_A$
$$Q_A = \operatorname*{argmin}_{Q'_A} \mathbb{D}\left(Q'_A \| P_A\right) \text{ s.t. } Q'_A \text{ is } n\text{-type}. \tag{2.47}$$
Then we have
$$\lim_{n\to\infty} \frac{1}{n} \mathbb{D}(P_{\tilde{\boldsymbol{A}}} \| P_A^n) = 0. \tag{2.48}$$

**Example 2.2: ACDM with an IDD Model**

Consider a DM with output length $n$ and a target PMF $P_A$. As an alternative to the CCDM we consider an ACDM with an IID model as follows
$$P_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s}) = P_A(a_j). \tag{2.49}$$

We observe in Figure 2.4 that the empirical output distribution (Definition 1.1) for such a DM approaches the target PMF $P_A$. Consider now the infinite precision implementation of such DM. The largest codeword interval has length
$$\left(P_A(a^*)\right)^n \text{ for } a^* = \operatorname*{argmax}_{a\in\mathcal{A}} P_A(a)$$
which leads to reduced matching rate (smaller implementation codebook) as compared to the CCDM. For the IID ACDM we have $\frac{k}{n} \leq -\log_2 P_A(a^*)$, and for the CCDM we have $\frac{k}{n} \leq \mathbb{H}(P_A)$. The reduced matching rate increases the normalized divergence via (1.31). The results for the IID model are also presented in Figure 2.3. The IID ACDM shows poor performance in terms of the normalized divergence despite matching the empirical output distribution very well.

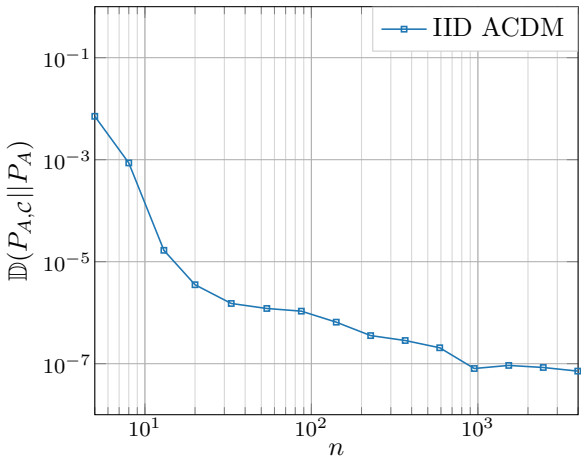



Figure 2.4.: Divergence of the empirical output distribution of the IID ACDM and the target PMF $P_A$. The plot corresponds to the the plots in Figure 2.3.

### 2.3.1. Practical Implementation

The CCDM can be implemented using the finite-precision arithmetic implementation presented in Section 2.2. The CCDM's conditional probabilities (2.45) can be represented exactly by choosing $\Theta = n-i$ in (2.13), i.e., $\Theta$ is decremented after each encoded symbol. This way the rounding is avoided and the interval length can be bounded by (2.34) with $\epsilon = 0$, i.e., we have

$$y(\boldsymbol{c}) \leq P_{\boldsymbol{C}}(\boldsymbol{c}) \prod_{i=0}^{n-1} \left(1 + \frac{2^{-w}}{P_{C_{i+1}|\boldsymbol{C}_1^i}(c_{i+1}|\boldsymbol{c}_1^i)}\right) \tag{2.50}$$

which results in a bound on the input length:

$$\begin{aligned} k &\leq -\log_2 \left( \max_{\boldsymbol{c}\in\mathcal{A}^n} P_{\boldsymbol{C}}(\boldsymbol{c}) \prod_{i=0}^{n-1} \left(1 + \frac{2^{-w}}{P_{C_{i+1}|\boldsymbol{C}_1^i}(c_{i+1}|\boldsymbol{c}_1^i)}\right)\right) && (2.51)\\ &= \log_2 |\mathcal{T}_{\boldsymbol{\gamma}}| - \max_{\boldsymbol{c}\in\mathcal{T}_{\boldsymbol{\gamma}}} \sum_{i=0}^{n-1} \log_2 \left(1 + \frac{2^{-w}}{P_{C_{i+1}|\boldsymbol{C}_1^i}(c_{i+1}|\boldsymbol{c}_1^i)}\right) && (2.52)\\ &= \log_2 |\mathcal{T}_{\boldsymbol{\gamma}}| - \Delta k. && (2.53) \end{aligned}$$

In (2.53) we introduced $\Delta k$ to denote the input length loss due to finite-precision implementation of the CCDM and $\boldsymbol{\gamma}$ is the composition used by the CCDM. $\Delta k$ denotes the input length loss before the rounding, i.e., $k$ will differ from $\log_2 |\mathcal{T}_{\boldsymbol{\gamma}}|$ by more than $\Delta k$ in general because $k$ is an integer. In fact, the optimizer of the above expression can be

found analytically for the CCDM, see Theorem 2.1. This theorem simplifies the choice of $k$ when implementing the CCDM in finite-precision arithmetic and allows to implement one-to-one CCDMs even for large values of $k$. The input length loss for different values of $w$ is presented in Table 2.1 and in Example 2.3.

| $n$ | 10 | 30 | 100 | 300 | 1000 | 3000 |
|---|---|---|---|---|---|---|
| $w = 6$ | 0.64 | 3.1 | 13.3 | 46.7 | 168.9 | 522.1 |
| $w = 30$ | 4.0e−8 | 2.1e−7 | 9.8e−7 | 4.0e−6 | 1.7e−5 | 6.2e−5 |

Table 2.1.: Input length loss $\Delta k$ due to finite-precision implementation of the CCDM and for target PMF $P_A = [0.538, 0.322, 0.115, 0.025]$. $\Delta k$ corresponds to the CCDM plots in Figure 2.3.

**Theorem 2.1: The worst-case sequence for CCDM implementation**

Consider a CCDM using the composition $\boldsymbol{\gamma} = [n_{a_1}, \ldots, n_{a_m}]$ with $n_{a_1} \leq n_{a_2} \leq \ldots \leq n_{a_m}$. A sequence

$$\boldsymbol{z} = [\underbrace{a_1 \ldots a_1}_{n_{a_1}} \underbrace{a_2 \ldots a_2}_{n_{a_2}} \ldots \underbrace{a_m \ldots a_m}_{n_{a_m}}], \tag{2.54}$$

has the largest upper bound (2.34) on the interval length among all sequences in $\mathcal{T}_{\boldsymbol{\gamma}}$, or equivalently, determines the loss

$$\Delta k = \max_{\boldsymbol{c} \in \mathcal{T}_{\boldsymbol{\gamma}}} \sum_{i=0}^{n-1} \log_2 \left( 1 + \frac{2^{-w}}{P_{C_{i+1}|\boldsymbol{C}_1^i}(c_{i+1}|\boldsymbol{c}_1^i)} \right). \tag{2.55}$$

*Proof.* See Appendix A. □

**Example 2.3: Optimal Codes**

In [23, Sec. III], T. Ramabadran considers binary CCDMs with composition $\boldsymbol{\gamma} = [\frac{n}{2}, \frac{n}{2}]$. The optimal codes (with $k = \lfloor \log_2 \binom{n}{n/2} \rfloor$) can be constructed by the CCDM when $\Delta k < 1$. $n_{\mathrm{MAX}}$ denotes the maximum length for which $\Delta k < 1$. By bounding, [23] obtains $n_{\mathrm{MAX}} \approx 2390$ for $w = 14$. By using Theorem 2.1 we obtain a tighter bound, i.e., $n_{\mathrm{MAX}} \approx 4440$ for $w = 14$, see Figure 2.5. To verify the result, we build a CCDM with $\boldsymbol{\gamma} = [1600, 1600], k = 3193$, and test the encoding and decoding for $10^{10}$ different input sequences.

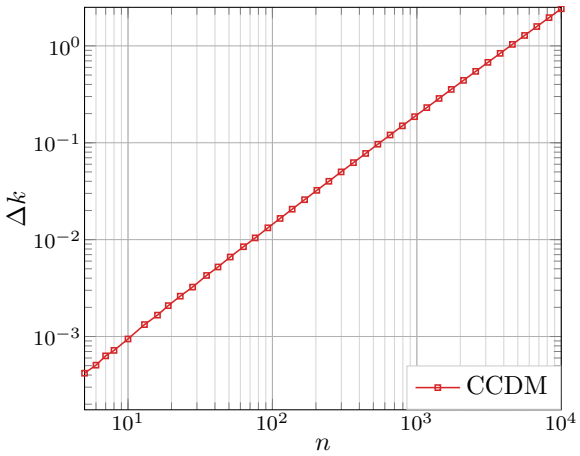



Figure 2.5.: $\Delta k$ for binary CCDM with composition $\boldsymbol{\gamma} = [\frac{n}{2}, \frac{n}{2}]$ and $w = 14$ obtained by Theorem 2.1

**Corollary 2.1: One-to-one, finite-precision CCDM is not asymptotically optimal**

[36] considers infinite-precision ACDM to implement a one-to-one CCDM. Such a DM is asymptotically optimal, as mentioned in Remark 2.2. We now consider a finite-precision implementation of a one-to-one CCDM as described in Section 2.2 and above. It turns out that the finite-precision rate-loss $\Delta k$ prevents the asymptotic optimality of the CCDM. Consider a CCDM with the finite-precision parameter $w$.

1. Suppose CCDM uses the $n$-type $Q_A$. The matching rate $R = \frac{k}{n}$ of the CCDM satisfies
$$\lim_{n\to\infty} R < \mathbb{H}(Q_A) - \log_2\left(1 + 2^{-w}\right). \tag{2.56}$$

2. Consider an arbitrary target distribution $P_A$ on the output alphabet $\mathcal{A}$, and a CCDM that chooses an arbitrary $n$-type $Q_A$. Then we have
$$\lim_{n\to\infty} \frac{1}{n}\mathbb{D}(P_{\tilde{\boldsymbol{A}}} \| P_A^n) > \log_2\left(1 + 2^{-w}\right). \tag{2.57}$$

The bounds are not tight and suffice only to show that the finite-precision implementation is not asymptotically optimal. Tighter bounds can be obtained by evaluating $\Delta k$ from Theorem 2.1. Figure 2.3 demonstrates the non-optimality of the finite-precision CCDM by using a rather low value $w = 6$ (a convenient value for an implementation on a machine with 8-bit arithmetic).

*Proof.* From (2.53) we have

$$R = \frac{k}{n} \leq \frac{1}{n}\log_2 |\mathcal{T}_{\boldsymbol{\gamma}}| - \frac{\Delta k}{n} \tag{2.58}$$

$$< \mathbb{H}(Q_A) - \max_{\boldsymbol{c}\in\mathcal{T}_{\boldsymbol{\gamma}}} \frac{1}{n}\sum_{i=0}^{n-1} \log_2\left(1 + 2^{-w}\frac{1}{P_{C_{i+1}|\boldsymbol{C}_1^i}(c_{i+1}|\boldsymbol{c}_1^i)}\right) \tag{2.59}$$

$$< \mathbb{H}(Q_A) - \log_2\left(1 + 2^{-w}\right) \tag{2.60}$$

where $\boldsymbol{\gamma}$ is the corresponding composition. The first inequality follows by the first property of the infinite-precision CCDM from Remark 2.2 and the definition of the finite-precision loss $\Delta k$ in (2.53). The second lines follows by bounding the conditional probabilities by one.

Next, using the divergence representation (1.31) for the CCDM with the $n$-type $Q_A$, we have

$$\frac{1}{n}\mathbb{D}(P_{\tilde{\boldsymbol{A}}}\|P_A^n) = \mathbb{H}(Q_A) - \frac{k}{n} + \mathbb{D}(Q_A||P_A) \tag{2.61}$$

$$> \log_2\left(1 + 2^{-w}\right) + \mathbb{D}(Q_A||P_A) \tag{2.62}$$

$$> \log_2\left(1 + 2^{-w}\right) \tag{2.63}$$

where the second line follows from (2.60). □

## 2.4. Multi-Composition Distribution Matcher

The CCDM is asymptotically optimal (with an infinite-precision implementation) and has a low-complexity implementation. However, the performance for small values of $n$ can be improved, e.g., see Figure 2.3. Equation (1.31) implies that lower divergence (and larger matching rate) can be achieved by using larger codebooks. This idea motivates Multi-Composition Distribution Matcher (MCDM), that uses more general models $P_{\boldsymbol{C}}$ to include codewords with different compositions in the implementation codebook. In this section we present a MCDM which applies a probability model motivated by the approach from [38]. There a weight function is introduced to assign a weight to each of the symbols in the alphabet. Later, only the $2^k$ sequences with the lowest weights are used as codewords.

### 2.4.1. Weight-Constrained Codebook

Consider the output length $n$, the target PMF $P_A$, and the input length $k$. We compute (1.24) as

$$\frac{1}{n}\mathbb{D}(P_{\tilde{\boldsymbol{A}}}\|P_A^n) = \frac{1}{n}\sum_{\boldsymbol{c}\in\mathcal{C}^I}\frac{1}{|\mathcal{C}^I|}\log\frac{|\mathcal{C}^I|^{-1}}{P_A^n(\boldsymbol{c})} \tag{2.64}$$

$$= -\frac{k}{n} + \frac{1}{n2^k}\sum_{\boldsymbol{c}\in\mathcal{C}^I}\sum_{i=1}^{n}(-\log P_A(c_i)). \tag{2.65}$$

The divergence can be minimized by choosing the $2^k$ codewords $\boldsymbol{c}$ with the lowest metric $\sum_{i=1}^{n}(-\log P_A(c_i))$. The metric is a sum of per-symbol metrics, thus we can assign a weight function

$$w(a) = -\log P_A(a) \text{ for } a \in \mathcal{A} \tag{2.66}$$

and define the weight of a sequence as

$$w(\boldsymbol{c}) = \sum_{i=1}^{n} w(c_i).$$

The optimal implementation codebook $\mathcal{C}_{\text{opt}}^I$ consists of the $2^k$ codewords with lowest total weight:

$$\mathcal{C}_{\text{opt}}^I = \underset{\mathcal{C}\subseteq\mathcal{A}^n\colon |\mathcal{C}|=2^k}{\operatorname{argmin}} \sum_{\boldsymbol{c}^n\in\mathcal{C}} w(\boldsymbol{c}). \tag{2.67}$$

The idea of using such a weight function for distribution matching was first presented in [38] and was combined with shell mapping [39]. Here, we apply the weight function approach to arithmetic coding based DM. In practice we need to restrict the weights to non-negative integers, i.e., the weight function we use is an approximation of (2.66)

$$w\colon \mathcal{A} \to \mathbb{N}_0^+ \tag{2.68}$$

and the weight of the codeword $\boldsymbol{c}$ is

$$w(\boldsymbol{c}) = \sum_{i=1}^{n} w(c_i). \tag{2.69}$$

The codebook $\tilde{\mathcal{C}}_{\text{opt}}^I$ containing the $2^k$ codewords with the lowest integer weight (2.69) is thus an approximation of the optimal codebook (2.67). The mismatch is the result

of (2.68) being an approximation[9] of (2.66) and is zero for certain target PMFs, e.g., a binary distribution or a Maxwell-Boltzmann distribution. We denote a superset that contains $\tilde{\mathcal{C}}_{\text{opt}}^{I}$ by

$$\mathcal{T}_{W_0}^{n} = \{\boldsymbol{c} \in \mathcal{A}^n \colon w(\boldsymbol{c}) \leq W_0\} \tag{2.70}$$

where $W_0$ is chosen such that $|\mathcal{T}_{W_0}^{n}| \geq 2^k$. Our MCDM approximates a DM with the model codebook $\mathcal{C}^M = \mathcal{T}_{W_0}^{n}$ by choosing the model

$$P_{\boldsymbol{C}}(\boldsymbol{c}) = \begin{cases} \frac{1}{|\mathcal{T}_{W_0}^{n}|}, & \text{if } \boldsymbol{c} \in \mathcal{T}_{W_0}^{n} \\ 0, & \text{otherwise.} \end{cases} \tag{2.71}$$

The conditional probabilities for encoding/decoding are

$$P_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s}) = \frac{P_{\boldsymbol{C}_1^{i+1}}(\boldsymbol{s}a_j)}{P_{\boldsymbol{C}_1^i}(\boldsymbol{s})} = \frac{\sum_{\boldsymbol{c}\in\mathcal{A}^n \colon \boldsymbol{c}_1^{i+1}=\boldsymbol{s}a_j} P_{\boldsymbol{C}}(\boldsymbol{c})}{\sum_{\boldsymbol{c}\in\mathcal{A}^n \colon \boldsymbol{c}_1^{i}=\boldsymbol{s}} P_{\boldsymbol{C}}(\boldsymbol{c})} = \frac{\sum_{\boldsymbol{c}\in\mathcal{T}_{W_0}^{n} \colon \boldsymbol{c}_1^{i+1}=\boldsymbol{s}a_j} 1}{\sum_{\boldsymbol{c}\in\mathcal{T}_{W_0}^{n} \colon \boldsymbol{c}_1^{i}=\boldsymbol{s}} 1} \tag{2.72}$$

$$= \frac{|\mathcal{T}_{W_0-w(\boldsymbol{s})-w(a_j)}^{n-(i+1)}|}{|\mathcal{T}_{W_0-w(\boldsymbol{s})}^{n-i}|}, \text{ for } a_j \in \mathcal{A}. \tag{2.73}$$

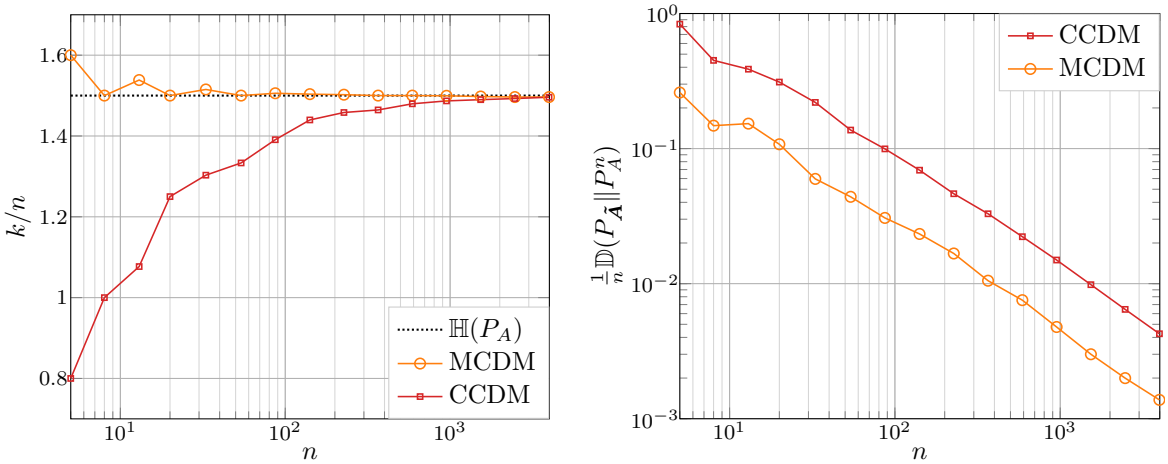
One can interpret the second line as follows. Starting from the denominator: since the prefix $\boldsymbol{s}$ of the codeword was already generated, the remaining suffix must have weight $W_0 - w(\boldsymbol{s})$ and length $n - i$. Thus, there are $|\mathcal{T}_{W_0-w(\boldsymbol{s})}^{n-i}|$ equiprobable suffixes. For the numerator, if the first symbol of the suffix is $a_j$, then the remaining part of the suffix must have weight $W_0 - w(\boldsymbol{s}) - w(a_j)$ and length $n - i - 1$. There are $|\mathcal{T}_{W_0-w(\boldsymbol{s})-w(a_j)}^{n-(i+1)}|$ such remaining parts of the suffix. The probability of the next symbol being $a_j$ is thus a ratio of the mentioned terms.

Note that the conditional probability depends only on the weight $w(\boldsymbol{s})$ of the prefix, the length $i$ of the prefix, and the symbol $a_j$. Thus, there are at most $W_0 n|\mathcal{A}|$ branching probabilities which can be stored in memory[10].

To find (2.73) it remains to compute the values $|\mathcal{T}_w^i|, i \leq n, w \leq W_0$. We use dynamic

[9] The mismatch can be reduced by approximating the scaled metric $\alpha(-\log P_A(a))$, where $\alpha > 1$. The greater the $\alpha$, the lower the integer rounding error. Note that the set of the $2^k$ codewords with the smallest metric is not changed by the metric scaling. However larger weights increase the complexity of the Shell Mapping Distribution Matcher(SMDM) [38] or MCDM. Note that shifting the weights, i.e., using $w'(a) = w(a) + b$ instead of $w(a)$, does not change the set of the $2^k$ lowest weight codewords, but can reduce the weights and the complexity of the SMDM and MCDM. An algorithm for choosing the weights for given target PMF is presented in [38].

[10] $W_0$ is at most linear in $n$ since $W_0 \leq \max_{a\in\mathcal{A}} nw(a)$. Thus, the memory requirements scale at most as $n^2$, which is often feasible.




(a) Matching rate.

(b) Normalized divergence.

Figure 2.6.: Matching rate and normalized divergence for MCDM and CCDM with the target PMF $P_A = [0.538, 0.322, 0.115, 0.025]$.

programming [40] starting with $|\mathcal{T}_0^0| = 1$, and apply iteratively the following formula for $i = 0, \ldots, n-1$ and $w = 0, \ldots, W_0$:

$$|\mathcal{T}_w^{i+1}| = \sum_{a \in \mathcal{A}} |\mathcal{T}_{w-w(a)}^i|. \tag{2.74}$$

The identity (2.74) follows because a sequence of length $i+1$ and weight $w$ must start with some symbol $a \in \mathcal{A}$. The number of sequences of length $i+1$ and weight $w$ starting with $a$ is $|\mathcal{T}_{w-w(a)}^i|$. Next, we sum length $i+1$ sequences starting with $a_1, a_2, \ldots, a_m$ to get the total number of length $i+1$ sequences.

**Remark 2.3: Branching probabilities property**

For the model (2.71), from (2.73) we have

$$P_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s}) = P_{C_1'}(a_j) \tag{2.75}$$

where the PMF $P_{\boldsymbol{C}}$ corresponds to the model $\boldsymbol{C} \sim \mathbb{U}\left[\mathcal{T}_{W_0}^n\right]$, the PMF $P_{\boldsymbol{C}'}$ corresponds to the model $\boldsymbol{C}' \sim \mathbb{U}\left[\mathcal{T}_{W_0-w(\boldsymbol{s})}^{n-i}\right]$, and $\mathbb{U}[\mathcal{B}]$ denotes a RV uniformly distributed on the set $\mathcal{B}$.

### 2.4.2. Practical Implementation

Following the implementation from Section 2.2, we scale the model probabilities (2.73) by $\Theta$ and store them as integers, see (2.13)–(2.14). We choose $\Theta = 2^w$ and additionally require

$$P_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s}) > 0 \implies \hat{P}_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s}) > 0.$$

This way no intervals are lost in the rounded model. This condition can result in an absolute representation error[11] of at most $2^{-w}$, i.e.,

$$\frac{1}{\Theta}\hat{P}_{C_{i+1}|\boldsymbol{C}_1^i}(a|\boldsymbol{s}) - P_{C_{i+1}|\boldsymbol{C}_1^i}(a|\boldsymbol{s}) < 2^{-w} = \epsilon.$$

Thus, intervals length can be bounded by (2.34) with $\epsilon = 2^{-w}$, i.e., we have

$$y(\boldsymbol{c}) \leq P_{\boldsymbol{C}}(\boldsymbol{c}) \prod_{i=0}^{n-1} \left(1 + \frac{2^{-w+1}}{P_{C_{i+1}|\boldsymbol{C}_1^i}(c_{i+1}|\boldsymbol{c}_1^i)}\right) \tag{2.76}$$

which results in

$$k \leq -\log_2 \left(\max_{\boldsymbol{c}\in\mathcal{A}^n} P_{\boldsymbol{C}}(\boldsymbol{c}) \prod_{i=0}^{n-1} \left(1 + \frac{2^{-w+1}}{P_{C_{i+1}|\boldsymbol{C}_1^i}(c_{i+1}|\boldsymbol{c}_1^i)}\right)\right) \tag{2.77}$$

$$= \min_{\boldsymbol{c}\in\mathcal{T}_{W_0}^n} \sum_{i=0}^{n-1} -\log_2 \left(P_{C_{i+1}|\boldsymbol{C}_1^i}(c_{i+1}|\boldsymbol{c}_1^i) + 2^{-w+1}\right) \tag{2.78}$$

where $\mathcal{T}_{W_0}^n$ is the superset used by the MCDM. The minimization in (2.78) can be solved efficiently using dynamic programing. Define

$$K_w^i = \min_{\boldsymbol{c}\in\mathcal{T}_w^i} \sum_{j=0}^{i-1} -\log_2 \left(P_{C_{j+1}|\boldsymbol{C}_1^j}(c_{j+1}|\boldsymbol{c}_1^j) + 2^{-w+1}\right) \tag{2.79}$$

where the branching probabilities correspond to a uniform distribution of the codewords on $\mathcal{T}_w^i$, i.e., (2.79) uses the model $\boldsymbol{C} \sim \mathbb{U}[\mathcal{T}_w^i]$. Now consider

$$K_w^{i+1} = \min_{\boldsymbol{c}\in\mathcal{T}_w^{i+1}} \sum_{j=0}^{i} -\log_2 \left(P_{C_{j+1}|\boldsymbol{C}_1^j}(c_{j+1}|\boldsymbol{c}_1^j) + 2^{-w+1}\right) \tag{2.80}$$

[11] The probabilities $P_{C_{i+1}|\boldsymbol{C}_1^i}(a|\boldsymbol{s}) \approx 0$ are represented by $\frac{1}{\Theta} = 2^{-w}$. Other probabilities for the same prefix $\boldsymbol{s}$, i.e., $P_{C_{i+1}|\boldsymbol{C}_1^i}(b|\boldsymbol{s}), b \neq a$, may be scaled down, which does not lead to problems since we are interested in the largest interval.

$$= \min_{\boldsymbol{c}\in\mathcal{T}_w^{i+1}} \; -\log_2\left(P_{C_1}(c_1)+2^{-w+1}\right) + \sum_{j=1}^{i} -\log_2\left(P_{C_{j+1}|\boldsymbol{C}_1^j}(c_{j+1}|\boldsymbol{c}_1^j)+2^{-w+1}\right) \tag{2.81}$$

$$= \min_{\boldsymbol{c}\in\mathcal{T}_w^{i+1}} \; -\log_2\left(P_{C_1}(c_1)+2^{-w+1}\right) + \underbrace{\sum_{j=0}^{i-1} -\log_2\left(P_{C_{j+1}|\boldsymbol{C}_1^j}(c_{j+2}|\boldsymbol{c}_2^{j+1})+2^{-w+1}\right)}_{\geq K^i_{w-w(c_1)} \text{ with } \boldsymbol{C}\sim\mathbb{U}\left[\mathcal{T}^i_{w-w(c_1)}\right]} \tag{2.82}$$

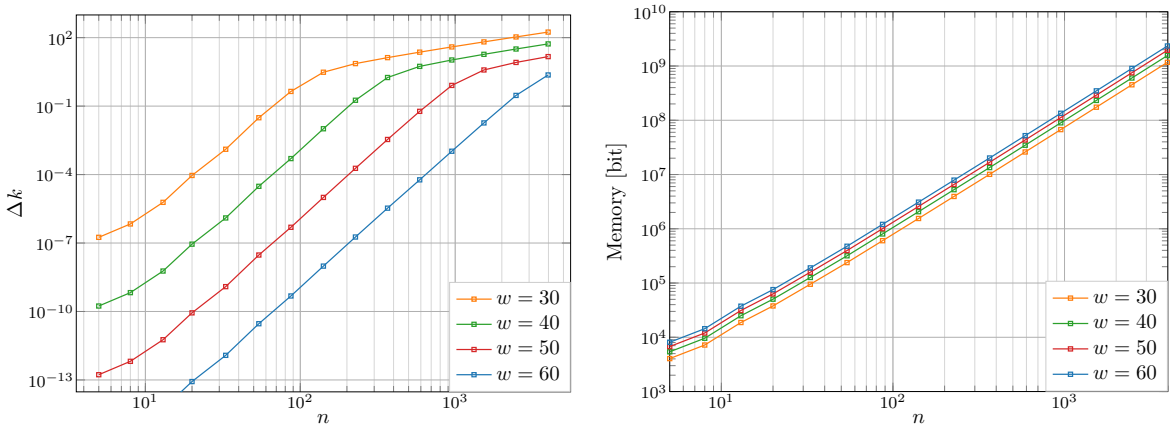
$$= \min_{a\in\mathcal{A}} -\log_2\left(P_{C_1}(a)+2^{-w+1}\right) + K^i_{w-w(a)}. \tag{2.83}$$

The fourth line follows because the branching probabilities for the model $\mathbb{U}\left[\mathcal{T}^i_{w-w(c_1)}\right]$ are the same as the branching probabilities for the model $\mathbb{U}\left[\mathcal{T}_w^{i+1}\right]$ conditioned on $c_1$ being the first codeword symbol, see (2.73) or Remark 2.3, and because the latter term in the third line is minimized by choosing a codeword $\boldsymbol{c}$ corresponding to the value $K^i_{w-w(c_1)}$. As we observe $K_w^{i+1}, w \in \{0,\dots,W_0\}$ can be computed given $K_w^i, w \in \{0,\dots,W_0\}$. This allows to efficiently find $K^n_{W_0}$ by starting with $K_i^w = 0$ for $w \in \{0,\dots,W_0\}, i = 0$ and progressing until $i = n$. Finally, observe that the branching probabilities in (2.79) are different than in (2.73) since they refer to different supersets, i.e., $\mathcal{T}_w^i$ and $\mathcal{T}^n_{W_0}$, respectively. However, the previously computed values (2.74) for the model $\boldsymbol{C}\sim\mathbb{U}\left[\mathcal{T}^n_{W_0}\right]$ contain also the values needed for the model $\boldsymbol{C}\sim\mathbb{U}\left[\mathcal{T}_w^i\right]$ (see (2.73) or Remark 2.3), thus no extra computation is needed. Finally, we set $k = \lfloor K^n_{W_0}\rfloor$, which guarantees that the DM implements a one-to-one mapping from $\{0,1\}^k$ to $\mathcal{C}^I \subseteq \mathcal{T}^n_{W_0}$.

As for the CCDM, one could define an input length loss due to fixed precision implementation (before rounding) as

$$\Delta k = \log_2|\mathcal{T}^n_{W_0}| - K^n_{W_0} \tag{2.84}$$

which is depicted in Figure 2.7(a). Observe that the presented MCDM requires relatively large precision (as compared to the CCDM) to avoid an excessive rate penalty. We speculate that this is because the branching probabilities for the MCDM can take lower values. In this case the absolute rounding error $\epsilon$ contributes to a larger interval extension. Values of $w$ larger than the machine codewords size, e.g., 32 or 64 bit, require extended precision arithmetic operations. This complicates implementation and increases processing complexity. To avoid large values of $w$ we compute also the Monte-Carlo bound on the input length from Remark 2.1. The results are presented in




(a) Input length loss

(b) Storage memory

Figure 2.7.: Input length loss due to implementation and the storage memory requirements for the discussed MCDM with the target PMF $P_A = [0.538, 0.322, 0.115, 0.025]$.

Table 2.2.

Figure 2.7(b) depicts the number of bits required to store the approximated branching probabilities. We assume that each probability is stored as a $(w+1)$-bit integer. For each prefix length and prefix weight we need to store $(|\mathcal{A}|-1)$ probabilities for all but the last symbols. The total storage requirement is thus $wW_0n(|\mathcal{A}|-1)$. We observe that the memory requirements can be a limiting factor when implementing MCDM for large $n$. E.g., for $n = 1000$ we require approximately $10^8$ bits, which corresponds to approximately 12.5 Mbytes. This is a large number if the memory needs high speed (processor cache preferably) for high throughput. The complexity of the ACDM is linear in $n$, thus the memory requirement is the only limiting factor, which we address in the next section.

**Remark 2.1: Properties of the MCDM**

Consider a target probability distribution $P_A$ defined on alphabet $\mathcal{A}$ and an MCDM using the weight function $w\colon \mathcal{A} \to \mathbb{N}_0^+$ such that

$$P_A(a_i) > P_A(a_j) \implies w(a_i) < w(a_j) \text{ for } a_i, a_j \in \mathcal{A}.$$

Suppose that the MCDM uses the probability model $\boldsymbol{C} \sim \mathbb{U}[\mathcal{T}_W^n]$ and we choose $W$ to be the weight of a sequence with a composition corresponding to an $n$-type $Q_A$ such

| $n$ | 100 | 300 | 1000 |
|---|---|---|---|
| $k_{\mathrm{MC}}, w = 14$ | 145 | 441 | 1494 |
| $k_{\mathrm{MC}}, w = 20$ | 148 | 449 | 1499 |
| $k_{\mathrm{MC}}, w = 30$ | 150 | 450 | 1500 |
| $k_{\mathrm{UB}}, w = 30$ | 149 | 439 | 1459 |
| $k_{\mathrm{UB}}, w = 40$ | 150 | 449 | 1488 |
| $k_{\mathrm{UB}}, w = 50$ | 150 | 450 | 1499 |
| $k_{\mathrm{UB}}, w = 60$ | 150 | 450 | 1500 |

Table 2.2.: Input length $k_{\mathrm{MC}}$ obtained by the Monte-Carlo bound from Remark 2.1 with $10^8$ samples and $k_{\mathrm{UB}}$ obtained from the exact bound (2.78) using dynamic programming. The target PMF is $P_A = [0.538, 0.322, 0.115, 0.025]$. The Monte-Carlo based DMs were additionally tested by encoding and decoding $10^8$ input sequences. No errors were observed.

that

$$Q_A = \operatorname*{argmin}_{Q'_A} \mathbb{D}\left(Q'_A \| P_A\right).$$

The infinite precision implementation of the MCDM is asymptotically optimal:

1. The matching rate $R = \frac{k}{n}$ satisfies

$$\lim_{n\to\infty} R = \mathbb{H}(Q_A). \tag{2.85}$$

2. The normalized divergence satisfies

$$\lim_{n\to\infty} \frac{1}{n} \mathbb{D}(P_{\tilde{\boldsymbol{A}}} \| P_A^n) = 0. \tag{2.86}$$

Now consider the MCDM in the finite-precision implementation with the precision parameter $w$. The MCDM is not asymptotically optimal:

3. The matching rate $R = \frac{k}{n}$ satisfies

$$\lim_{n\to\infty} R < \mathbb{H}(Q_A) - \log_2\left(1 + 2^{-w}\right). \tag{2.87}$$

4. The normalized divergence satisfies

$$\lim_{n\to\infty} \frac{1}{n}\mathbb{D}(P_{\tilde{\boldsymbol{A}}}\|P_A^n) > \log_2\left(1+2^{-w}\right). \tag{2.88}$$

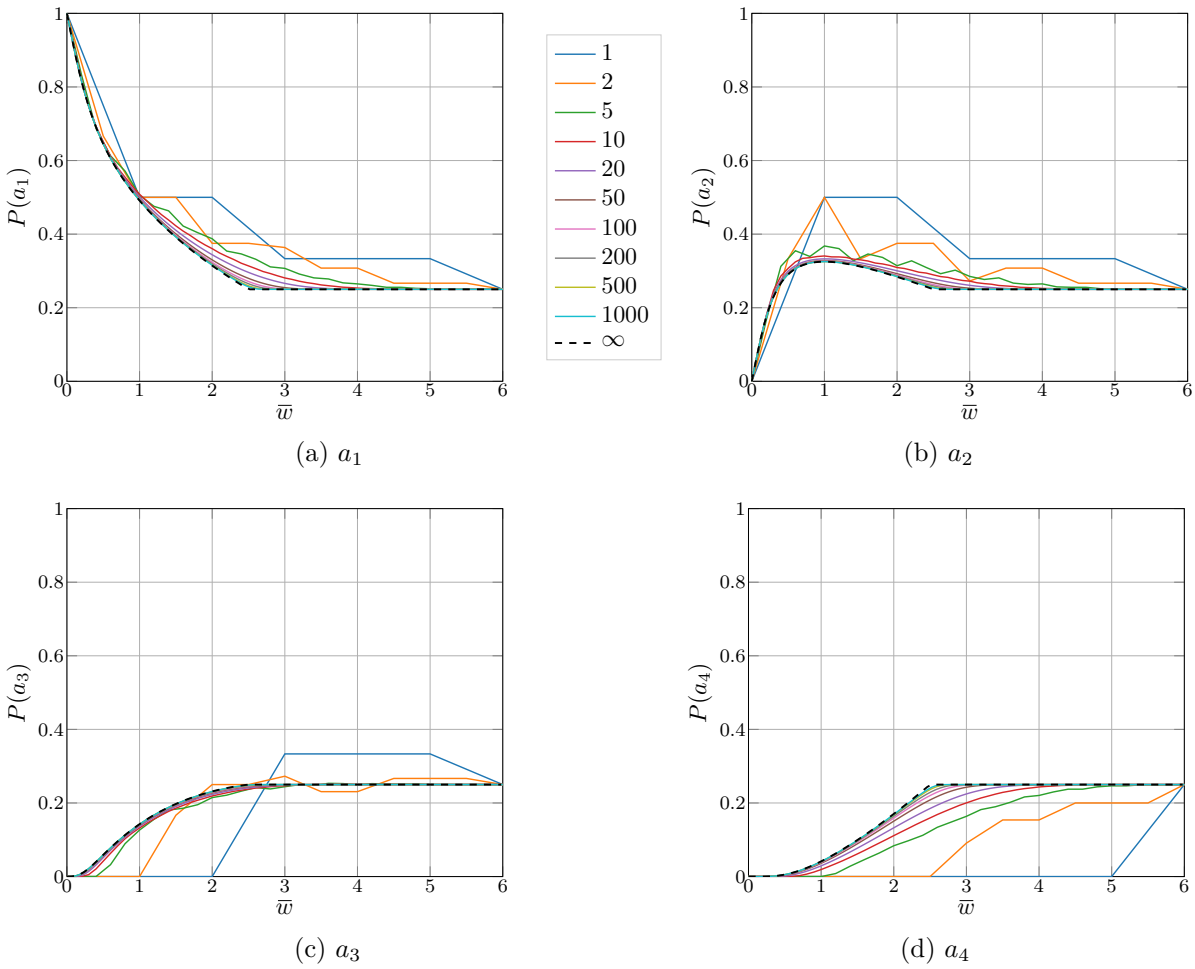
*Proof.* The second property follows since the model codebook of such an MCDM contains the model codebook of the optimal CCDM from Remark 2.2. The extra codewords in the MCDM's model codebook have lower weight and thus higher probability (according to the weight function) and lead to lower divergence via (2.65). Thus the normalized divergence of the MCDM is upper-bounded by the normalized divergence of the CCDM, which goes to zero for large $n$. See Remark 2.2.

The fist property follows since the matching rate of the MCDM is greater than or equal to the matching rate of the CCDM since the MCDM's model codebook has at least as many codewords. On the other hand, the matching rate of the MCDM is upper bounded by $\mathbb{H}(Q_A)$ since $\frac{1}{n}\mathbb{D}(P_{\tilde{\boldsymbol{A}}}\|P_A^n) \to 0$ [41, Proposition 8].

Since MCDM is asymptotically optimal, the third and fourth properties can be proved as for the CCDM in Corollary 2.1. □

### 2.4.3. Approximated Weight Constrained Codebook

Following the design rule for the ACDM from Remark 2.1 we can decouple the probability modeling process from the coding process, just like in arithmetic coding compression schemes. Basically, we can use any model $P_{\boldsymbol{C}}$ and then find $k$ such that the ACDM can decode with high probability. In this section we use this fact to simplify the architecture from Section 2.4.1. Specifically, we use the weight function and the weight constrained codebook, but we approximate some branching probabilities to lower the memory requirements. It turns out that the branching probabilities converge for large $n$ to a probability distribution specified by the average remaining weight per symbol. Theorem 2.2 makes the idea precise and an example is shown in Figure 2.8.




(a) $a_1$ (b) $a_2$

(c) $a_3$ (d) $a_4$

Figure 2.8.: Probability of the first codewords symbol for a model as in (2.71) with the superset $\mathcal{T}^n_{n\overline{w}}$ for different values of $n$. $\overline{w}$ denotes the average remaining weight per symbol. The weight function is $w(a_1) = 0$, $w(a_2) = 1$, $w(a_3) = 3$, $w(a_4) = 6$, which corresponds to Maxwell-Boltzmann distribution on the alphabet $\mathcal{A} = \{1, 3, 5, 7\}$.

**Theorem 2.2: Asymptotic Branching Probabilities**
Consider the output alphabet $\mathcal{A} = \{a_1, \ldots, a_m\}$, a non-constant weight function $w\colon \mathcal{A} \to \mathbb{N}_0^+$ such that $w(a_1) \leq w(a_2) \leq \ldots \leq w(a_m)$, and a probability distribution on the codewords $P_{\boldsymbol{C}}$ as in (2.71) with the maximum allowed weight $n\overline{W_0}$, i.e., $\boldsymbol{C} \sim \mathbb{U}\left[\mathcal{T}^n_{n\overline{W_0}}\right]$. We assume that $w(a_1) < \overline{W_0}$. For $n \to \infty$, the probability distribution of the first symbol $C_1$ from $\boldsymbol{C}$ converges, i.e., we have

$$P_{C_1}(a_j) \xrightarrow{n\to\infty} Q_A(a_j) = \begin{cases} \dfrac{2^{-\alpha w(a_j)}}{\sum_{a\in\mathcal{A}} 2^{-\alpha w(a)}}, & \text{if } \overline{W_0} < \dfrac{1}{|\mathcal{A}|}\sum_{a\in\mathcal{A}} w(a) \\ \dfrac{1}{|\mathcal{A}|}, & \text{if } \overline{W_0} \geq \dfrac{1}{|\mathcal{A}|}\sum_{a\in\mathcal{A}} w(a) \end{cases} \tag{2.89}$$

where $\alpha$ is chosen such that $\sum_{a\in\mathcal{A}} Q_A(a)w(a) = \overline{W_0}$.

*Proof.* See Appendix A. □

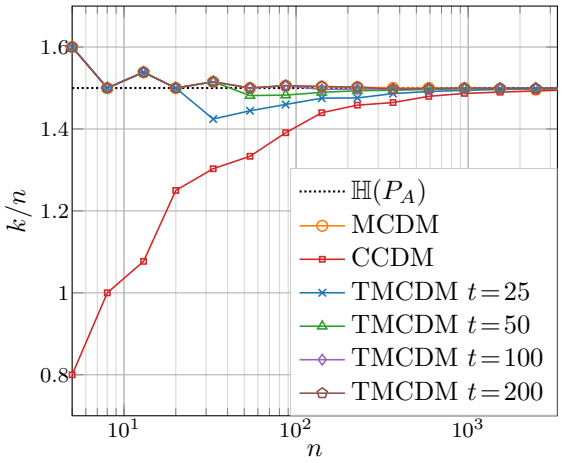
The truncated MCDM (TMCDM) works as follows. When the remaining length of the codeword is below some threshold value $t$, we use the branching probabilities as in the approach from the previous section, see (2.73). When the remaining length of the codeword is above the threshold we use the asymptotic branching probabilities from Theorem 2.2, namely

$$P_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s}) = \begin{cases} \dfrac{|\mathcal{T}^{n-(i+1)}_{W_0-w(\boldsymbol{s})-w(a_j)}|}{|\mathcal{T}^{n-i}_{W_0-w(\boldsymbol{s})}|} & \text{if } n-i \leq t \\ Q_A(a_j) \text{ with } \overline{W_0} = \dfrac{W_0 - w(\boldsymbol{s})}{n-i} & \text{if } n-i > t. \end{cases} \tag{2.90}$$

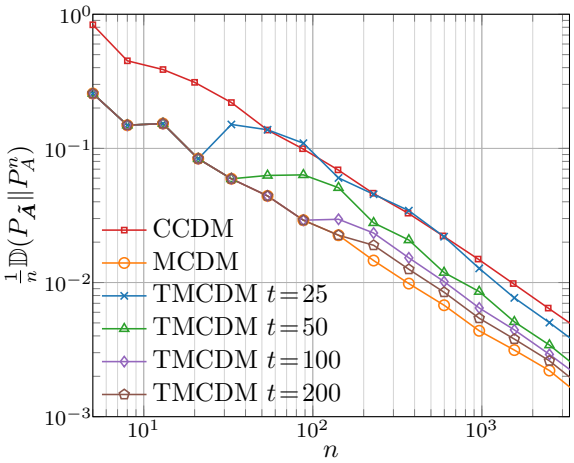
This way we need to store at most $(|\mathcal{A}|-1)t^2 w_{\mathrm{MAX}}$ with $w_{\mathrm{MAX}} = \max_{a\in\mathcal{A}} w(a)$ branching probabilities and $|\mathcal{A}| - 1$ sampled functions $Q(\overline{w})$. By choosing larger $t$ the approximations become more precise at the cost of higher memory requirements.

The results obtained for the precision parameter $w = 30$, and different values of the threshold $t$ are presented in Figure 2.9. We used the Monte Carlo based choice of $k$. In Figure 2.9(b) we observe that the approximated branching probabilities increase the divergence. The increase is lower for larg values of $t$ since the approximation is more accurate for large $t$. However, we can still improve on the CCDM. For the memory plot in Figure 2.9(c) we observe that the TMCDM memory usage diverges from the MCDM

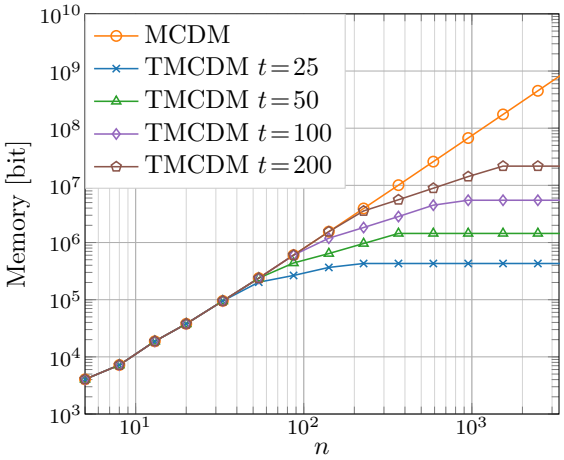
memory usage at the threshold value. The TMCDM memory usage keeps increasing after the threshold because the maximum weight for the branching probabilities up to the threshold $t$ keeps increasing. Once the look-up-table table reaches the full size $t \times tw_{\text{MAX}} \times (|\mathcal{A}| - 1)$, memory usage stays constant.




(a) Matching rate.




(b) Normalized divergence.




(c) Storage memory requirements

Figure 2.9.: Matching rate, normalized divergence, and storage memory requirements for TMCDM, MCDM, and CCDM with the target PMF $P_A = [0.538, 0.322, 0.115, 0.025]$

# 3

# Multi-Stream Distribution Matching

Multi-Stream Distribution Matching (MSDM) replaces a single DM with a non-binary output alphabet by multiple DMs with smaller output alphabets. Namely, the input sequence is first split and fed to multiple DMs. Next, the output sequences from the multiple DMs are combined and mapped to symbols from the primary output alphabet. MSDM was introduced in [42] as Bit-level Distribution Matcher (BLDM) and in [43] as Product Distribution Matching. In [42] the binary output alphabets were used for the constituting DMs (we refer to the inner DMs as *constituting* DMs) with the goal of increasing the throughput of distribution matching.

Arithmetic coding based DMs are inherently sequential. For example, see Section 2.1 and observe that the codeword symbols are produced/read sequentially. MSDM allows for parallel processing of input bits and contributes to increased throughput [44]. Moreover, the independent processing of sequences for each stream can be beneficial in some scenarios. For example, in communication systems with higher order modulation, each bit-level experiences a different channel. If one cannot recover the full message, then it may still be possible to decode the information transmitted in the better-protected bit-levels. Such an approach will not work with symbol-level distribution matching.

Since the introduction of MSDM, many new architectures for distribution matching have been proposed, e.g., a shell mapping based approach [38], enumerative coding based approaches [45,46], or the MCDM from Section 2.4.1. These approaches have storage requirements which grow with the output alphabet size. By employing MSDM, non-binary distribution matching can be performed by using only binary DMs. Moreover, the con-

stituting DMs can share the memory (the same information is needed for all constituting binary DMs). The MSDM is an interesting solution for increasing the throughput, allowing more decoding flexibility, and minimizing the memory requirements.

## 3.1. Architecture

We consider MSDM with independent streams and binary output alphabets for constituting DMs, i.e., we consider BLDM. The analysis and conclusions can often be applied when the constituting DMs use non-binary alphabets. An example of a BLDM with three bit-levels (which corresponds to the output alphabet $\mathcal{A}$ of size 8) is presented in Figure 3.1(a). The binary input sequence $\boldsymbol{U}$ is first split into three sequences: $\boldsymbol{U}_1, \boldsymbol{U}_2, \boldsymbol{U}_3$, which are input to the binary $\mathrm{DM}_1, \mathrm{DM}_2, \mathrm{DM}_3$, respectively. The dimensions of the input sequences $\boldsymbol{U}_1, \boldsymbol{U}_2, \boldsymbol{U}_3$ are different in general and depend on the parameters of $\mathrm{DM}_1, \mathrm{DM}_2, \mathrm{DM}_3$. Each constituting DM generates a binary output sequence $\tilde{\boldsymbol{B}}_l$ with a target distribution $P_{B_l}$. Next, the bits from the output sequences are mapped to the symbols from the primary alphabet $\mathcal{A}$ by a fixed, bijective mapping $f_{\mathrm{M}}$, i.e., the $i$-th symbol of the output sequence $\tilde{\boldsymbol{A}}$ is

$$[\tilde{\boldsymbol{A}}]_i = f_{\mathrm{M}}([\tilde{\boldsymbol{B}}_1]_i, [\tilde{\boldsymbol{B}}_2]_i, [\tilde{\boldsymbol{B}}_3]_i). \tag{3.1}$$

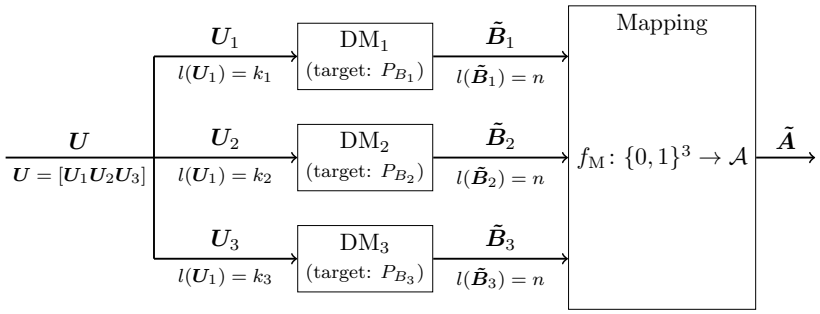
Intuitively, we expect worse divergence performance from the BLDM (as compared to the standard symbol-level DM), since the empirical output PMF $P_{A,\mathcal{C}}$ is restricted to be a product distribution.

The MSDM should be distinguished from the Split-parallelized DM (SPDM), where the constituting DMs operate with the primary output alphabet. The architecture is presented in Figure 3.1(b). We also expect worse divergence performance from the SPDM (as compared to the standard symbol-level DM), since the constituting DMs operate on shorter sequences. For long output sequences, we expect the solution to perform similarly to the standard symbol-level DM.

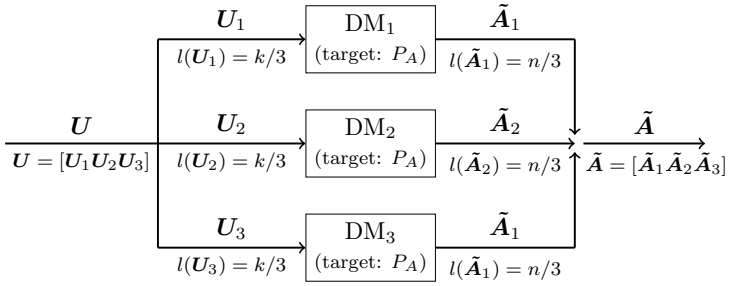
Consider now a BLDM with $L$ bit-levels, an output alphabet $\mathcal{A}$, and a target PMF $P_A$ on $\mathcal{A}$. The normalized divergence is (see (1.31) and Definition 2.2 on page 21)

$$\frac{1}{n}\mathbb{D}(P_{\tilde{\boldsymbol{A}}} \| P_A^n) = \mathbb{H}(P_{A,\mathcal{C}^I}) - \frac{\log|\mathcal{C}^I|}{n} + \mathbb{D}(P_{A,\mathcal{C}^I} || P_A). \tag{3.2}$$

Since the mapping $f_{\mathrm{M}}$ is injective, the BLDM implementation codebook (Definition 2.2)




(a) Bit-level DM




(b) Split-parallelized DM

Figure 3.1.: Architectures for parallelization and complexity reduction of a non-binary DM with the parallelization factor $L \approx 3$.

satisfies

$$|\mathcal{C}^I| = \prod_{l=1}^{L} |\mathcal{C}_l^I| \tag{3.3}$$

where $\mathcal{C}_l^I,\ l = 1, \ldots, L$, are the implementation codebooks of the constituting DMs. Denote by $P_{B_l,\mathcal{C}_l^I},\ l = 1, \ldots, L$, the empirical output probability distributions (the PMF of a single symbol output by the DM, i.e., Definition 1.1) of the constituting DMs. To decompose the empirical output PMF as

$$P_{A,\mathcal{C}^I} = \prod_{l=1}^{L} P_{B_l,\mathcal{C}_l^I} \tag{3.4}$$

we assume that each symbol in the output sequence $\tilde{\boldsymbol{B}}_l$ for $l = 1, \ldots, L$ has approximately the same marginal probability distribution.[1] The assumption is justified because many DM architectures select codebooks using additive metrics, e.g., codeword weight or composition, and do not consider the order of the symbols in the codeword. This way all permutations of a codeword should have the same probability and the marginal distribution for each symbol should be approximately equal. Finally, the normalized divergence of the BLDM becomes

$$\begin{aligned}\frac{1}{n}\mathbb{D}(P_{\tilde{\boldsymbol{A}}} \| P_A^n) &= \mathbb{H}\left(\prod_{l=1}^{L} P_{B_l,\mathcal{C}_l^I}\right) - \frac{\log \prod_{l=1}^{L} |\mathcal{C}_l^I|}{n} + \mathbb{D}\left(\prod_{l=1}^{L} P_{B_l,\mathcal{C}_l^I} || P_A\right) \\ &= \sum_{l=1}^{L}\left(\mathbb{H}(P_{B_l,\mathcal{C}_l^I}) - \frac{\log |\mathcal{C}_l^I|}{n}\right) + \mathbb{D}\left(\prod_{l=1}^{L} P_{B_l,\mathcal{C}_l^I} || P_A\right). \end{aligned} \tag{3.5}$$

The first term is a sum of *rate-losses*, i.e., the differences between the target entropy and the matching rate, of the constituting DMs. In Chapter 2 we observed that for the CCDM, the rate-loss is close to zero for large values of $n$. Similarly, for the MCDM the rate-loss is close to zero, as is the case for other DMs [38, 45]. Thus, for the BLDM we focus on minimizing the latter term, which is also the minimal normalized divergence achieved by the BLDM.

[1] Otherwise (3.4) may not hold. Consider a BLDM with 2 bit-levels and the implementation codebooks of the constituting DMs: $\mathcal{C}_1^I = \{000111, 001011\}, \mathcal{C}_2^I = \{000111, 001011\}$ and the mapping $f_{\mathrm{M}}(b_1, b_2) = a_{2b_1+b_2+1}$. Bits in the output sequence have different marginal distributions and we have $P_{A,\mathcal{C}^I} \neq \prod_{l=1}^{2} P_{B_l,\mathcal{C}_l^I}$. Namely, $P_{B_1,\mathcal{C}_1^I}(0) = P_{B_1,\mathcal{C}_1^I}(1) = P_{B_2,\mathcal{C}_2^I}(0) = P_{B_2,\mathcal{C}_2^I}(1) = 0.5$ and $P_{A,\mathcal{C}^I}(a_1) = P_{A,\mathcal{C}^I}(a_4) = \frac{10}{24}$, $P_{A,\mathcal{C}^I}(a_2) = P_{A,\mathcal{C}^I}(a_3) = \frac{2}{24}$.

We follow the steps as in (1.30) to compute

$$\mathbb{H}(P_{B_l,\mathcal{C}_l^I}) - \frac{\log|\mathcal{C}_l^I|}{n} = \frac{1}{n}\mathbb{D}(P_{\tilde{\boldsymbol{B}}_l}\|P^n_{B_l,\mathcal{C}_l^I}) \geq 0 \text{ for } l = 1,\ldots,L \tag{3.6}$$

which gives

$$\frac{1}{n}\mathbb{D}(P_{\tilde{\boldsymbol{A}}}\|P_A^n) \geq \mathbb{D}\left(\prod_{l=1}^{L} P_{B_l,\mathcal{C}_l^I}||P_A\right). \tag{3.7}$$

The right-hand-side term in (3.7), and thus the BLDM normalized divergence, is in general bounded away from zero. This is because the divergence is equal to zero only for $\prod_{l=1}^{L} P_{B_l,\mathcal{C}_l} = P_A$, and not every target distribution $P_A$ can be represented as a product of distributions.

Now consider the SPDM from Figure 3.1(b) with $L$ identical constituting DMs, each with the implementation codebook $\mathcal{C}_L^I$, and the empirical output distribution $P_{A,\mathcal{C}_L^I}$ equal to empirical output distribution $P_{A,\mathcal{C}^I}$ of the SPDM. Since the output sequences of the constituting DMs are concatenated, it follows that

$$|\mathcal{C}^I| = |\mathcal{C}_L^I|^L. \tag{3.8}$$

The divergence of the SPDM becomes

$$\begin{aligned}\frac{1}{n}\mathbb{D}(P_{\tilde{\boldsymbol{A}}}\|P_A^n) &= \mathbb{H}(P_{A,\mathcal{C}_L^I}) - \frac{\log|\mathcal{C}_L^I|^L}{n} + \mathbb{D}(P_{A,\mathcal{C}_L^I}||P_A) \\ &= \left(\mathbb{H}(P_{A,\mathcal{C}_L^I}) - \frac{\log|\mathcal{C}_L^I|}{\frac{n}{L}}\right) + \mathbb{D}\left(P_{A,\mathcal{C}_L^I}||P_A\right) \\ &= \frac{1}{\frac{n}{L}}\mathbb{D}(P_{\tilde{\boldsymbol{A}}_L}\|P_A^{\frac{n}{L}})\end{aligned} \tag{3.9}$$

where $\tilde{\boldsymbol{A}}_L$ is the output of a constituting DM. Thus, the divergence of the SPDM with output length $n$ is equal to the divergence of a constituting DM with output length $\frac{n}{L}$. The SPDM will thus have worse divergence performance as compared to a standard symbol-level DM, since the effective output length gets shortened by the parallelization factor $L$. However, for large $n$ the SPDM should perform similar to the standard symbol-level DM.

## 3.2. Rate-loss

Consider a DM with a target distribution $P_A$ on the alphabet $\mathcal{A}$. If the DM is asymptotically optimal, i.e., $\frac{1}{n}\mathbb{D}(P_{\tilde{\boldsymbol{A}}}\|P_A^n) \to 0$ as $n \to \infty$, then the matching rate $R = \frac{k}{n}$ is upper bounded by $\mathbb{H}(P_A)$ [41, Proposition 8]. The difference

$$R_{\text{loss}} = \mathbb{H}(P_A) - R \tag{3.10}$$

is called the rate-loss of the DM and can be decomposed as

$$nR_{\text{loss}} = \underbrace{n\mathbb{H}(P_A) - \mathbb{H}(P_{\boldsymbol{C}})}_{nR_{\text{loss}}^M} + \underbrace{\mathbb{H}(P_{\boldsymbol{C}}) - \mathbb{H}(P_{\tilde{\boldsymbol{A}}})}_{nR_{\text{loss}}^I} \tag{3.11}$$

where $P_{\boldsymbol{C}}$ is the probabilistic model of the output of the DM, e.g., $P_{\boldsymbol{C}}$ may be a uniform distribution on a set of constant-composition sequences for the CCDM. The term $nR_{\text{loss}}^M$ corresponds to the model rate-loss, i.e., the entropy difference between the IID model $P_A$ and the model used by a DM. The latter term $nR_{\text{loss}}^I$ is the implementation rate-loss due to imperfect implementation of the probabilistic model $P_{\boldsymbol{C}}$ by the DM, e.g., some sequences from the model are not used because binary input sequences of fixed length are used at the input, or some of the data points' spacing in the arithmetic-coding based DM may have to be increased to guarantee a one-to-one mapping. The implementation rate-loss can be minimized by choosing proper parameters for the DM, e.g., the precision parameter $w$ for the arithmetic-coding based DM. Then the implementation rate-loss occurs because only $2^k$ sequences from the support of $P_{\boldsymbol{C}}$ are used, and the implementation rate-loss is upper-bounded by one bit (per codeword).

### 3.2.1. Rate-loss of the BLDM and the CCDM

Consider two DMs: $\text{DM}_1$ a standard symbol-level CCDM, and $\text{DM}_2$ a BLDM with binary CCDMs used as constituting DMs. Consider an $n$-type target distribution $P_A$ and suppose $\text{DM}_1$ chooses the composition $[nP_A(a_1), \ldots, nP_A(a_{2^L})]$. Suppose that $P_A$ can be represented as a product of binary distributions for some mapping $f_{\text{M}}$, and that $\text{DM}_2$ chooses the mapping and its output distribution $Q_{A,\mathcal{C}} = \prod_{l=1}^{L} Q_{B_l,\mathcal{C}_l}$ to match the PMF $P_A$. $\text{DM}_1$ and $\text{DM}_2$ are asymptotically optimal: $\text{DM}_1$ by Remark 2.2 and $\text{DM}_2$ because the normalized divergence is equal to a sum of the constituting DMs' rate-losses, see (3.5). The model rate-loss for the BLDM is less than or equal to the model rate-loss

of the symbol-level CCDM. The model rate-loss for the CCDM is

$$nR_{\text{loss}}^{M} = n\mathbb{H}(P_A) - \mathbb{H}(P_{\boldsymbol{C}}); \tag{3.12}$$

$P_{\boldsymbol{C}}$ is uniform on the set

$$\mathcal{C}_{\text{SL}}^{M} = \{\boldsymbol{c} \in \mathcal{A}^n \colon n_a(\boldsymbol{c}) = nP_{A,\mathcal{C}}(a)\ \forall a \in \mathcal{A}\}; \tag{3.13}$$

the model rate-loss for the BLDM is

$$nR_{\text{loss}}^{M} = n\mathbb{H}(P_A) - \mathbb{H}(P_{\boldsymbol{C}}); \tag{3.14}$$

$P_{\boldsymbol{C}}$ is uniform on the set

$$\mathcal{C}_{\text{BL}}^{M} = \{\boldsymbol{c} \in \mathcal{A}^n \colon c_i = f_{\text{M}}\left([\boldsymbol{b}_1]_i, \ldots, [\boldsymbol{b}_L]_i\right), \boldsymbol{b}_l \in \{0,1\}^n \colon n_0(\boldsymbol{b}_l) = nQ_{B_l,\mathcal{C}_l}(0)\}. \tag{3.15}$$

The model codebook $\mathcal{C}_{\text{BL}}^{M}$ of the BLDM contains at least as many codewords as the model codebook $\mathcal{C}_{\text{SL}}^{M}$ of the CCDM. $\mathcal{C}_{\text{SL}}^{M}$ imposes a symbol-level constant-composition constraint, which, after mapping to binary sequences via the mapping $f_{\text{M}}^{-1}$, results in bit-level sequences that satisfy the constant-composition constraints, i.e., each codeword from the CCDMs' model codebook is also in the BLDMs' model codebook. To see this, consider $\boldsymbol{c} \in \mathcal{C}_{\text{SL}}^{M}$ and define

$$\boldsymbol{b}_l = \left[\left[f_{\text{M}}^{-1}(c_1)\right]_l, \left[f_{\text{M}}^{-1}(c_2)\right]_l, \ldots, \left[f_{\text{M}}^{-1}(c_n)\right]_l\right].$$

The number of zeros in $\boldsymbol{b}_l$ is

$$n_0(\boldsymbol{b}_l) = \sum_{\substack{a \in \mathcal{A} \\ [f_{\text{M}}^{-1}(a)]_l = 0}} nP_A(a) = n \sum_{\substack{a \in \mathcal{A} \\ [f_{\text{M}}^{-1}(a)]_l = 0}} P_A(a) = nQ_{B_l,\mathcal{C}_l}(0)$$

which shows that $\boldsymbol{c} \in \mathcal{C}_{\text{SL}}^{M} \implies \boldsymbol{c} \in \mathcal{C}_{\text{BL}}^{M}$. Thus, the model rate-loss of the BLDM is at most the rate-loss for the symbol-level CCDM when the target distribution can be represented as a product of the distributions. In practice, even when the target distribution cannot be factored, the BLDM often achieves lower rate-loss than the CCDM, but the comparison is less clear since both DMs generate different distributions.

### 3.2.2. Rate-loss of the BLDM and weight-constrained DMs

Consider two DMs: $\mathrm{DM}_1$ a standard symbol-level weight-constrained DM (e.g., the MCDM from Section 2.4.1 or the DMs [38, 45]), and $\mathrm{DM}_2$ a BLDM with binary weight-constrained DMs used as constituting DMs. Consider a target distribution $P_A$ and suppose that $\mathrm{DM}_1$ chooses the weight constraint $W_0$ to match its output distribution $P_{A,\mathcal{C}}$ to $P_A$. $\mathrm{DM}_1$ is asymptotically optimal by Remark 2.1. Suppose that $P_A$ can be represented as a product of binary distributions for some mapping $f_{\mathrm{M}}$, i.e., $P_A = \prod_{l=1}^{L} P_{B_l}$, and that $\mathrm{DM}_2$ chooses the same weight constraint $W_0$ as $\mathrm{DM}_1$. The model rate-loss for the BLDM is higher than for the symbol-level DM. First, consider the weight function used by $\mathrm{DM}_1$ (we assume no integer rounding of the weight function):

$$\begin{aligned} w(a) &= -\log_2 P_A(a) \\ &= -\log_2 \prod_{l=1}^{L} P_{B_l}\left(\left[f_{\mathrm{M}}^{-1}(a)\right]_l\right) \\ &= \sum_{l=1}^{L} w_l\left(\left[f_{\mathrm{M}}^{-1}(a)\right]_l\right). \end{aligned}$$

The weight of a symbol $a \in \mathcal{A}$ is a sum of the weights of the bits corresponding to $a$ via the mapping $f_{\mathrm{M}}$. The model rate-loss for the $\mathrm{DM}_1$ is

$$nR_{\mathrm{loss}}^{M} = n\mathbb{H}(P_A) - \mathbb{H}(P_{\boldsymbol{C}}) \tag{3.16}$$

and $P_{\boldsymbol{C}}$ is uniform on the set

$$\mathcal{C}_{\mathrm{SL}}^{M} = \{\boldsymbol{c} \in \mathcal{A}^n \colon w(\boldsymbol{c}) \le W_0\} \tag{3.17}$$

$$= \{\boldsymbol{c} \in \mathcal{A}^n \colon \sum_{l=1}^{L} w_l\left(\boldsymbol{b}_l\right) \le W_0\} \tag{3.18}$$

where $\boldsymbol{b}_l$ is the bit sequence corresponding to the $l$-th bit-level in the codeword $\boldsymbol{c}$, i.e., we have

$$\boldsymbol{b}_l = \left[\left[f_{\mathrm{M}}^{-1}(c_1)\right]_l, \left[f_{\mathrm{M}}^{-1}(c_2)\right]_l, \ldots, \left[f_{\mathrm{M}}^{-1}(c_n)\right]_l\right].$$

The model rate-loss for the BLDM is

$$nR_{\mathrm{loss}}^{M} = n\mathbb{H}(P_A) - \mathbb{H}(P_{\boldsymbol{C}}) \tag{3.19}$$

and $P_{\boldsymbol{C}}$ is uniform on the set

$$\mathcal{C}_{\mathrm{BL}}^{M} = \{\boldsymbol{c} \in \mathcal{A}^n \colon w_l\left(\boldsymbol{b}_l\right) \leq W_l,\ \sum_{l=1}^{L} W_l = W_0\}. \tag{3.20}$$

Observe that $\boldsymbol{c} \in \mathcal{C}_{\mathrm{BL}}^{M} \implies \boldsymbol{c} \in \mathcal{C}_{\mathrm{SL}}^{M}$, since the model codebook of the BLDM constrains each bit-level separately and the symbol-level DM has a sum constraint. Thus, the model rate-loss for the BLDM is at least the model rate-loss for the symbol-level DM. This is the opposite behavior to the one for the CCDM.

## 3.3. Finding the BLDM Parameters

A natural problem that arises when employing the BLDM is choosing the mapping $f_{\mathrm{M}}$ and the target distribution of the constituting DMs. Following the arguments above, we aim to find the target distribution that minimizes the right-hand side term in (3.7). Since we know that good DMs achieve empirical output distributions (Definition 1.1) close to their target distribution[2], we will optimize the divergence with respect to the target distributions rather the empirical output distributions. We thus look for a solution to the following problem

$$\begin{aligned}\min_{f_{\mathrm{M}}, P_{B_1}, \ldots, P_{B_L}} \mathbb{D}\left(\prod_{l=1}^{L} P_{B_l} || P_A\right) \text{ s.t. } & f_{\mathrm{M}} \colon \{0,1\}^L \to \mathcal{A} \text{ is injective} \\ & \text{and } P_{B_l},\ l = 1, \ldots, L \text{ are PMFs.}\end{aligned} \tag{3.21}$$

The mapping $f_{\mathrm{M}}$ enters the objective function via

$$\mathbb{D}\left(\prod_{l=1}^{L} P_{B_l} || P_A\right) = \sum_{a \in \mathcal{A}} P_{\boldsymbol{B}}\left(f_{\mathrm{M}}^{-1}(a)\right) \log_2\left(\frac{P_{\boldsymbol{B}}\left(f_{\mathrm{M}}^{-1}(a)\right)}{P_A(a)}\right) \tag{3.22}$$

where $f_{\mathrm{M}}^{-1}$ is the inverse mapping from symbols to binary labels, and where we introduced $P_{\boldsymbol{B}}$ to denote the product distribution $\prod_{l=1}^{L} P_{B_l}$. Finding a solution to (3.21) requires a joint optimization with respect to both the mapping $f_{\mathrm{M}}$ and the target probabilities $P_{B_l},\ l = 1, \ldots, L$. This problem seems difficult to solve, see Example 3.1. In what follows, we consider the optimization with respect to $f_{\mathrm{M}}$ and the bit-level PMFs

[2]This is a necessary condition to achieve low divergence, see (1.31).

separately. Note that

$$\min_{f_{\mathrm{M}},P_{B_1},\ldots,P_{B_L}} \mathbb{D}\left(\prod_{l=1}^{L} P_{B_l}||P_A\right) = \min_{f_{\mathrm{M}}} \min_{P_{B_1},\ldots,P_{B_L}} \mathbb{D}\left(\prod_{l=1}^{L} P_{B_l}||P_A\right). \tag{3.23}$$

We first focus on optimizing with respect to $f_{\mathrm{M}}$ and later on finding the bit-level target PMFs that minimize the divergence for a given mapping $f_{\mathrm{M}}$.

**Example 3.1: BLDM Output Divergence**

Consider BLDM with the output alphabet $\mathcal{A}$ and a target PMF $P_A$ on $\mathcal{A}$. Consider $|\mathcal{A}| = 4$ and two bit-levels $B_1, B_2$. The divergence to optimize decomposes as

$$\mathbb{D}\left(P_{\boldsymbol{B}}||P_A\right) = -\mathbb{H}\left(P_{\boldsymbol{B}}\right) + \mathbb{E}_{\boldsymbol{B}\sim P_{\boldsymbol{B}}}\left[-\log_2 P_A\left(f_{\mathrm{M}}(\boldsymbol{B})\right)\right] \tag{3.24}$$

$$= \sum_{l=1}^{L} -H_2(P_{B_l}(0)) + \mathbb{E}_{\boldsymbol{B}\sim P_{\boldsymbol{B}}}\left[-\log_2 P_A\left(f_{\mathrm{M}}(\boldsymbol{B})\right)\right] \tag{3.25}$$

where $H_2(\cdot)$ is the binary entropy function. The optimization is with respect to the probabilities $P_{B_l}(0), l = 1, \ldots, L$, which results in the objective function

$$g\left(P_{B_1}(0), \ldots, P_{B_L}(0)\right) = \sum_{l=1}^{L} -H_2(P_{B_l}(0)) + \mathbb{E}_{\boldsymbol{B}\sim P_{\boldsymbol{B}}}\left[-\log_2 P_A\left(f_{\mathrm{M}}(\boldsymbol{B})\right)\right] \tag{3.26}$$

$$= \sum_{l=1}^{L} -H_2(P_{B_l}(0)) + \sum_{a\in\mathcal{A}} \prod_{l=1}^{L} \left(P_{B_l}(0)\right)^{1-[f_{\mathrm{M}}(a)]_l} \left(1 - P_{B_l}(0)\right)^{[f_{\mathrm{M}}(a)]_l} \log_2 \frac{1}{P_A(a)}. \tag{3.27}$$

Now consider the Hessian of the function $g$, i.e., the matrix of second derivatives with respect to the optimization parameters:

$$\nabla^2(g) = \begin{bmatrix} \frac{\partial^2 g}{\partial P_{B_1}(0)\,\partial P_{B_1}(0)} & \frac{\partial^2 g}{\partial P_{B_1}(0)\,\partial P_{B_2}(0)} & \cdots & \frac{\partial^2 g}{\partial P_{B_1}(0)\,\partial P_{B_L}(0)} \\ \frac{\partial^2 g}{\partial P_{B_2}(0)\,\partial P_{B_1}(0)} & \ddots & & \vdots \\ \vdots & & \ddots & \frac{\partial^2 g}{\partial P_{B_{L-1}}(0)\,\partial P_{B_L}(0)} \\ \frac{\partial^2 g}{\partial P_{B_L}(0)\,\partial P_{B_1}(0)} & \cdots & \frac{\partial^2 g}{\partial P_{B_L}(0)\,\partial P_{B_{L-1}}(0)} & \frac{\partial^2 g}{\partial P_{B_L}(0)\,\partial P_{B_L}(0)} \end{bmatrix} \tag{3.28}$$

which becomes

$$\nabla^2(g) = \begin{bmatrix} \dfrac{1}{P_{B_1}(0)(1-P_{B_1}(0))} & \log_2 \dfrac{P_A(a_1)P_A(a_2)}{P_A(a_3)P_A(a_4)} \\ \log_2 \dfrac{P_A(a_1)P_A(a_2)}{P_A(a_3)P_A(a_4)} & \dfrac{1}{P_{B_2}(0)(1-P_{B_2}(0))} \end{bmatrix} \tag{3.29}$$

where the arrangement of terms under the logarithms depends on the mapping $f_{\mathrm{M}}$. The Hessian is symmetric because the function $g$ is continuous in the optimization variables. Depending on the target PMF $P_A$ and the mapping $f_{\mathrm{M}}$, the matrix can be either positive-definite or indefinite (negative-definite is not possible because of the positive trace of the Hessian). A positive-definite matrix corresponds to a convex optimization problem, and in general the problem is non-convex. Similar examples can be constructed for larger alphabets $\mathcal{A}$.

---

### 3.3.1. Finding the BLDM Mapping

We investigate the problem of finding the optimal mapping $f_{\mathrm{M}}$. Consider a BLDM with $L$ bit-levels, i.e., the output alphabet $\mathcal{A}$ has $|\mathcal{A}| = 2^L$. Without loss of generality assume that the target PMF $P_A$ satisfies

$$P_A(a_1) \geq P_A(a_2) \geq \ldots \geq P_A(a_{2^L}).$$

Our approach for finding the optimal mapping is based on exhaustive search.

There are $(2^L)!$ mappings from $\mathcal{A}$ to $\{0,1\}^L$. However, by exploiting the symmetry we can significantly reduce the number of mappings that need to be investigated. Following Theorem 3.1 and 3.2, we reduce the search to the mappings for which the bit-level PMFs satisfy

$$P_{B_1}(0) \geq \ldots \geq P_{B_L}(0) \geq 0.5 \tag{3.30}$$

and which are *probabilistically ordered mappings*. The number of probabilistically ordered mappings with the bit-level PMFs satisfying (3.30) is much smaller than $(2^L)!$ and allows for an exhaustive search for some values of interest. Table 3.1 presents the number of such mappings for different values of $L$. Table 3.2 presents the two possible mappings for $L = 3$. We observe that the distribution $P_A$ from Table 3.2 can be represented as a product of distributions for one of the mappings. Thus, searching for

| $L$ | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| number of probabilistically ordered mappings | 1 | 2 | 14 | 516 |
| $(2^L)!$ | 24 | 40320 | $2.1 \times 10^{13}$ | $2.6 \times 10^{35}$ |

Table 3.1.: Number of probabilistically ordered mappings with bit-levels satisfying (3.30) versus the number of all mappings $f_{\mathrm{M}} \colon \mathcal{A} \to \{0,1\}^L$.

| $P_A$ | $a$ | $f_{\mathrm{M},1}^{-1}(a)$ | $f_{\mathrm{M},2}^{-1}(a)$ |
|---|---|---|---|
| $\frac{8}{27}$ | $a_1$ | 000 | 000 |
| $\frac{4}{27}$ | $a_2$ | 001 | 001 |
| $\frac{4}{27}$ | $a_3$ | 010 | 010 |
| $\frac{4}{27}$ | $a_4$ | 011 | 100 |
| $\frac{2}{27}$ | $a_5$ | 100 | 011 |
| $\frac{2}{27}$ | $a_6$ | 101 | 101 |
| $\frac{2}{27}$ | $a_7$ | 110 | 110 |
| $\frac{1}{27}$ | $a_8$ | 111 | 111 |

Table 3.2.: The probabilistically ordered mappings for alphabet of size 8. The mapping $f_{\mathrm{M},1}^{-1}$ achieves the minimal divergence of 0.021 (with an exhaustive search over the bit-level PMFs), the mapping $f_{\mathrm{M},2}^{-1}$ achieves zero divergence (with $P_{B_1}(0) = P_{B_2}(0) = P_{B_3}(0) = \frac{2}{3}$).

the optimal mapping is essential to minimize the divergence. Unfortunately, for larger values of $L$ it is not possible to evaluate the performance of all the mappings. Instead, a random sampling could be used to find a good mapping.

**Definition 3.1: Probabilistically Ordered Mapping**

An invertible mapping $f_{\mathrm{M}} \colon \{0,1\}^L \to \mathcal{A}$ is a probabilistically ordered mapping for the bit-level PMFs $P_{B_l}$, $l = 1, \ldots, L$, if

$$P_{\boldsymbol{B}}(f_{\mathrm{M}}^{-1}(a_1)) \geq P_{\boldsymbol{B}}(f_{\mathrm{M}}^{-1}(a_2)) \geq \ldots \geq P_{\boldsymbol{B}}(f_{\mathrm{M}}^{-1}(a_{2^L})) \tag{3.31}$$

with $P_{\boldsymbol{B}} = \prod_{l=1}^{L} P_{B_l}$.

**Theorem 3.1: BLDM Mapping Properties**

Consider the target PMF $P_A$ on the alphabet $\mathcal{A}$ such that $|\mathcal{A}| = 2^L$, and the minimization problem (3.21) which is

$$\min_{f_{\mathrm{M}}, P_{B_1}, \ldots, P_{B_L}} \sum_{\boldsymbol{b} \in \{0,1\}^L} P_{\boldsymbol{B}}(\boldsymbol{b}) \log_2 \frac{P_{\boldsymbol{B}}(\boldsymbol{b})}{P_A(f_{\mathrm{M}}(\boldsymbol{b}))} \text{ s.t. } f_{\mathrm{M}} : \{0,1\}^L \to \mathcal{A} \text{ is injective,}$$
$$P_{\boldsymbol{B}} = \prod_{l=1}^{L} P_{B_l}, \text{ and } P_{B_l},\ l = 1, \ldots, L \text{ are PMFs.}$$

Let $D^*$ be the minimum of the optimization problem (the minimum is attained since the feasible set of the optimization parameters is closed and the function is continuous on the set). Any parameters $(f_{\mathrm{M}}, P_{\boldsymbol{B}})$ that attain the minimum are called a *minimizer* of the problem. It follows that:

1. There exists a minimizer $(f_{\mathrm{M}}^{*1}, P_{\boldsymbol{B}}^{*1})$ such that $P_{B_l}^{*1}(0) \geq 0.5,\ l = 1, \ldots, L$.
2. There exists a minimizer $(f_{\mathrm{M}}^{*2}, P_{\boldsymbol{B}}^{*2})$ such that $P_{B_1}^{*2}(0) \geq \ldots \geq P_{B_L}^{*2}(0) \geq 0.5$.

*Proof.* The first property follows because we can invert the bit-level PMFs and the corresponding bits in the mapping $f_{\mathrm{M}}$ until the property is satisfied. More precisely, consider a maximizer $(f_{\mathrm{M}}^*, P_{\boldsymbol{B}}^*)$. We define a set of bit-level indices which satisfy $P_{B_l}^*(0) < 0.5$, i.e., $I = \left\{l : P_{B_l}^*(0) < 0.5\right\}$, and the following bit-level PMFs

$$P_{B_l}^{*1}(b) = \begin{cases} P_{B_l}^*(1-b), & \text{if } l \in I \\ P_{B_l}^*(b), & \text{otherwise} \end{cases} \quad \text{for } b \in \{0,1\},\ l = 1, \ldots, L \tag{3.32}$$

which are inverted whenever $P_{B_l}^*(0) < 0.5$. We have $P_{B_l}^{*1}(0) \geq 0.5,\ l = 1, \ldots, L$. Similarly, we define the mapping $f_{\mathrm{M}}^{*1}$ so that

$$\left[f_{\mathrm{M}}^{*1\,-1}(a)\right]_l = \begin{cases} 1 - \left[f_{\mathrm{M}}^{*\,-1}(a)\right]_l, & \text{if } l \in I \\ \left[f_{\mathrm{M}}^{*1\,-1}(a)\right]_l, & \text{otherwise} \end{cases} \quad \text{for } a \in \mathcal{A},\ l = 1, \ldots, L \tag{3.33}$$

which has inverted ($0 \to 1$ and $1 \to 0$) bits associated with the bit-levels from $I$. Clearly, the mapping remains injective. Let $\boldsymbol{b}' \in \{0,1\}^L$ be a binary label equal to a label $\boldsymbol{b}$ with

inverted bits on the positions from $I$, i.e., we set

$$[\boldsymbol{b}']_l = \begin{cases} 1 - [\boldsymbol{b}]_l\,, & \text{if } l \in I \\ [\boldsymbol{b}]_l\,, & \text{otherwise} \end{cases} \quad \text{for } \; l = 1, \ldots, L. \tag{3.34}$$

We have

$$P^*_{\boldsymbol{B}}(\boldsymbol{b}) = P^{*1}_{\boldsymbol{B}}(\boldsymbol{b}') \tag{3.35}$$

$$f^*_{\mathrm{M}}(\boldsymbol{b}) = f^{*1}_{\mathrm{M}}(\boldsymbol{b}') \tag{3.36}$$

which gives

$$\begin{aligned} D^* &= \sum_{\boldsymbol{b}\in\{0,1\}^L} P^*_{\boldsymbol{B}}(\boldsymbol{b}) \log_2 \frac{P^*_{\boldsymbol{B}}(\boldsymbol{b})}{P_A(f^*_{\mathrm{M}}(\boldsymbol{b}))} = \sum_{\boldsymbol{b}\in\{0,1\}^L} P^{*1}_{\boldsymbol{B}}(\boldsymbol{b}') \log_2 \frac{P^{*1}_{\boldsymbol{B}}(\boldsymbol{b}')}{P_A(f^{*1}_{\mathrm{M}}(\boldsymbol{b}'))} \\ &= \sum_{\boldsymbol{b}\in\{0,1\}^L} P^{*1}_{\boldsymbol{B}}(\boldsymbol{b}) \log_2 \frac{P^{*1}_{\boldsymbol{B}}(\boldsymbol{b})}{P_A(f^{*1}_{\mathrm{M}}(\boldsymbol{b}))} = D^{*1} \end{aligned}$$

where the second line follows by changing the order of summands in the sum over the elements of $\{0,1\}^L$. Finally, $(f^{*1}_{\mathrm{M}}, P^{*1}_{\boldsymbol{B}})$ is a maximizer as well.

The second property holds because we can arrange the order of the bit-levels' PMFs and the corresponding bits in the mapping $f_{\mathrm{M}}$ until the property is satisfied. More precisely, consider a maximizer $(f^*_{\mathrm{M}}, P^*_{\boldsymbol{B}})$ satisfying the first property. We define a sorting function $s\colon \{1,\ldots,L\} \to \{1,\ldots,L\}$, that sorts the bit-level PMFs according to the probability of zero with $P^*_{B_{s(1)}}(0) \geq P^*_{B_{s(2)}}(0) \geq \ldots \geq P^*_{B_{s(L)}}(0)$. We define the bit-level PMFs

$$P^{*2}_{B_l}(b) = P^*_{B_{s(l)}}(b) \text{ for } b \in \{0,1\},\ l = 1, \ldots, L \tag{3.37}$$

so that $P^{*2}_{B_1}(0) \geq P^{*2}_{B_2}(0) \geq \ldots \geq P^{*2}_{B_L}(0) \geq 0.5$, and we define the mapping that sorts the bits according to $s(\cdot)$ as

$$\left[f^{*2\,-1}_{\mathrm{M}}(a)\right]_l = \left[f^{*\,-1}_{\mathrm{M}}(a)\right]_{s(l)} \text{ for } a \in \mathcal{A},\ l = 1, \ldots, L. \tag{3.38}$$

Let $\boldsymbol{b}' \in \{0,1\}^L$ be a binary label equal $\boldsymbol{b}$ with bits sorted according to $s(\cdot)$:

$$[\boldsymbol{b}']_l = [\boldsymbol{b}]_{s(l)} \;\text{ for } \; l = 1, \ldots, L. \tag{3.39}$$

We have

$$P^*_{\boldsymbol{B}}(\boldsymbol{b}) = P^{*2}_{\boldsymbol{B}}(\boldsymbol{b}') \tag{3.40}$$

$$f^*_{\mathrm{M}}(\boldsymbol{b}) = f^{*2}_{\mathrm{M}}(\boldsymbol{b}') \tag{3.41}$$

which gives

$$\begin{aligned} D^* &= \sum_{\boldsymbol{b}\in\{0,1\}^L} P^*_{\boldsymbol{B}}(\boldsymbol{b}) \log_2 \frac{P^*_{\boldsymbol{B}}(\boldsymbol{b})}{P_A(f^*_{\mathrm{M}}(\boldsymbol{b}))} = \sum_{\boldsymbol{b}\in\{0,1\}^L} P^{*2}_{\boldsymbol{B}}(\boldsymbol{b}') \log_2 \frac{P^{*2}_{\boldsymbol{B}}(\boldsymbol{b}')}{P_A(f^{*2}_{\mathrm{M}}(\boldsymbol{b}'))}, \\ &= \sum_{\boldsymbol{b}\in\{0,1\}^L} P^{*2}_{\boldsymbol{B}}(\boldsymbol{b}') \log_2 \frac{P^{*2}_{\boldsymbol{B}}(\boldsymbol{b}')}{P_A(f^{*2}_{\mathrm{M}}(\boldsymbol{b}'))} = D^{*2} \end{aligned} \tag{3.42}$$

where the second line follows by changing the order of summands in the sum over the elements of $\{0,1\}^L$. Finally, $(f^{*2}_{\mathrm{M}}, P^{*2}_{\boldsymbol{B}})$ is a maximizer as well. □

**Theorem 3.2: Probabilistically Ordered Mapping is Optimal**

Consider the target PMF $P_A$ on the alphabet $\mathcal{A}$ such that $|\mathcal{A}| = 2^L$. Without loss of generality, assume that $P_A(a_1) \geq P_A(a_2) \geq \ldots \geq P_A(a_{2^L})$, and let $(f^*_{\mathrm{M}}, P^*_{\boldsymbol{B}})$ be a minimizer of the divergence (3.21). It follows that

$$P^*_{\boldsymbol{B}}({f^*_{\mathrm{M}}}^{-1}(a_1)) \geq P^*_{\boldsymbol{B}}({f^*_{\mathrm{M}}}^{-1}(a_2)) \geq \ldots \geq P^*_{\boldsymbol{B}}({f^*_{\mathrm{M}}}^{-1}(a_L)). \tag{3.43}$$

*Proof.* The proof follows by contradiction. Suppose there is a minimizer $(f^*_{\mathrm{M}}, P^*_{\boldsymbol{B}})$ which does not satisfy (3.43). It follows that there exists $i < j$ such that $P_A(a_i) > P_A(a_j)$ and $P^*_{\boldsymbol{B}}({f^*_{\mathrm{M}}}^{-1}(a_i)) < P^*_{\boldsymbol{B}}({f^*_{\mathrm{M}}}^{-1}(a_j))$. We introduce the mapping $f^{*1}_{\mathrm{M}}$ which is the same as $f^*_{\mathrm{M}}$ but swaps $a_i$ and $a_j$:

$$\begin{aligned} {f^{*1}_{\mathrm{M}}}^{-1}(a_i) &= {f^*_{\mathrm{M}}}^{-1}(a_j) \\ {f^{*1}_{\mathrm{M}}}^{-1}(a_j) &= {f^*_{\mathrm{M}}}^{-1}(a_i). \end{aligned}$$

The difference between the divergence $D^*$ achieved for the parameters $(f^*_{\mathrm{M}}, P^*_{\boldsymbol{B}})$ and the divergence $D^{*1}$ achieved for the parameters $(f^{*1}_{\mathrm{M}}, P^*_{\boldsymbol{B}})$ is

$$\begin{aligned} D^* - D^{*1} = P_{\boldsymbol{B}}(a_i) \log_2 \frac{P_{\boldsymbol{B}}(a_i)}{P_A(a_i)} + P_{\boldsymbol{B}}(a_j) \log_2 \frac{P_{\boldsymbol{B}}(a_j)}{P_A(a_j)} \\ - P_{\boldsymbol{B}}(a_j) \log_2 \frac{P_{\boldsymbol{B}}(a_j)}{P_A(a_i)} - P_{\boldsymbol{B}}(a_i) \log_2 \frac{P_{\boldsymbol{B}}(a_i)}{P_A(a_j)} \end{aligned}$$

$$= (P_{\boldsymbol{B}}(a_j) - P_{\boldsymbol{B}}(a_i)) \log_2 \frac{P_A(a_i)}{P_A(a_j)}$$
$$> 0 \tag{3.44}$$

where we used a shortened notation $P_{\boldsymbol{B}}(a)$ for the term $P^*_{\boldsymbol{B}}({f^*_{\mathrm{M}}}^{-1}(a))$. □

### 3.3.2. Finding the BLDM Target Distributions

In this section we focus on minimizing the divergence (3.21) for a fixed mapping $f_{\mathrm{M}}$ with respect to the bit-level PMFs. Even in this case, the problem is non-convex: the constraints on the target probabilities in (3.21) are convex, but the divergence (3.21) is non-convex in general, see Example 3.1. Consequently, we resort to heuristic optimization to find the bit-level target distributions.

#### Grid Search

Gird search is a basic algorithm that can be used to optimize non-convex functions. It evaluates the function for certain parameters (usually a Cartesian product of the one-dimensional search sets for each parameter) and returns the parameters which achieve the lowest value of the function. The complexity of the grid search is strongly connected with the accuracy (denser search grid increases complexity and accuracy) and grows exponentially with the number of parameters. The condition (3.30) on the bit-levels' PMFs allows to reduce the search space.

#### Divergence Relaxation

The approach taken in [42] replaces the original objective function with a new function for which we can find an analytical solution. Namely, we invert the order of the arguments in the divergence. That is, instead of solving

$$\min_{P_{B_1}(0),\dots,P_{B_L}(0)} \mathbb{D}\left(\prod_{l=1}^{L} P_{B_l} \middle\| P_A\right) \tag{3.45}$$

we solve

$$\min_{P_{B_1}(0),\dots,P_{B_L}(0)} \mathbb{D}\left(P_A \middle\| \prod_{l=1}^{L} P_{B_l}\right). \tag{3.46}$$

The problems are different (divergence is not symmetric) and achieve the minima for different arguments in general. However, if we know that the minimum for any of the

problems is zero, the minimizer is the same for both problems. Thus, this approach finds the optimum bit-level distributions when the distribution $P_A$ can be represented as a product of binary distributions with the selected mapping $f_{\mathrm{M}}$.

This is an instance of a more general statement. Suppose the minimum (3.45) is $\epsilon_1$ and is achieved for a distribution $P^{*1}_{\boldsymbol{B}} = \prod_{l=1}^{L} P^{*1}_{B_l}$, and the minimum (3.46) is $\epsilon_2$ and is achieved for a distribution $P^{*2}_{\boldsymbol{B}} = \prod_{l=1}^{L} P^{*2}_{B_l}$. Using Pinsker's inequality [47] and the triangle inequality, we have

$$\|P^{*1}_{\boldsymbol{B}} - P_A\| \leq \sqrt{2\epsilon_1} \tag{3.47}$$

$$\|P^{*1}_{\boldsymbol{B}} - P_A\| \leq \sqrt{2\epsilon_2} \tag{3.48}$$

$$\|P^{*2}_{\boldsymbol{B}} - P^{*1}_{\boldsymbol{B}}\| \leq \sqrt{2\epsilon_1} + \sqrt{2\epsilon_2}. \tag{3.49}$$

Thus, the solution of (3.45) is close to the solution of (3.46) if both minima are close to zero. The inverted divergence in (3.46) can be written

$$\mathbb{D}\left(P_A || \prod_{l=1}^{L} P_{B_l}\right) = -\mathbb{H}(P_A) - \sum_{a\in\mathcal{A}} P_A(a) \log_2 \left(\prod_{l=1}^{L} P_{B_l}\Big([f_{\mathrm{M}}^{-1}(a)]_l\Big)\right) \tag{3.50}$$

$$= -\mathbb{H}(P_A) - \sum_{l=1}^{L} \sum_{a\in\mathcal{A}} P_A(a) \log_2 P_{B_l}\Big([f_{\mathrm{M}}^{-1}(a)]_l\Big) \tag{3.51}$$

$$= -\mathbb{H}(P_A) - \sum_{l=1}^{L} \sum_{b\in\{0,1\}} \sum_{\substack{a\in\mathcal{A} \\ [f_{\mathrm{M}}^{-1}(a)]_l = b}} P_A(a) \log_2 P_{B_l}(b) \tag{3.52}$$

$$= -\mathbb{H}(P_A) - \sum_{l=1}^{L} \sum_{b\in\{0,1\}} \Pr\Big[[f_{\mathrm{M}}^{-1}(A)]_l = b\Big] \log_2 P_{B_l}(b) \tag{3.53}$$

$$= -\mathbb{H}(P_A) + \sum_{l=1}^{L} \mathbb{E}_{B'_l \sim [f_{\mathrm{M}}^{-1}(A)]_l} \left[-\log_2 P_{B_l}(B'_l)\right]. \tag{3.54}$$

The expectation terms are cross-entropies between the bit-level PMFs $P_{B'_l}$, $l = 1, \ldots, L$, induced by the PMF $P_A$ via the mapping $f_{\mathrm{M}}$ and the optimization bit-level PMFs $P_{B_l}$, $l = 1, \ldots, L$. The cross entropy between two distributions $Q_X, P_X$ is

$$\mathbb{E}_{X\sim Q_X}\left[-\log_2 P_X(X)\right] = \mathbb{H}(Q_X) + \mathbb{D}\left(Q_X || P_X\right) \tag{3.55}$$

which gives

$$\mathbb{D}\left(P_A||\prod_{l=1}^{L}P_{B_l}\right) = -\mathbb{H}(P_A) + \sum_{l=1}^{L}\left\{\mathbb{H}(P_{B_l'}) + \mathbb{D}\left(P_{B_l'}||P_{B_l}\right)\right\}. \tag{3.56}$$

Only the latter divergence terms depend on the choice of the optimization PMFs $P_{B_l},\ l = 1,\ldots,L$, and can be minimized by choosing

$$P_{B_l}(b) = P_{B_l'}(b) = \sum_{\substack{a\in\mathcal{A}\\ [f_{\mathrm{M}}(a)]_l = b}} P_A(a) \qquad b\in\{0,1\},\ l = 1,\ldots,L. \tag{3.57}$$

Thus, $P_{B_l}$ is a marginal distribution of the $l$-th bit-level induced by $P_A$ and $f_{\mathrm{M}}$.

### Coordinate Descent

Recall the divergence in (3.21):

$$\mathbb{D}\left(P_{\boldsymbol{B}}||P_A\right) = \sum_{l=1}^{L} -H_2(P_{B_l}(0)) + \mathbb{E}_{\boldsymbol{B}\sim P_{\boldsymbol{B}}}\left[-\log_2 P_A\left(f_{\mathrm{M}}(\boldsymbol{B})\right)\right]. \tag{3.58}$$

Now consider a partial derivative with respect to the $i$-th parameter:

$$\begin{aligned}\frac{\partial\mathbb{D}}{\partial P_{B_i}(0)} = \log_2\frac{P_{B_i}(0)}{1-P_{B_i}(0)} &+ \sum_{\substack{a\in\mathcal{A}\\ \left[f_{\mathrm{M}}^{-1}(a)\right]_i = 0}}\ \prod_{\substack{l=1\\ l\neq i}}^{L} P_{B_l}\left(\left[f_{\mathrm{M}}^{-1}(a)\right]_l\right)\log_2\frac{1}{P_A(a)}\\ &- \sum_{\substack{a\in\mathcal{A}\\ \left[f_{\mathrm{M}}^{-1}(a)\right]_i = 1}}\ \prod_{\substack{l=1\\ l\neq i}}^{L} P_{B_l}\left(\left[f_{\mathrm{M}}^{-1}(a)\right]_l\right)\log_2\frac{1}{P_A(a)}\end{aligned}$$

and observe that only the first term depends on $P_{B_i}(0)$. We conclude that a solution to

$$\frac{\partial\mathbb{D}}{\partial P_{B_i}(0)} = 0 \tag{3.59}$$

is a global minimum of the divergence (with respect to $P_{B_i}(0)$) since the partial derivative is negative for $P_{B_i}(0)$ smaller than the solution, and positive for $P_{B_i}(0)$ greater than the solution. Thus, the $P_{B_i}^*(0)$ minimizing the divergence (the solution to (3.59)) is

$$P_{B_i}^*(0) = \frac{1}{1+2^{-\alpha}} \tag{3.60}$$

with

$$\alpha = \sum_{\substack{a\in\mathcal{A} \\ \left[f_{\mathrm{M}}^{-1}(a)\right]_i=1}} \prod_{\substack{l=1 \\ l\neq i}}^{L} P_{B_l}\left(\left[f_{\mathrm{M}}^{-1}(a)\right]_l\right) \log_2 \frac{1}{P_A(a)} - \sum_{\substack{a\in\mathcal{A} \\ \left[f_{\mathrm{M}}^{-1}(a)\right]_i=0}} \prod_{\substack{l=1 \\ l\neq i}}^{L} P_{B_l}\left(\left[f_{\mathrm{M}}^{-1}(a)\right]_l\right) \log_2 \frac{1}{P_A(a)}. \tag{3.61}$$

Equations (3.60) and (3.61) allow to formulate an iterative optimization procedure called coordinate descent where we update only one coordinate (parameter) at a time. At each step, we update successively all bit-level PMFs according to (3.60) and (3.61). Repeat the steps until convergence. The algorithm converges because during each update we decrease the objective function and the objective function (divergence) is non-negative. The algorithm coverages to a local minimum in general.

### Application Example: One-bit BLDM

Using the approach of Section 3.3.1 and coordinate descent optimization, one may find good parameters for a BLDM for the so-called one-bit shaping schemes [48]. There only one of the bit-level enters a DM and the other bit-levels remain unchanged, see Figure 3.2. One-bit shaping can achieve a significant shaping gain with a small complexity increase. The condition (3.30) becomes

$$P_{B_1}(0) \geq 0.5 = P_{B_2}(0) = \ldots = P_{B_L}(0)$$

and there is effectively only one probabilistically ordered mapping $f_{\mathrm{M}}$, i.e., the NBC mapping (the mapping $f_{\mathrm{M},1}$ in Table 3.2). It remains to find the probability distribution of the first bit-level, which can be solved by the coordinate descent. We have

$$P_{B_1}^{*}(0) = \frac{1}{1+2^{-\alpha}} \tag{3.62}$$

where $\alpha$ is the average logarithm of probability ratio of the symbols with the first bit 0 and the symbols with the first bit 1 (average a-priori information of the first bit):

$$\alpha = \frac{1}{2^{L-1}} \sum_{\boldsymbol{b}\in\{0,1\}^{L-1}} \log_2 \frac{P_A\left(f_{\mathrm{M}}\left([0,\boldsymbol{b}]\right)\right)}{P_A\left(f_{\mathrm{M}}\left([1,\boldsymbol{b}]\right)\right)}. \tag{3.63}$$

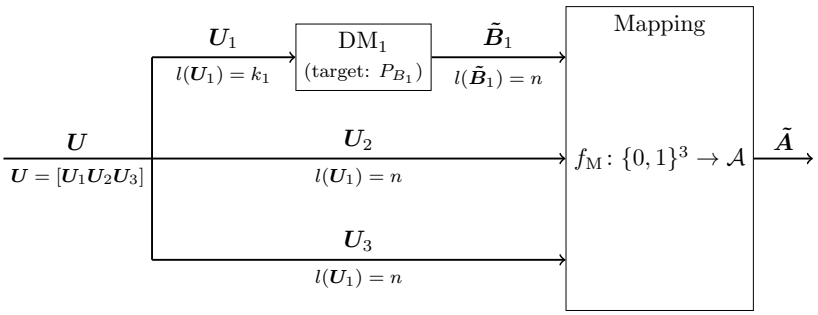



Figure 3.2.: Bit-level DM for one-bit shaping for the alphabet $\mathcal{A}$ of size 8.

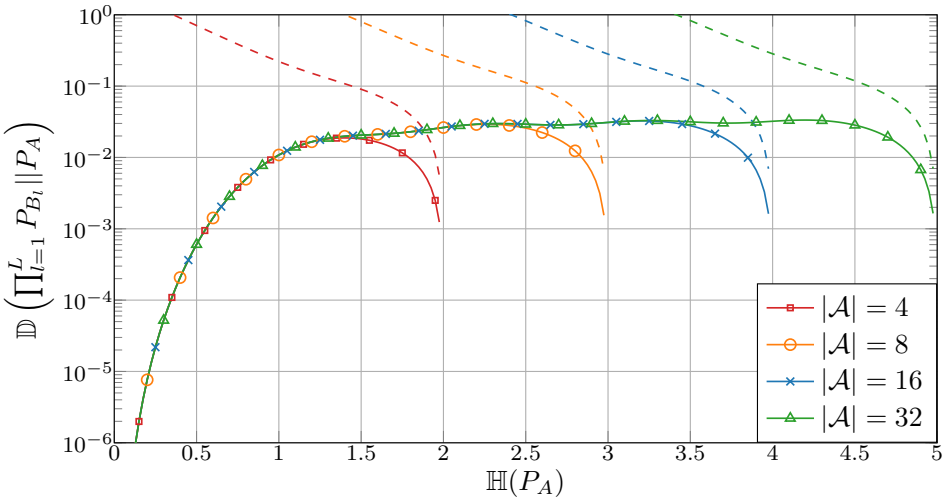


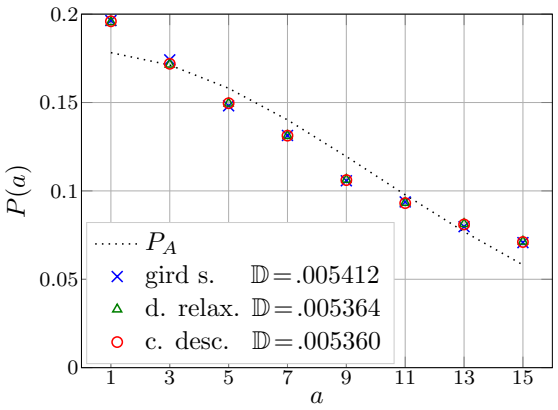

Figure 3.3.: The minimal achievable normalized divergence (3.7) of the BLDM (solid lines) and the one-bit BLDM (dashed lines) for a half Maxwell-Boltzmann target PMF, i.e., $P_A(a) \propto e^{-va^2}$, $a \in \mathcal{A} = \{1, 3, 5, \ldots, 2^{L+1} - 1\}$. The entropy of the distribution is varied by changing the parameter $v$.

The minimal achievable divergence for the one-bit BLDM is presented in Figure 3.3.

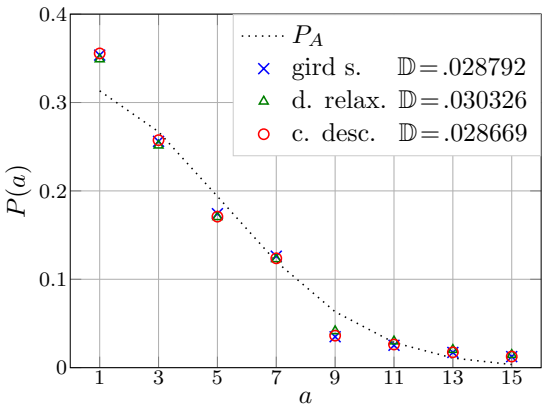
## 3.4. Optimization Results

We apply now the optimization approach from the previous sections to find a BLDM that achieves the minimal divergence (3.7). The results for different target PMFs and alphabet sizes are presented in Figures 3.3 and 3.4. To plot the figures, we performed an exhaustive search over probabilistically ordered mappings for bit-levels satisfying (3.30), and for each mapping we used the the optimization algorithms from the previous section.

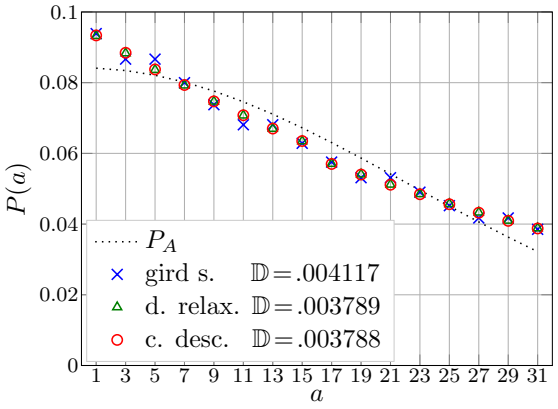
All algorithms achieve similar results. Interestingly, coordinate descent performed the



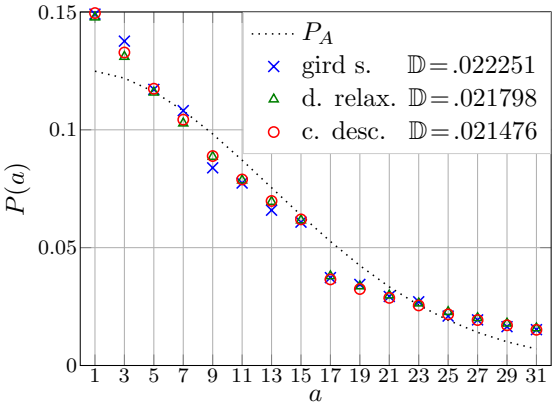

(a) $P_A(a) \propto e^{-0.005a^2}$



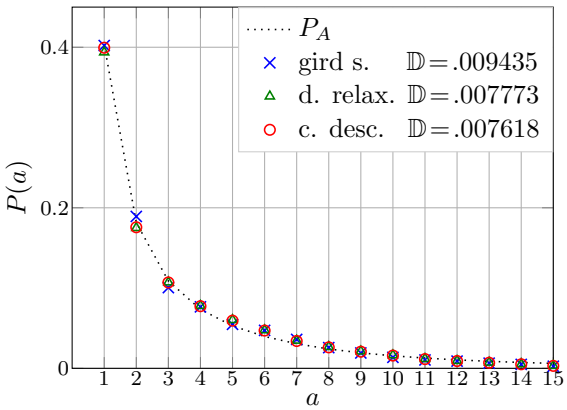

(b) $P_A(a) \propto e^{-0.02a^2}$



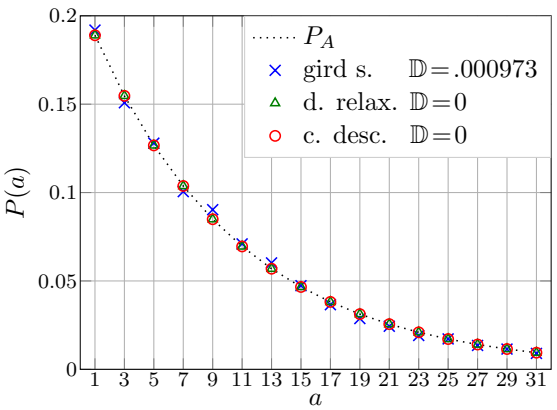

(c) $P_A(a) \propto e^{-0.001a^2}$




(d) $P_A(a) \propto e^{-0.003a^2}$




(e) $P_A(a) \propto \frac{1}{a}0.9^a$




(f) $P_A(a) \propto e^{-0.1a}$

Figure 3.4.: Empirical distributions (and corresponding divergences in the legend) of a single symbol generated by a BLDM for different target distributions $P_A$.

best in all scenarios. Coordinate descent converges to a local minimum, which suggests that the local minimum for the tested distributions is the global minimum. To check this, a grid search returned distributions which are very close to the distributions found by coordinate descent. The performance of the grid search is limited by granularity. Coordinate descent used 100 iterations, and is much faster than the grid search which evaluated the divergence at about $10^6$ points for each mapping.

Figure 3.3 presents the divergence obtained via optimization for a range of Maxwell-Boltzmann target distributions with different entropies. For low entropy, the target distribution effectively becomes binary and the BLDM can generate the distribution exactly. For intermediate entropy values, the divergence stays around 0.02 and 0.05 while slowly increasing. The increase is because the number of free parameters for the product distribution $P_{\boldsymbol{B}} = \prod_{l=1}^{L} P_{B_l}$ grows linearly with the number $L$ of the bit-levels, while the number of free parameters for an arbitrary distribution $P_A$ grows exponentially. For high entropy values, i.e., values close to the maximum entropy for a given alphabet size, the divergence decreases. This is expected, since in this case the Maxwell-Boltzmann distribution approaches the uniform distribution, which can be factored into a product of binary distributions, i.e., would have zero divergence. After testing different alphabet sizes and distribution parameters, we conjecture that the smallest divergence for Maxwell-Boltzmann distributions can be obtained with a natural binary code (NBC) mapping, see the mapping $f_{\mathrm{M},1}$ in Table 3.2. Theorem 3.3 states necessary and sufficient conditions for the NBC mapping to be a probabilistically ordered mapping. The NBC mapping was also used for the BLDM in [42–44]. Finally, the dashed lines in the plot in Figure 3.3 represent the divergences for the one-bit BLDM. For low-entropy, the target distribution $P_A$ accumulates most of the probability mass on the symbol 1, and the divergence goes to infinity since the one-bit BLDM has to put non-zero probability mass on at least half of the alphabet. For high entropy values for a given alphabet size, i.e., $\log_2 |\mathcal{A}| - 0.5 \leq \mathbb{H}(P_A) \leq \log_2 |\mathcal{A}|$, one-bit BLDM performs well.

Figure 3.4 presents the distributions obtained for different target PMFs. For better distinction, the target distribution $P_A$ is plotted with a dotted line (although it is discrete). The obtained divergence is low and suggest that BLDM can successfully improve the throughput or complexity of distribution matching. All figures except Figure 3.4(e) were obtained using the NBC mapping. Finally, Figure 3.4(f) was generated with an exponential target distribution that can be represented as a product distribution, and thus zero divergence was obtained by the relaxed divergence and the coordinate descent optimization algorithms.

**Theorem 3.3: Probabilistically Ordered NBC Mapping**
Consider $L$ independent bits $B_1, \ldots, B_L$ with corresponding probabilities $P_{B_1}(0), \ldots, P_{B_L}(0)$, and the joint PMF $P_{\boldsymbol{B}} = \prod_{l=1}^{L} P_{B_l}$. The bit labels $\boldsymbol{b}_i,\ i = 1, \ldots, 2^L$, ordered according to NBC with the most significant bit first, i.e., $[\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_{2^L}] = [0\ldots00, 0\ldots01, \ldots, 1\ldots11]$. These vectors have non-increasing probabilities, i.e., $P_{\boldsymbol{B}}(\boldsymbol{b}_i) \geq P_{\boldsymbol{B}}(\boldsymbol{b}_{i+1})$ if and only if

$$\forall l \in \{1, \ldots, L\} \quad \mathrm{LLR}(B_l) \geq \sum_{i=l+1}^{L} \mathrm{LLR}(B_i) \tag{3.64}$$

where $\mathrm{LLR}(B_l) := \log_2\left(\frac{P_{B_l}(0)}{P_{B_l}(1)}\right)$ is the log-likelihood ratio (LLR) of the bit $B_l$.

*Proof.* See Appendix B. □

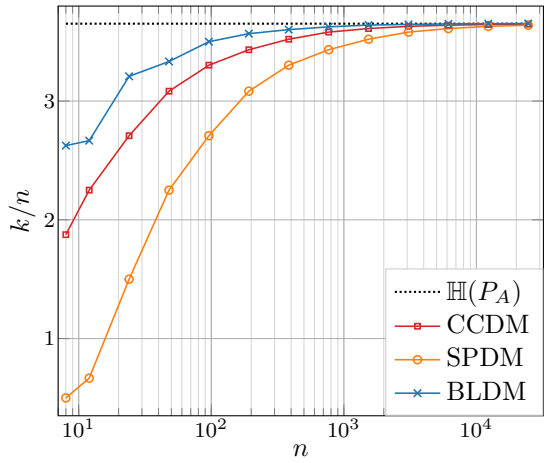
## 3.4.1. Normalized Divergence

We now compare a symbol-level DM with the parallelized architectures BLDM and SPDM in Figure 3.1 which use the constituting DMs of the same type as the symbol-level DM. We consider two DM implementations: the CCDM and the enumerative approach from [45] that we refer to as Enumerative Distribution Matcher (EDM). The EDM is a weight-constrained DM similar to the MCDM from Section 2.4.1, but it has lower memory requirements for large alphabets. The alphabet size is 16, which corresponds to $L = 4$ bit-levels and $L = 4$ constituting DMs in the BLDM. We use four constituting DMs for the SPDM as well.

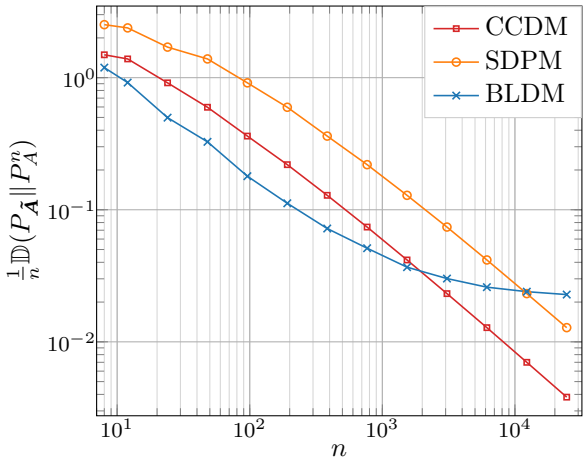
### CCDM-based Architectures

Figure 3.5 presents the results for the symbol-level CCDM (denoted by CCDM) and the parallelized architectures. Figure 3.5(a) shows that the BLDM has higher matching rate than the symbol-level CCDM. The effect is particularity pronounced for shorter output sequences. The behavior is because the bit-level constant-composition constraints are more relaxed that the symbol-level constant-composition constraint, see Section 3.2.1. We also observe that the rate of the SPDM corresponds to the rate of the symbol-level CCDM with length $\frac{n}{L}$. This follows in a similar way as the divergence result from (3.9).

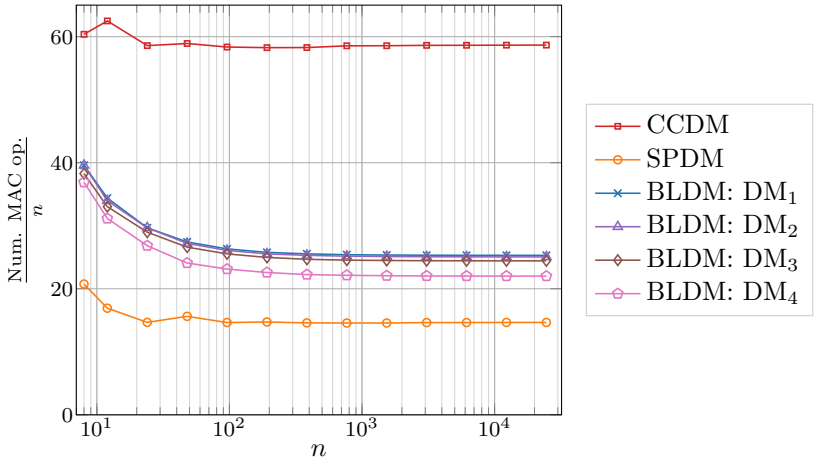
Figure 3.5(b) presents the normalized divergence of the DMs. For output lengths below 2000, the BLDM achieves lower normalized divergence than the symbol-level CCDM.




(a) Matching rate




(b) Normalized divergence




(c) Encoder operations per symbol

Figure 3.5.: Results for the CCDM and the parallelized CCDM-based DMs for a half Maxwell-Boltzmann target distribution $P_A$ from Figure 3.4(d), i.e., $P_A(a) \propto e^{-0.003a^2}$, $a \in \mathcal{A} = \{1, 3, 5, \ldots, 31\}$.

This is the length region where the performance of the CCDM is dominated by the rate-loss and the BLDM achieves lower rate-loss due to a larger implementation codebook. For larger values of $n$, the normalized divergence of the BLDM approaches the lower-bound (3.7) which equals $\approx 0.0215$ for this target $P_A$ (see Figure 3.4(d)). Finally, the divergence of the SPDM corresponds to the divergence of the symbol-level CCDM with length $\frac{n}{L}$, as expected from (3.9).

Figure 3.5(c) presents the number of multiply–accumulate (MAC) operations per output symbol for each of the DMs. A MAC operation is defined as

$$a \leftarrow a + b \times c \tag{3.65}$$

and is a key figure of merit of computational complexity for a digital signal processor (DSP). The ACDM implementation described in Section 2.2 performs three different types of integer operations: multiplication, addition, and comparison. For simplicity we consider all these operations as MAC operations and count the total number of performed operations during encoding (or decoding). The number of MAC operations at the encoder and decoder are similar. Finally, we average the number of operations for encoding $10^5$ random input sequences and plot the results in Figure 3.5(c).

First, consider the symbol-level CCDM and the SPDM. For the SPDM we present the number of MAC operations for a single constituting DM normalized by the total output length $n$. Thus, the number of MAC operations per output symbol is $L$ times lower for the SPDM. This is because each constituting DM generates only $\frac{n}{L}$ symbols at the output. Observe in Algorithms 2.1 and 2.2 that the number of operations increases linearly with the length of the output sequence (for each extra symbol the algorithm searches for the appropriate interval). Thus, when a separate DSP core can be employed for each of the constituting DMs, the SPDM throughput will be approximately $L$ times higher than for the symbol-level CCDM.

Next, consider the symbol-level CCDM and the BLDM. The output length of the constituting DMs is the same as for the symbol-level CCDM, so the reduction in the number of MAC operation is not caused by the shorter output sequence as for the SPDM. For the BLDM, the constituting DMs operate on binary alphabets which means that only one comparison has to be performed to find the next interval, see Algorithms 2.1 and 2.2. For the symbol-level CCDM, in the worst case we need $2^L - 1$ comparisons to find the next interval. Naturally, the bit-levels that have higher entropy (more uniform distribution) perform more operations. For instance, a constituting DM with the output composition $[1, n-1]$ can generate the zeros suffix of the sequence once the 1-bit was

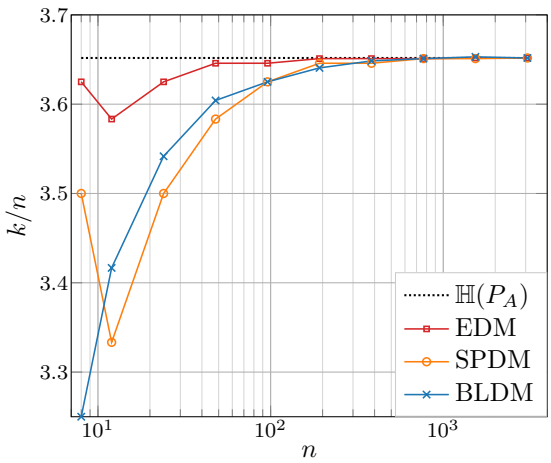
generated.

### Architectures Based on Weight-Constrained DM

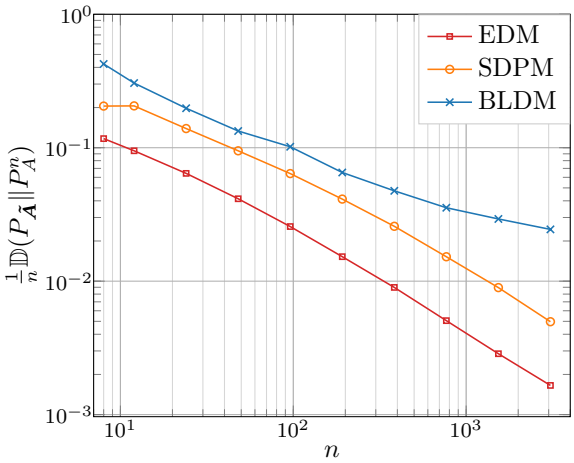
Figure 3.6 presents the results for the symbol-level EDM [45] (denoted by EDM) and the parallelized architectures. Figure 3.6(a) shows that the BLDM has lower matching rate than the symbol-level EDM. The effect is particularity pronounced for short lengths. The behavior is because the bit-level weight constraints are more strict that the symbol-level weight constraint, see Section 3.2.2. We also observe that the rate of the SPDM corresponds to the rate of the symbol-level EDM with length $\frac{n}{L}$. This follows in a similar way as the divergence result from (3.9). Finally, we note that the matching rates are much higher than the rates obtained for the CCDM-based architectures.

Figure 3.6(b) presents the normalized divergence of the DMs. The BLDM achieves higher normalized divergence than the symbol-level EDM. This is caused partially by the lower matching rate of the BLDM. However, the BLDM with the constituting EDMs achieves better divergence than the BLDM with the constituting CCDMs in the previous section. For larger values of $n$, the normalized divergence of the BLDM approaches the lower-bound (3.7) which is $\approx 0.0215$ for this target $P_A$ (see Figure 3.4(d)). Finally, the divergence of the SPDM corresponds to the divergence of the symbol-level EDM with length $\frac{n}{L}$, as expected from (3.9).

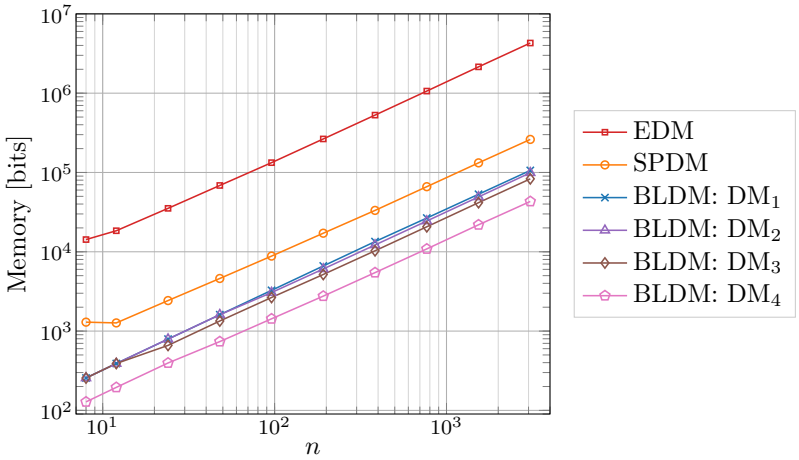
The EDM needs to store a trellis of size $nW_0(w_m + w_p)$, where $W_0$ is the maximum output sequence weight, and $w_n, w_p$ are precision parameters that we set to $w_m = w_p = 30$ bits. Figure 3.6(c) presents the memory storage requirements for each of the DMs. Naturally, the constituting DMs in the SPDM can use the same trellis since they are identical. For the BLDM, only the largest trellis needs to be stored. The constituting DMs for each bit-level can simply start processing the trellis from different starting nodes to generate sequences from different model codebooks. We observe that the storage requirements for the BLDM are indeed lower than for the SPDM, however the normalized divergence is worse as well. Thus, one could use the SPDM with higher parallelization factors to reduce the memory requirements even further.




(a) Matching rate




(b) Normalized divergence




(c) Memory storage requirements

Figure 3.6.: Results for the EDM and the parallelized EDM-based DMs for a half Maxwell-Boltzmann target distribution $P_A$ from Figure 3.4(d), i.e., $P_A(a) \propto e^{-0.003a^2}$, $a \in \mathcal{A} = \{1, 3, 5, \ldots, 31\}$

# 4

# Distribution Matching and Efficient Communication

## 4.1. Probabilistic Amplitude Shaping for AWGN Channel

In this section we evaluate the performance of a PAS system with the CCDM and MCDM. As observed in Chapter 2, the MCDM achieves lower divergence and higher matching rate than CCDM and this should contribute to a better PAS system. A lower divergence allows to operate closer to the channel capacity, see [19, Eq. 5]. A higher matching rate allows the MCDM to use a "more shaped" distribution. The more informative prior distribution helps the channel decoder and should result in a lower frame error rate (FER).

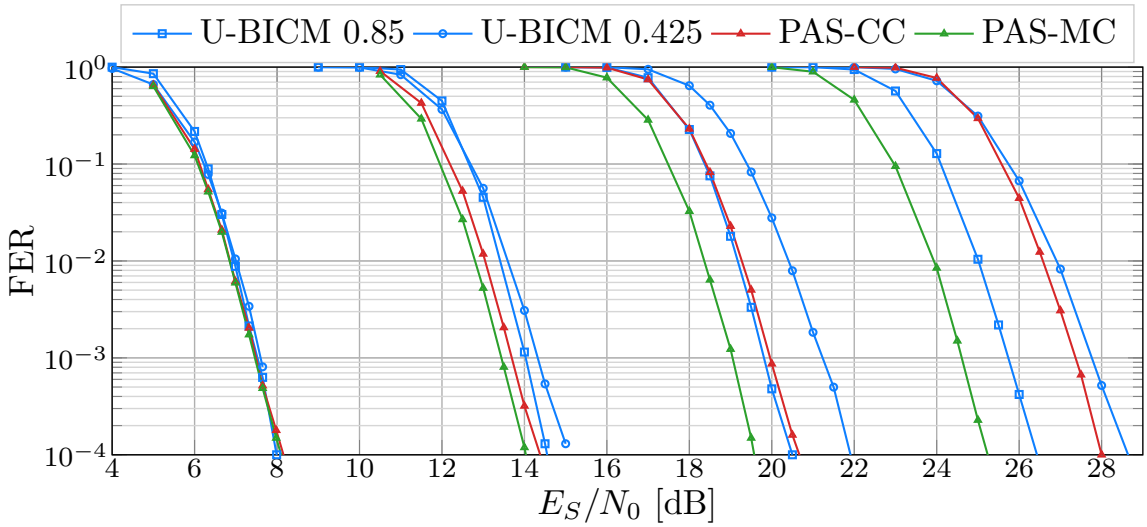
Figure 4.1 presents FERs for classic BICM with uniform transmit distribution (U-BICM), PAS with CCDM (PAS-CC), and PAS with MCDM (PAS-MC). The curve groups correspond to transmission rates of 1.70, 3.40, 5.10, 6.80 bit per channel use (b/CU), respectively. The effective coding rate of a PAS scheme is a product of the rate of the channel code (equal to 0.85) and the matching rate of a DM (equal to 0.5). Thus, the effective coding rate for the PAS schemes is 0.425. We compare the PAS schemes with U-BICM using exactly the same channel code and lower order modulation (denoted by U-BICM 0.85), and U-BICM using the same constellation and the same effective

coding rate (denoted by U-BICM 0.425). We use the two U-BICM baselines since the former U-BICM scheme has longer codewords (in terms of symbols), which may affect the performance, and the latter U-BICM scheme uses a different LDPC code, which also may have different performance. Accordingly, all schemes have the same transmission rate and use 100 iterations of belief propagation at the decoder. The PAS schemes and the U-BICM scheme with the code rate 0.425 use 16-, 64-, 256-, and 1024-QAM constellations for the transmission rates of 1.70, 3.40, 5.10, 6.80 b/CU respectively. The U-BICM scheme with the code rate 0.85 uses the QPSK, 16-, 64-, 256-QAM constellations for the transmission rates of 1.70, 3.40, 5.10, 6.80 b/CU respectively.

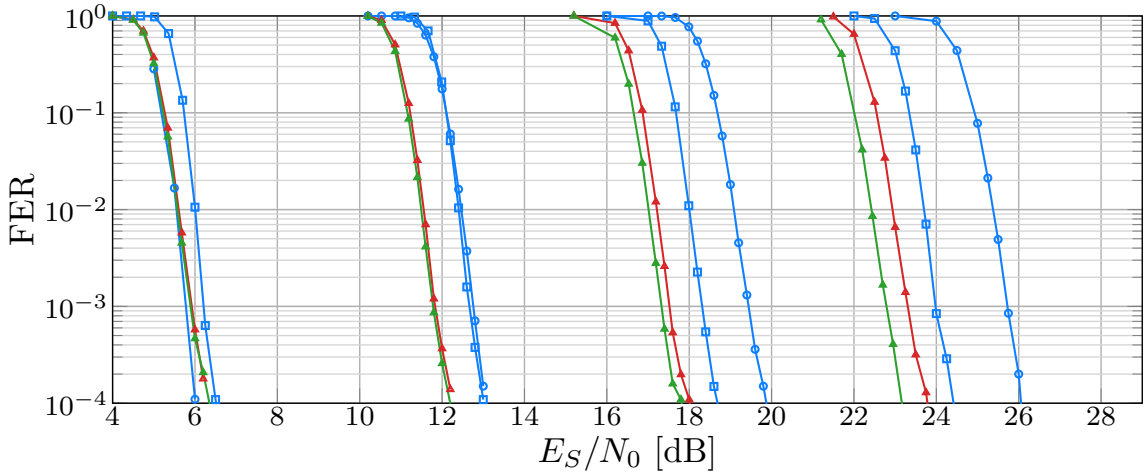
Figure 4.1a presents the FER curves for a codeword of 360 bits. The PAS-MC scheme performs better than the PAS-CC scheme for all transmission rates. E.g., at the FER of $10^{-3}$, the advantage of PAS-MC is 0.01, 0.27, 0.91, and 2.75 dB for transmission rates of 1.70, 3.40, 5.10, 6.80 b/CU respectively. The gains are greater for short codewords since the gap in matching rate between the MCDM and CCDM is greater for short output sequences, e.g., see Figure 2.6. The gap in normalized divergence seems to offer a small advantage according to [19, Eq. 5], e.g., reducing the divergence by $10^{-2}$ reduces the gap to the capacity by $10^{-2}$ bits per codeword. The PAS-CC performs worse than the U-BICM baseline for 256-, and 1024-QAM. This is due to increasing rate-loss of the CCDM for larger alphabets. However, PAS-MC performs better than the U-BICM baselines for all transmission rates. This demonstrates the necessity of using efficient distribution matching for short transmission frames.

Figure 4.1b presents the FER curves for a codeword of 2400 bits. The PAS-MC scheme performs better than the PAS-CC scheme for all transmission rates. E.g., at the FER of $10^{-3}$, the advantage of PAS-MC is 0.015, 0.045, 0.195, and 0.5 dB for transmission rates of 1.70, 3.40, 5.10, 6.80 b/CU respectively. The gains are lower than for the 360-bits codeword. Both PAS schemes beat the U-BICM baselines.

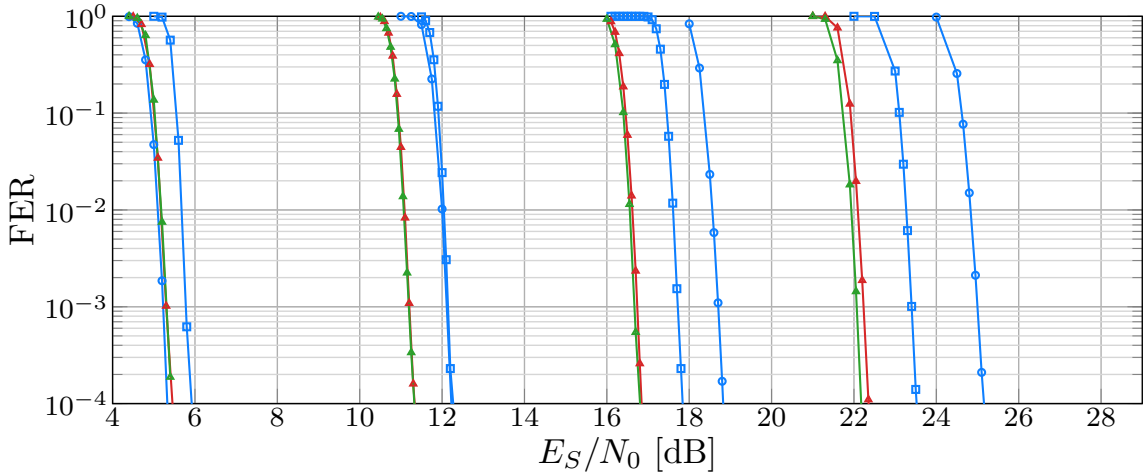
Figure 4.1b presents the FER curves for a codeword of 9600 bits. Due to increasing complexity of MCDM, we implemented the PAS-MC scheme with an enumerative DM from [45]. The PAS-MC scheme performs better than the PAS-CC scheme for all transmission rates. E.g., at the FER of $10^{-3}$, the advantage of PAS-MC is 0.005, 0.01, 0.06, and 0.16 dB for transmission rates of 1.70, 3.40, 5.10, 6.80 b/CU respectively. The advantage is rather small and comes with increased complexity when compared to the PAS-CC scheme. Both PAS schemes beat the U-BICM baselines.




(a) Codeword length = 360 bits




(b) Codeword length = 2400 bits




(c) Codeword length = 9600 bits

Figure 4.1.: Frame error rates for transmission rates of 1.70, 3.40, 5.10, 6.80 bits per channel use, respectively.

## 4.2. Efficient Communication for Rayleigh Block Fading Channel

This section is based on [49] and considers communication schemes for a Rayleigh Block Fading Channel (RBFC). The fading experienced by consecutive symbols of wireless links is usually correlated and the channel can be predicted to some extent. The RBFC models predictability via constant fading within a block of $T$ symbols, and channel fluctuations via fading that changes independently from block to block [50]. We consider the RBFC where the fading coefficients are unknown to the transmitter and the receiver, i.e., the channel is non-coherent and the transmitter and the receiver have no channel state information (CSI).

Consider the Single Input-Single Output (SISO) RBFC with $T \geq 2$. The capacity-achieving coding scheme applies discrete per-block power modulation and, conditioned on the block power, transmits isotropically distributed vectors [50]. Various other signaling strategies have been studied in the literature. Pilot schemes, where the receiver estimates the channel from training symbols, were examined in [51]. *Unitary Space-Time Modulation* (USTM), proposed in [52], follows the information-theoretic guidelines from [50] but does not use per-block power modulation. Finally, Gaussian IID input signals, studied in [53], are optimal when the receiver knows the channel, and perform well for large $T$ in general.

### Channel Model

A SISO RBFC has fading that remains the same for a block of $T$ consecutive symbols and that changes between blocks in an independent manner. The parameter $T$ represents the channel coherence time. The channel fading coefficient $H$ is drawn independently for each block from a circular-symmetric zero-mean unit-variance complex Gaussian distribution, i.e., $H \sim \mathcal{N}_{\mathbb{C}}(0, 1)$. The channel input-output relation is

$$\boldsymbol{Y} = H\sqrt{\gamma}\boldsymbol{X} + \boldsymbol{N} \tag{4.1}$$

where $\boldsymbol{X}$ and $\boldsymbol{Y}$ are vectors with $T$ transmitted and received symbols, respectively, and $\boldsymbol{N}$ is a vector of $T$ IID RVs $\sim \mathcal{N}_{\mathbb{C}}(0, N_0)$. We fix $N_0 = 1$ and normalize the signal via

$$\mathbb{E}\left[\|\boldsymbol{X}\|^2\right] = T \tag{4.2}$$

so that $\gamma$ represents the average signal-to-noise ratio (SNR) for a single received symbol.

### Input Signal

*1) Capacity-Achieving Input Signal:* The input that achieves capacity over a non-coherent SISO RBFC can be written as

$$\boldsymbol{X} = \boldsymbol{\Phi} V \tag{4.3}$$

where $\boldsymbol{\Phi}$ is a complex vector of dimension $T$ and $V$ is a real non-negative scalar RV, with $\boldsymbol{\Phi}$ and $V$ being independent [50]. The distribution (4.3) is commonly referred to as a product form. The complex vector $\boldsymbol{\Phi}$ is equally likely to point in any direction in $T$-dimensional complex space, i.e., $\boldsymbol{\Phi}$ is uniformly distributed on a $T$-dimensional complex sphere with radius $\sqrt{T}$. The $V$ that achieves capacity is discrete [50]. Because $\boldsymbol{\Phi}$ is isotropical, each $X_i$, for $i = 1, \ldots, T$, has the same marginal distribution. We thus have

$$\mathbb{E}\left[\|\boldsymbol{X}\|^2\right] = \mathbb{E}\left[\sum_{i=1}^{T} |X_i|^2\right] = T\mathbb{E}\left[V^2\right]. \tag{4.4}$$

The power constraint (4.2) is met as long as $\mathbb{E}[V^2] = 1$.

*2) Discrete Product Form:* The $\boldsymbol{\Phi}$ in (4.3) is a continuous RV on the $T$-dimensional complex sphere. In practice, we must discretize $\boldsymbol{X}$ by discretizing $\boldsymbol{\Phi}$. To this end, we sample a $T$-dimensional complex sphere with radius $\sqrt{T}$ by choosing the following points

$$\boldsymbol{\Theta} = [\Theta_1, \ldots, \Theta_T] \tag{4.5}$$

where $\Theta_i,\ i = 1, \ldots, T$, are IID uniformly distributed RVs on the QPSK alphabet

$$\mathcal{A}_4 = \{1, -1, j, -j\}.$$

It is clear that $\boldsymbol{\Theta}$ lies on a $T$-dimensional complex sphere with radius $\sqrt{T}$ because $\boldsymbol{\Theta}\boldsymbol{\Theta}^\dagger = T$.

The next step is to choose $V$. We choose an on-off distribution so that the power constraint (4.2) is met:

$$V = \begin{cases} 0, & \text{with probability } 1 - P_1 \\ v_1 = \sqrt{\frac{1}{P_1}}, & \text{with probability } P_1 \end{cases} \tag{4.6}$$

where $P_1$ is a parameter that depends on the SNR. We can write the discrete input signal as

$$\boldsymbol{X} = \begin{cases} \mathbf{0}, & \text{with probability } 1 - P_1 \\ \sqrt{\frac{1}{P_1}}\boldsymbol{\Theta} & \text{with probability } P_1. \end{cases} \tag{4.7}$$

### Computing the Mutual Information

The methods for computing the mutual information for RBFCs presented in [50,53] rely on the rotational invariance of $\boldsymbol{X}$, i.e., $\boldsymbol{X} \sim \boldsymbol{X}\mathbb{U}$ for any unitary matrix $\mathbb{U}$. For such input signals closed-form expressions for the PDF of the channel output are available in [54,55], which allows to estimate the mutual information using MC techniques. However, a discrete signal is not rotationally invariant.

We present a nested MC approach tailored to estimate the mutual information over SISO RBFCs with $\boldsymbol{X}$ as in (4.7). Consider the identity

$$\mathbb{I}(\boldsymbol{Y};\boldsymbol{X}) = h\left(\boldsymbol{Y}\right) - h\left(\boldsymbol{Y}|\boldsymbol{X}\right) \tag{4.8}$$

where $h$ denotes differential entropy. From (4.1) observe that if $\boldsymbol{X} = \boldsymbol{x}$, then $\boldsymbol{Y}$ has a complex circular-symmetric Gaussian distribution with $\mathbf{0}$ mean and covariance matrix $\mathbb{I} + \gamma \boldsymbol{x}^{\dagger}\boldsymbol{x}$. Therefore, we have

$$\begin{aligned} h\left(\boldsymbol{Y}|\boldsymbol{X}\right) &= \mathbb{E}\left[\log\det\left(\pi e\left(\mathbb{I} + \gamma \boldsymbol{X}^{\dagger}\boldsymbol{X}\right)\right)\right] \\ &\overset{\text{(a)}}{=} \mathbb{E}\left[\log\left(1 + \gamma \boldsymbol{X}\boldsymbol{X}^{\dagger}\right)\right] + T\log\pi e \\ &\overset{\text{(b)}}{=} P_1 \log\left(1 + \gamma\frac{T}{P_1}\right) + T\log\pi e \end{aligned} \tag{4.9}$$

where (a) follows by Sylvester's determinant identity, i.e., $\det(\mathbb{I}_n + \boldsymbol{y}^{\dagger}\boldsymbol{x}) = \det(1 + \boldsymbol{x}\boldsymbol{y}^{\dagger})$, $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{C}^n$, and (b) follows by (4.7).

Next, we have

$$h\left(\boldsymbol{Y}\right) = -\mathbb{E}_{\boldsymbol{Y}}\left[\log p(\boldsymbol{Y})\right] \tag{4.10}$$

which can be evaluated by MC averaging with respect to $p(\boldsymbol{y})$. However, a direct computation of $p(\boldsymbol{y})$ as a marginal of $p(\boldsymbol{y}, \boldsymbol{x})$ is infeasible even for short block lengths as it involves summing over all possible input vectors, i.e., we have

$$p(\boldsymbol{y}) = \sum_{\boldsymbol{x}} P(\boldsymbol{x}) p(\boldsymbol{y}|\boldsymbol{x}) \tag{4.11}$$

and the number of $\boldsymbol{x}$ grows exponentially with $T$. On the other hand, $\boldsymbol{Y}$ conditioned on $V$ and $H$ is a vector of IID RVs, which can be seen from (4.1) and (4.7). In this case the summation (4.11) can be simplified.

First, we expand

$$\begin{aligned} p(\boldsymbol{y}) &= \Pr(V{=}0)p(\boldsymbol{y}|V{=}0) + \Pr(V{=}v_1)p(\boldsymbol{y}|V{=}v_1) \\ &\overset{\text{(a)}}{=} (1-P_1)\frac{1}{\pi^T}e^{-\|\boldsymbol{y}\|^2} + P_1 p(\boldsymbol{y}|V{=}v_1) \end{aligned} \tag{4.12}$$

where (a) follows by $(\boldsymbol{Y}|V{=}0) \sim \mathcal{N}_{\mathbb{C}}(\boldsymbol{0}, \mathbb{I})$. We next compute the term $p(\boldsymbol{y}|V{=}v_1)$. For convenience, we denote the condition $V{=}v_1$ by writing $v_1$ after the conditioning mark. By further conditioning on $H$, we have

$$\begin{aligned} p(\boldsymbol{y}|v_1) &= \mathbb{E}_H\left[p(\boldsymbol{y}|v_1, H)\right] \\ &= \mathbb{E}_H\left[\prod_{i=1}^{T} p(y_i|v_1, H)\right] \\ &= \mathbb{E}_H\left[\prod_{i=1}^{T} \sum_{\theta_i \in \mathcal{A}_4} P(\theta_i) p(y_i|\theta_i, v_1, H)\right] \\ &= \mathbb{E}_H\left[\prod_{i=1}^{T} \sum_{\theta_i \in \mathcal{A}_4} \frac{1}{4}\ \frac{1}{\pi} e^{-|y_i - H\sqrt{\gamma}\theta_i v_1|^2}\right]. \end{aligned} \tag{4.13}$$

We define the argument of the expectation in (4.13) as

$$f(\boldsymbol{y}, h) = \prod_{i=1}^{T} \sum_{\theta_i \in \mathcal{A}_4} \frac{1}{4\pi} e^{-|y_i - h\sqrt{\gamma}\theta_i v_1|^2}. \tag{4.14}$$

The complexity of computing (4.14) grows linearly with respect to $T$ and the alphabet size of the symbols $\Theta_i,\ i = 1, \ldots, T$. To evaluate the expectation in (4.13) we apply MC averaging.

Finally, using (4.10), (4.12) and (4.13) we formulate a nested MC routine for evaluating $h(\boldsymbol{Y})$, which is presented in Algorithm 4.1, with $f(\boldsymbol{y}, h)$ being the function (4.14). Using Algorithm 1, (4.9) and (4.8), we estimate the mutual information achieved by the proposed discrete signal.

Our approach works well for moderate block lengths, e.g., $T = 50$, whereas computation with (4.11) would be impossible (each sum would require $2^{100}$ additions). The

accuracy can be increased at the cost of computing time by increasing the number of samples $N_1$, $N_2$. For the results in the next section, the number of samples was chosen so that the relative error of the mutual information estimates lies within 3% with probability not smaller than 90%. The presented algorithm can be adapted for any discrete IID signal $\boldsymbol{X}$.

**Algorithm 4.1** Evaluation of $h(\boldsymbol{Y})$

**Input:** $N_1, N_2, T$

**Output:** An estimate of $h(\boldsymbol{Y})$

$\boldsymbol{s} \leftarrow \lambda$, $x(\lambda) \leftarrow 0$, $y(\lambda) \leftarrow 1$
**for** $i = 1$ **to** $N_1$ **do**
  generate $\boldsymbol{y}_i$ using (4.1)
  **for** $j = 1$ **to** $N_2$ **do**
    generate $h_j \sim \mathcal{N}_{\mathbb{C}}(0, 1)$
    $a_j \leftarrow f(\boldsymbol{y}_i, h_j)$
  **end for**
  $\hat{p}(\boldsymbol{y}_i|v_1) \leftarrow \frac{1}{N_2} \sum_{j=1}^{N_2} a_j$
  $\hat{p}(\boldsymbol{y}_i) \leftarrow (1 - P_1)\frac{1}{\pi^T} \exp(-\|\boldsymbol{y_i}\|^2) + P_1 \hat{p}(\boldsymbol{y}_i|v_1)$
  $b_i \leftarrow -\log(\hat{p}(\boldsymbol{y}_i))$
**end for**
**return** $\frac{1}{N_1} \sum_{i=1}^{N_1} b_i$

### Results

We compare our discrete product form with several other input distributions from the literature. The rates are presented versus the SNR per information bit, which is

$$\frac{E_b}{N_0} = \frac{\gamma}{\frac{1}{T} I(\boldsymbol{X}; \boldsymbol{Y})}.$$

*1) Perfect CSI Capacity:* Gaussian IID input signals achieve capacity when the receiver has CSI. In this case, the capacity is independent of the block length $T$ and is

$$C(\gamma) = \mathbb{E}_H \left[ \log(1 + \gamma |H|^2) \right]. \tag{4.15}$$

The perfect CSI capacity is an upper-bound to non-CSI rates.

*2) Gaussian IID Modulation:* Gaussian IID inputs do not achieve capacity when the receiver has no CSI. We present the rate estimates computed by the algorithm proposed in [53].

*3) On-Off Gaussian IID Modulation:* The schemes from Sections 4.2 and 4.2 below are on-off schemes. To make a fair comparison, we consider an on-off Gaussian signaling, i.e., we use

$$\boldsymbol{X} = \begin{cases} \mathbf{0}, & \text{with probability } 1 - Q_1 \\ \sqrt{\frac{1}{Q_1}}\boldsymbol{W} & \text{with probability } Q_1 \end{cases} \tag{4.16}$$

where $\boldsymbol{W} \sim \mathcal{N}_{\mathbb{C}}(\mathbf{0}, \mathbb{I})$ and $Q_1$ is chosen to maximize mutual information. A modified version of the algorithm from [53] is used to compute the rates.

*4)Unitary Space-Time Modulation:* Unitary Space-Time Modulation (USTM) as in [52] has

$$\boldsymbol{X} = \boldsymbol{\Phi} \tag{4.17}$$

where $\boldsymbol{\Phi}$ is an isotropically distributed complex vector of radius $\sqrt{T}$. The algorithm from [52] is used to compute the mutual information.

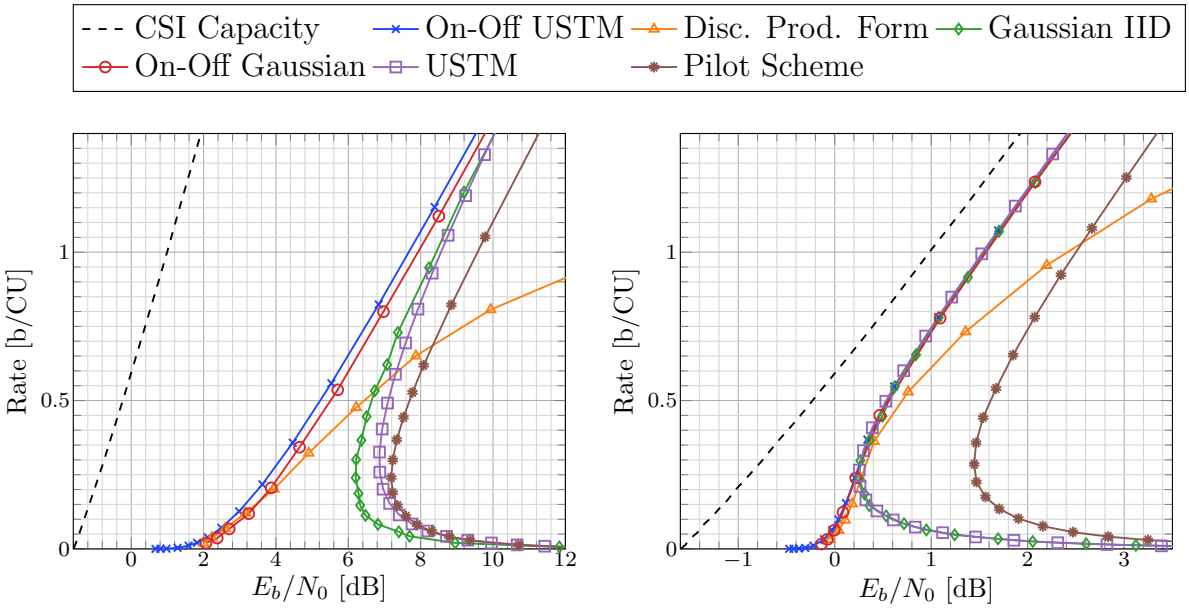
*5)On-Off Unitary Space-Time Modulation:* We use the product form (4.3) and we choose $V$ to be on-off distributed as in (4.6), which yields

$$\boldsymbol{X} = \begin{cases} \mathbf{0}, & \text{with probability } 1 - P_1 \\ \sqrt{\frac{1}{P_1}}\boldsymbol{\Phi} & \text{with probability } P_1 \end{cases} \tag{4.18}$$

where $\boldsymbol{\Phi}$ is an isotropically distributed complex vector of radius $\sqrt{T}$ and $P_1$ is chosen to maximize mutual information. This scheme can also be seen as an isotropical counterpart of the signal (4.7). The algorithm from [50] is used to compute the mutual information.

*6)Discrete Product Form:* We use the signal (4.7) and the $P_1$ obtained from the optimization for the on-off unitary space-time modulation. Although better $P_1$ may be found, this approach is much faster than performing the optimization with respect to mutual information estimate for the discrete scheme, i.e., using Algorithm 4.1 in the objective function. After fixing $P_1$, Algorithm 4.1 is used to estimate the mutual information.

*7)Pilot Scheme:* We consider a pilot based scheme, where pilot symbols are inserted during each fading block. At the receiver the pilot symbols are used to estimate the channel and the estimate is used to decode the data. The scheme is allowed to use *pilot-power-boosting*, i.e., pilot symbols can have different power than data symbols. For




(a) $T = 2$ (b) $T = 50$

Figure 4.2.: Information rates for block length $T = 2$ and $T = 50$.

the SISO RBFC, using one pilot symbol per fading block maximizes a lower-bound on the capacity [51]. The plotted rate, also presented in [51], is a lower-bound to the rate achieved by the best pilot scheme with separate recovery of pilots and data.

### Discussion

Figures 1 and 2 present results for $T = 2$ and $T = 50$ respectively. The gap between the CSI capacity and the presented schemes decreases with increasing block length, and the schemes that use isotropically distributed alphabets perform similarly at high SNR. The discrete product form achieves rates comparable to its isotropical counterpart up to 0.4 bits per channel use (b/CU) and saturates approximately at 1 b/CU and 2 b/CU, respectively.

*1)Using the* **0***-symbol:* USTM achieves the minimum $E_b/N_0$ at a rate of about 0.3 b/CU and below this rate becomes power inefficient. Introducing non-zero probability mass at the **0**-symbol improves power efficiency. A significant improvement of the on-off USTM over the standard USTM occurs for rates below 1.5 b/CU for $T = 2$ and for rates below 0.25 b/CU for $T = 50$. The same behavior follows for Gaussian IID and on-off Gaussian signaling. The importance of using the **0**-symbol is known in the low-SNR

regime [56, 57]. The effect is relevant also for higher SNR for small $T$.

*2)On-a-sphere signal:* For $T = 2$ a gap between the on-off Gaussian modulation and the on-off USTM can be observed. Both signals are on-off schemes so the gain is due to the on-a-sphere structure of the on-off USTM during the on-cycle. For rates below 0.15 b/CU, the discrete product form, which uses a sampled sphere signal during the on-cycle also performs better than the on-off Gaussian signaling.

*3)Power efficiency:* The presented plots for the on-off USTM and discrete product form coincide in the wideband regime. We are interested whether the schemes are first order optimal, i.e., whether they can achieve the highest power efficiency achieved by Gaussian inputs with CSI, which is given by $(E_b/N_0)_{\min} = -1.59$ dB. The scheme from Section 4.2 is a peaky *flash signaling* (if $P_1 \to 0$ as $\gamma \to 0$), i.e., the mixture of a probability distribution concentrating at $\mathbf{0}$ and a probability distribution that migrates to infinity as SNR vanishes [56].

In [56, Thm 5] it is shown that flash signaling achieves $(E_b/N_0)_{\min} = -1.59$ dB over the RBFC. The scheme from Section 4.2 is optimized with respect to $P_1$, so it also achieves $(E_b/N_0)_{\min} = -1.59$ dB. In addition, [56, Thm 7] states that flash signaling is needed to achieve $-1.59$ dB. We conclude that the schemes from Sections 4.2 and 4.2 are types of flash signaling and achieve $-1.59$ dB as they use the same $P_1$. The argument is consistent with our results that show that $P_1$ tends to 0 as the SNR decreases.

*4)Wideband slope:* The curves of the isotropical product form and discrete product form flatten as they approach $-1.59$ dB. In fact, their wideband slope $S_0$ (the slope of the rate curve at $(E_b/N_0)_{\min}$) equals 0. In [56, Thm 16] it is shown that flash signaling has $S_0 = 0$.

The non-CSI capacity must also achieve the power efficiency of $-1.59$ dB, so it must be a form of flash signaling with $S_0 = 0$. In this sense, the on-off USTM and discrete product form are first and second order optimal as they achieve the same power efficiency and wideband slope as the non-CSI capacity [56].

## 4.3. Probabilistic Amplitude Shaping for Rayleigh Block Fading Channel

In this section we evaluate performance of simple communication schemes for RBFCs.

### 4.3.1. Channel Estimation and Demodulation

Consider a transmitter as in Figure 1.3 and a RBFC with block length $T$. The receiver first demodulates the received symbols, i.e., computes the probability of the codewords' bits given the received symbols, and then the channel decoder uses the probabilities to recover the transmitted massage. In the presence of block-fading, one usually transmits pilot symbols to simplify the receiver design and this effectively reduces the transmission rate. We consider a receiver which estimates the channel and uses the estimate as if it was correct (which transfers the estimation noise to the additive noise). When pilot-power-boosting is allowed, it suffices to use one pilot symbol per block [51]. We denote the power used for pilot transmission by $\gamma_P$ and the power for data transmission by $\gamma_D$. The channel input-output relation for the pilot-based commutation system is

$$\boldsymbol{Y} = H\sqrt{\gamma}\boldsymbol{X} + \boldsymbol{N} \tag{4.19}$$

$$[Y_1, Y_2, \ldots, Y_T] = H[\sqrt{\gamma_P}, \sqrt{\gamma_D}X_2, \ldots, \sqrt{\gamma_D}X_T] + [N_1, N_2, \ldots, N_T] \tag{4.20}$$

with $\mathbb{E}[\|\boldsymbol{X}\|^2] = T$. To satisfy the power constraint as in (4.1)-(4.2) it follows that

$$\gamma_P + (T-1)\gamma_D = T\gamma. \tag{4.21}$$

Since $Y_1$ and $H$ are jointly Gaussian, the conditional distribution of $H$ given that $Y_1 = y_1$ is

$$(H|Y_1 = y_1) \sim \mathcal{N}_{\mathbb{C}}\left(\frac{\sqrt{\gamma_P}}{1+\gamma_P}y_1, \frac{1}{1+\gamma_P}\right). \tag{4.22}$$

We introduce the conditional mean channel estimate of $H$

$$\mu_{H|Y_1} = \frac{\sqrt{\gamma_P}}{1+\gamma_P}y_1$$

and the estimation error

$$\tilde{H} \sim \mathcal{N}_{\mathbb{C}}\left(0, \frac{1}{1+\gamma_P}\right).$$

Using the knowledge obtained from the pilot signal, we can see the new channel as

$$[\boldsymbol{Y}]_2^T = \mu_{H|Y_1}[\boldsymbol{X}]_2^T + \underbrace{\tilde{H}[\boldsymbol{X}]_2^T + [\boldsymbol{N}]_2^T}_{\text{noise}}. \tag{4.23}$$

The receiver knows $\mu_{H|Y_1}$ and treats the value as if it was correct (we do not update the estimate). We process the symbols independently neglecting the fact that all symbols

have the same estimation error $\tilde{H}$. This corresponds to treating the two latter expressions on the right-hand-side of (4.23) as independent additive noise. The demodulator first computes the posterior symbol probabilities

$$P_{X_t|Y_t}(x|y_t),\ x \in \mathcal{X} \tag{4.24}$$

for $t = 2, \ldots, T$, and uses the symbol probabilities to get the bit probabilities for BMD as in (1.20). Note that in the channel model (4.23), $Y_t$ conditioned on $X_t$ is Gaussian. Thus, the symbol probabilities (4.24) are straight-forward to compute using

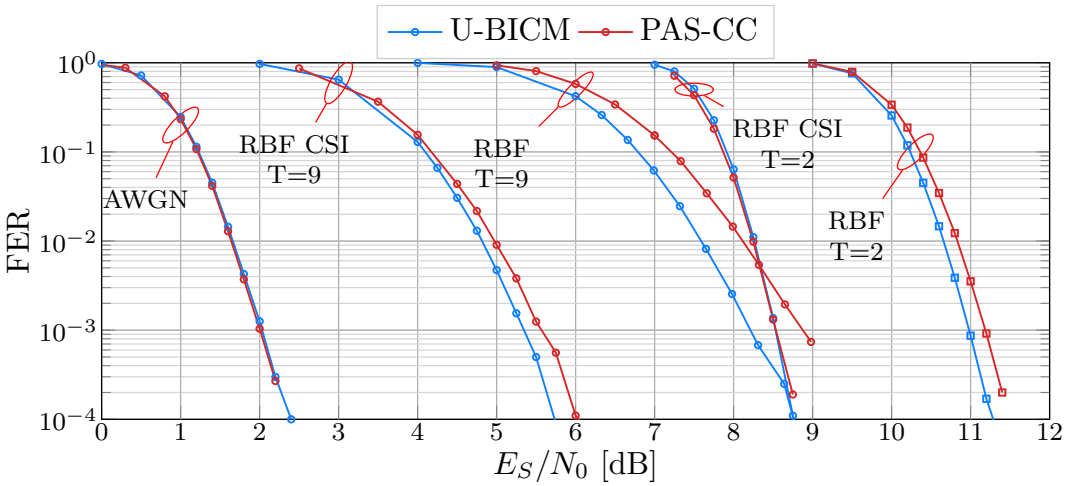
$$\begin{aligned} P_{X_t|Y_t}(x|y_t) &\propto P_{X_t}(x) p_{Y_t|X_t}(y_t|x_t) \\ &= P_{X_t}(x) \mathcal{N}_{\mathbb{C}}\left(y_t;\ \mu_{H|Y_1} x,\ 1 + \frac{|x|^2}{1+\gamma_P}\right) \end{aligned} \tag{4.25}$$

where $\mathcal{N}_{\mathbb{C}}(x; \mu, \sigma^2)$ denotes the value of the PDF of a univariate complex circular-symmetric Gaussian RV with mean $\mu$ and variance $\sigma^2$ evaluated at point $x$.

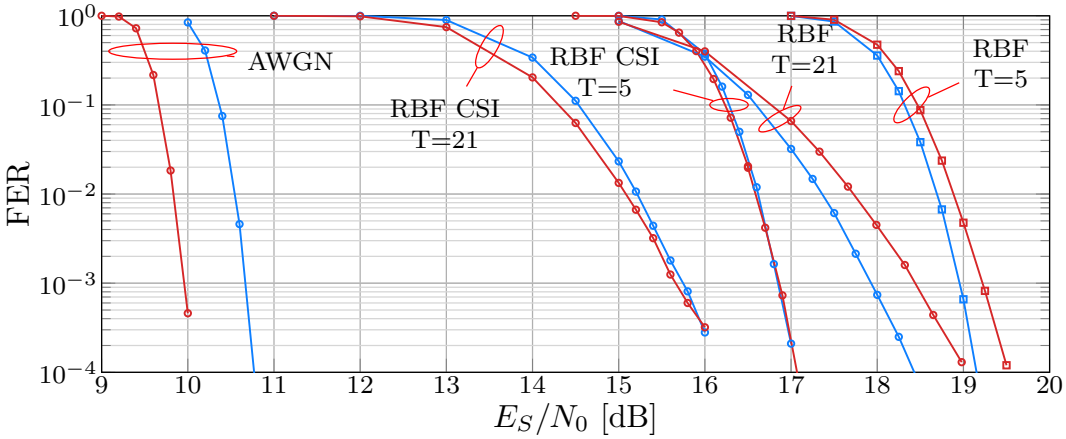
A standard BICM system would use the block channel (4.19) several times and combine the several channel uses into one codeword. For each fading block, the receiver separately estimates the channel and computes the posterior bit probabilities. Finally, the posterior bit probabilities from several fading blocks are fed to the channel decoder. Note that the demodulation step for a PAS system and a standard BICM system is the same. The only difference is (4.25) where PAS uses a nonuniform PMF $P_{X_t}$, as opposed to the standard BICM which uses a uniform $P_{X_t}$.

### 4.3.2. Results

Figure 4.3 presents FERs for classic BICM with a uniform transmit distribution (U-BICM) and PAS with CCDM (PAS-CC) for the following channels: AWGN, RBFC (denoted by RBF on the figure), RBFC when the receiver perfectly estimates the fading coefficient $H$ (denoted by RBF CSI on the figure), i.e., one slot is allocated to pilot transmission and the receiver knows $H$ exactly. Thus the performance of a scheme for the RBF CSI channel constitutes an upperbound to the performance for the RBF channel. Both schemes use the same codeword length, constellation size, and effective coding rate. One pilot symbol is used for non-AWGN channels and all schemes have the same transmission rate. The transmission rate for the non-AWGN channels includes the pilot overhead.




(a) Transmission rate = 0.89 b/CU




(b) Transmission rate = 3.0 b/CU

Figure 4.3.: Frame error rates for different channels.

Figure 4.3a presents results for the transmission rate 0.89 b/CU. All schemes use 16 QAM modulation and a codeword of length 2400 bits. For these parameters the performance of PAS-CC should be very close to that of PAS-MC from Section 4.2, thus we evaluate only the PAS-CC scheme. The PAS-CC scheme uses a channel code with coding rate 0.5 and adjusts the matching rate according to the transmission rate. PAS-CC and U-BICM perform the same for the AWGN channel and the PAS-CC scheme is slightly worse for the channels with CSI. When no CSI is available, the gap between the U-BICM and PAS-CC performances increases. The curves for larger value of $T$ are flat because the codeword compromises fewer fading blocks and it is less likely that blocks with unfavorable fading get averaged out by the blocks with favorable fading. The curves for larger values of $T$ are in general closer to the AWGN curves because of less pilot overhead.

Figure 4.3b presents results for the transmission rate 3.0 b/CU. All schemes use 64 QAM modulation and a codeword of length 9600 bits. The PAS-CC scheme uses a channel code with coding rate 2/3, and it performs significantly better than U-BICM for the AWGN channel and is slightly better for the channels with CSI. When no CSI is available, the PAS-CC performance is worse than that of U-BICM. The behavior is similar to the case with 0.89 b/CU. Both schemes do not operate very well for fading channels due to the poor performance of BMD for such channels, however PAS-CC seems to be affected more than U-BICM.

Figure 4.4 compares the performance of U-BICM and PAS-CC with the asymptotic performance of the schemes from Figure 4.2. The new curves correspond to transmission rates at an FER $10^{-3}$. All curves are computed without CSI at the receiver. Each transmitted frame constitutes 1200 fading blocks. The U-BICM schemes use different constellations. The two PAS-CC schemes use 16QAM and channel codes of rates 1/2 and 2/3, respectively. We observe that U-BICM is more energy efficient than PAS-CC. In addition the PAS-CC scheme with coding rate 1/2 outperforms the PAS-CC scheme with coding rate 2/3.

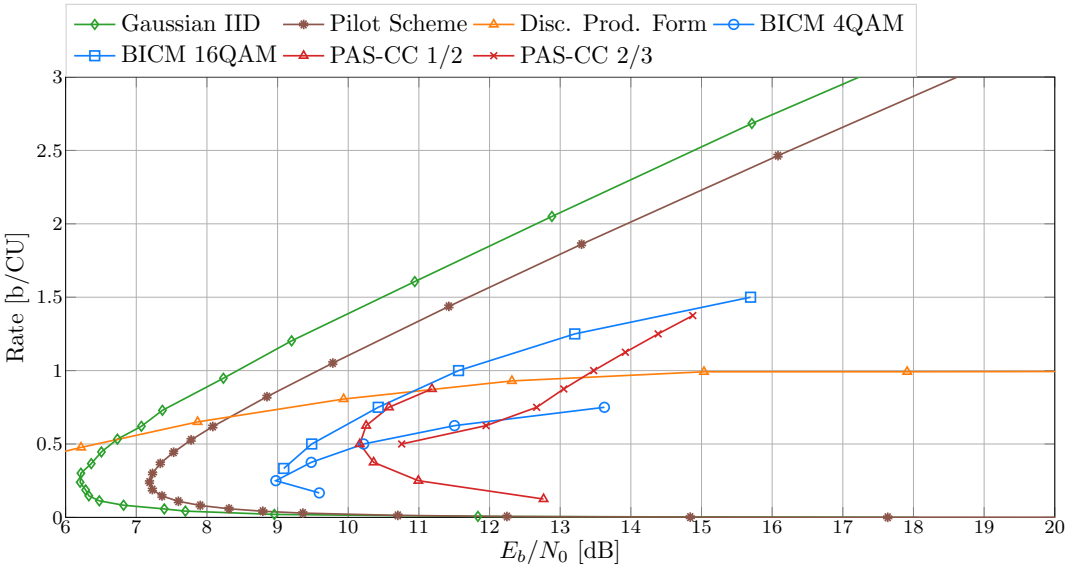



Figure 4.4.: Communication rates for RBFC with $T$=2.

# A

# Proofs for Chapter 2

## A.1. Proof of Theorem 2.1

**Lemma A.1: Some properties of** $\log_2(1+\delta x)$

Consider the function $f(x) = \log_2(1+\delta x)$ for $\delta > 0, x \geq 0$. The function $f$ has the following properties.

1.
$$x_2 > x_1 \geq 0 \implies f(x_2) - f(x_1) \leq f(x_2 - x_1) \tag{A.1}$$

2.
$$x_2 \geq 0, x_1 \geq 0 \implies f(x_1 + x_2) \leq f(x_1) + f(x_2) \tag{A.2}$$

3.
$$x_1, \ldots, x_k \geq 0 \implies f\left(\sum_{i=1}^{k} x_i\right) \leq \sum_{i=1}^{k} f(x_i) \tag{A.3}$$

*Proof.* The proof of the first property follows by

$$\begin{aligned} f(x_2) - f(x_1) &= \log_2\left(\frac{1+\delta x_1 + \delta(x_2 - x_1)}{1+\delta x_1}\right) \\ &= \log_2\left(1 + \delta\frac{x_2 - x_1}{1+\delta x_1}\right) \end{aligned}$$

$$\leq \log_2(1 + \delta(x_2 - x_1)).$$

The proof of the second property follows by

$$\begin{aligned} f(x_1) + f(x_1) &= \log_2\left(1 + \delta x_1 + \delta x_2 + \delta^2 x_1 x_2\right) \\ &\geq \log_2\left(1 + \delta x_1 + \delta x_2\right) \\ &= f(x_1 + x_2). \end{aligned}$$

The proof of the third property follows by applying the second property multiple times.

□

---

**Lemma A.2: Binary maximizer of the cost function**

Consider real numbers $x_1 > x_2 > \ldots > x_k \geq 0$ and the function $f(x) = \log_2(1 + \delta x)$ with $\frac{1}{x_1} \geq \delta > 0$. Consider a sequence $\boldsymbol{s} \in \{0,1\}^k$ with composition $\gamma(\boldsymbol{s}) = [n_0, n_1]$ where $n_0 \leq n_1$. Define the cost function

$$c(\boldsymbol{s}) = \sum_{i=1}^{k} f\left(\frac{x_i}{n_{s_i} - n_{s_i}(\boldsymbol{s}_1^{i-1})}\right). \tag{A.4}$$

**Claim:** The maximizer of the cost function is

$$\boldsymbol{z} = [\underbrace{0\ldots0}_{n_0}\underbrace{1\ldots1}_{n_1}] = \underset{\boldsymbol{s}\in\{0,1\}^k\,:\,\gamma(\boldsymbol{s})=[n_0,n_1]}{\operatorname{argmax}} c(\boldsymbol{s}). \tag{A.5}$$

Furthermore, if $n_0 < n_1$ then the maximizer $\boldsymbol{z}$ is unique.

---

*Proof.* The proof of (A.5) follows by induction. It is easy to check that for $k = 2$ the maximizer admits the form (A.5). Thus, we assume that (A.5) holds and consider the case of $\boldsymbol{s} \in \{0,1\}^{k+1}$.

Consider first the case $n_0 = n_1$. We have two candidates $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ for a maximizer of (A.4):

$$\boldsymbol{s}_1 = [0\overbrace{00\ldots0011\ldots11}^{\boldsymbol{\gamma}=[n_0-1,n_1]}] \tag{A.6}$$

$$\boldsymbol{s}_2 = [1\underbrace{11\ldots1100\ldots00}_{\boldsymbol{\gamma}=[n_0,n_1-1]}]. \tag{A.7}$$

This is because a candidate has to start with either 0 or 1 and the remaining $k$ bits follow from the inductive assumption for sequences of length $k$. Both candidates $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ match (A.5), i.e., the $\boldsymbol{s}_2$ symbols could be relabeled and achieve the same cost (A.4).

Suppose next that $n_0 < n_1$. Again, we have only two[1] candidates $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ for a maximizer of (A.4):

$$\boldsymbol{s}_1 = [0\overbrace{00\ldots0011\ldots11}^{\boldsymbol{\gamma}=[n_0-1,n_1]}] \tag{A.8}$$

$$\boldsymbol{s}_2 = [1\underbrace{00\ldots0001\ldots11}_{\boldsymbol{\gamma}=[n_0,n_1-1]}]. \tag{A.9}$$

The candidate $\boldsymbol{s}_1$ has the desired form (A.5) and we will show that it has a larger cost. The cost of each sequence is

$$c(\boldsymbol{s}_1) = f\left(\frac{x_1}{n_0}\right) + f\left(\frac{x_2}{n_0-1}\right) + \ldots + f\left(\frac{x_{n_0}}{1}\right) + f\left(\frac{x_{n_0+1}}{n_1}\right) + f\left(\frac{x_{n_0+2}}{n_1-1}\right) + \ldots + f\left(\frac{x_{k+1}}{1}\right)$$

$$c(\boldsymbol{s}_2) = f\left(\frac{x_1}{n_1}\right) + f\left(\frac{x_2}{n_0}\right) + f\left(\frac{x_3}{n_0-1}\right) + \ \ldots\ + f\left(\frac{x_{n_0+1}}{1}\right) + f\left(\frac{x_{n_0+2}}{n_1-1}\right) + \ldots + f\left(\frac{x_{k+1}}{1}\right).$$

The last $k - n_0$ terms are the same and the difference $\Delta c = c(\boldsymbol{s}_1) - c(\boldsymbol{s}_2)$ is

$$\Delta c = \left(\sum_{i=1}^{n_0} f\left(\frac{x_i}{n_0+1-i}\right) - f\left(\frac{x_{i+1}}{n_0+1-i}\right)\right) - \left(f\left(\frac{x_1}{n_1}\right) - f\left(\frac{x_{n_0+1}}{n_1}\right)\right). \tag{A.10}$$

We can upper-bound the latter bracket by

$$f\left(\frac{x_1}{n_1}\right) - f\left(\frac{x_{n_0+1}}{n_1}\right) \overset{(A.1)}{\leq} f\left(\frac{x_1 - x_{n_0+1}}{n_1}\right) \tag{A.11}$$

$$= f\left(\sum_{i=1}^{n_0} \frac{x_i - x_{i+1}}{n_1}\right) \tag{A.12}$$

$$\overset{(A.3)}{\leq} \sum_{i=1}^{n_0} f\left(\frac{x_i - x_{i+1}}{n_1}\right). \tag{A.13}$$

This results in a lower-bound

$$\Delta c \geq \sum_{i=1}^{n_0} f\left(\frac{x_i}{n_0+1-i}\right) - f\left(\frac{x_{i+1}}{n_0+1-i}\right) - f\left(\frac{x_i - x_{i+1}}{n_1}\right). \tag{A.14}$$

[1] If $n_1 = n_0 + 1$ we could have a third candidate $\boldsymbol{s}_3$ which would be the same as $\boldsymbol{s}_2$ but with the last $k$ bits inverted. However, the costs of $\boldsymbol{s}_2$ and $\boldsymbol{s}_3$ are the same due to uniform composition of the last $k$ symbols. Thus, we refrain from including $\boldsymbol{s}_3$ in the analysis.

We consider the $i$-th summand for $f(x) = \log(1 + \delta x)$. We compute

$$\begin{aligned} f\left(\frac{x_i}{n_0+1-i}\right) - f\left(\frac{x_{i+1}}{n_0+1-i}\right) - f\left(\frac{x_i - x_{i+1}}{n_1}\right) = \\ = \log_2\left(1+\delta\frac{x_i}{n_0+1-i}\right) - \log_2\left(1+\delta\frac{x_{i+1}}{n_0+1-i}\right) - \log_2\left(1+\delta\frac{x_i-x_{i+1}}{n_1}\right) \\ = \log_2\left(1+\delta\frac{x_i-x_{i+1}}{n_0+1-i+\delta x_{i+1}}\right) - \log_2\left(1+\delta\frac{x_i-x_{i+1}}{n_1}\right). \end{aligned} \tag{A.15}$$

It follows that the summands are positive if for all $i = 1, \ldots, n_0$ we have

$$\begin{aligned} n_0 + 1 - i + \delta x_{i+1} &< n_1 \\ \delta x_{i+1} &< n_1 - n_0 - 1 + i \end{aligned}$$

which is satisfied if $\delta x_2 < 1$. This follows by the assumption $\frac{1}{x_1} \geq \gamma > 0$. Thus, $\Delta c > 0$ and $\boldsymbol{s}_1$ is maximizes of (A.4). It follows that (A.5) is true.

Now we turn our attention to uniqueness. The proof also follows by induction. Consider first the non-trivial case $k = 3, n_0 = 1 < n_1 = 2$. We have $\boldsymbol{z} = [011], \boldsymbol{s}_2 = [101], \boldsymbol{s}_3 = [110]$. Using the arguments above for sequences in (A.8)–(A.9) we have $c(\boldsymbol{z}) > c(\boldsymbol{s}_2) = c(\boldsymbol{s}_3)$. Thus, we assume uniqueness for sequences of length $k$ and we consider sequences of length $k+1$. Consider sequences starting with 1-bit:

$$c(\boldsymbol{z}) > c([1\underbrace{00\ldots0001\ldots11}_{\boldsymbol{\gamma}=[n_0,n_1-1]}]) \geq c([1\boldsymbol{s}]) \text{ for } \boldsymbol{s} \in \{0,1\}^k \colon \gamma_{\boldsymbol{s}} = [n_0, n_1 - 1] \tag{A.16}$$

where the first inequality follows because $\Delta c > 0$ as for sequences in (A.8)–(A.9), and the second inequality because the suffix on the left hand side is a maximizer of length $k$ according to the first part of the lemma (note that $n_0 \leq n_1 - 1$). It remains to consider the sequences starting with 0. For $n_0 = 1$ there is only one such sequence, thus the maximizer is unique. For $n_0 > 1$ we have

$$c(\boldsymbol{z}) - c(0\boldsymbol{s}) = c(\boldsymbol{z}_2^{k+1}) - c(\boldsymbol{s}) > 0 \text{ for } \boldsymbol{s} \in \{0,1\}^k \colon \gamma_{\boldsymbol{s}} = [n_0 - 1, n_1] \tag{A.17}$$

where the inequality follows by the inductive assumption. This completes the proof. □

**Lemma A.3: Optimality of a greedy maximizer**
Consider a sequence $\boldsymbol{a} \in \{a_1, \dots, a_m\}^n = \mathcal{A}^n$, a composition $\boldsymbol{\gamma} = [n_{a_1}, \dots, n_{a_m}]$ with $\sum_1^n n_{a_i} = n$, and a scalar function $f(x) = \log_2(1 + \delta x)$ with $\frac{1}{n} \geq \delta > 0$. Define the cost function

$$c(\boldsymbol{s}) = \sum_{i=1}^{n} f\left(\frac{n+1-i}{n_{s_i} - n_{s_i}(\boldsymbol{s}_1^{i-1})}\right) \tag{A.18}$$

and consider the optimization

$$\max_{\boldsymbol{s} \in \mathcal{A}^n \colon \gamma(\boldsymbol{s}) = \boldsymbol{\gamma}} c(\boldsymbol{z}). \tag{A.19}$$

**Claim:** The greedy solution $\boldsymbol{z}$, which is defined below, is a global maximizer of the optimization. The next symbol $z_i \in \mathcal{A}$ is obtained by

$$z_i = \operatorname*{argmin}_{a \in \mathcal{A} \colon n_a > n_a(\boldsymbol{s}_1^{i-1})} n_a - n_a(\boldsymbol{s}_1^{i-1}) \tag{A.20}$$

where the constraint ensures that $\boldsymbol{z}$ has the required composition $\boldsymbol{\gamma}$ and the chosen $z_i$ maximizes the instantaneous cost increment.

*Proof.* The proof follows by contradiction. We first assume that there exists a maximizer $\boldsymbol{s}$ different then $\boldsymbol{z}$. Next, we show that $\boldsymbol{s}$ can be improved and the greedy solution $\boldsymbol{z}$ must be optimal.

Consider the maximizer $\boldsymbol{s}$ and let $i_1$ be the first position where $\boldsymbol{s}$ and $\boldsymbol{z}$ differ. We define $s_{i_1} = y$, $z_{i_1} = x$, and $I_{xy} = \{i \colon s_i = x \vee s_i = y\} = \{i_1, \dots, i_k\}$. We construct a sequence $\boldsymbol{t}$ with $t_i = s_i$ for $i \notin I_{xy}$ and $(t_{i_1}, t_{i_2}, \dots, t_{i_k}) = (x, \dots, x, y, \dots, y)$, i.e, we have

$$\boldsymbol{s} = [\dots \overbrace{s_{i_1}}^{=y} \dots s_{i_2} \dots s_{i_{k-1}} \dots s_{i_k} \dots] \tag{A.21}$$

$$\boldsymbol{t} = [\dots \; x \dots \; x \dots \; y \dots \quad y \dots \;]. \tag{A.22}$$

Sequences $\boldsymbol{t}$ and $\boldsymbol{s}$ are the same on all positions excluding $i \notin I_{xy}$, thus the difference $c(\boldsymbol{t}) - c(\boldsymbol{s})$ depends only on $\boldsymbol{s}' = [s_{i_1}, \dots, s_{i_k}]$ and $\boldsymbol{t}' = [t_{i_1}, \dots, t_{i_k}]$. $\boldsymbol{s}'$ and $\boldsymbol{t}'$ have the same composition $\boldsymbol{\gamma} = [n'_x, n'_y]$ and $n'_x < n'_y$ because $\boldsymbol{z}$ is a greedy solution, and $i_1$ is the first position where $\boldsymbol{s}$ and $\boldsymbol{z}$ differ ($s_{i_1} = y, z_{i_1} = x$). From Lemma A.2 it follows that $c(\boldsymbol{t}) - c(\boldsymbol{s}) > 0$. Thus, we improved on the optimal solution $\boldsymbol{s}$, which proves the lemma. □

**Theorem A.1: The worst-case sequence for CCDM implementation**
Consider a CCDM using the composition $\boldsymbol{\gamma} = [n_{a_1}, \ldots, n_{a_m}]$ with $n_{a_1} \leq n_{a_2} \leq \ldots \leq n_{a_m}$. A sequence

$$\boldsymbol{z} = [\underbrace{a_1 \ldots a_1}_{n_{a_1}} \underbrace{a_2 \ldots a_2}_{n_{a_2}} \ldots \underbrace{a_m \ldots a_m}_{n_{a_m}}] \tag{A.23}$$

has the largest upper bound (2.34) on the interval length among all sequences in $\mathcal{T}_{\boldsymbol{\gamma}}$, or equivalently, determines the loss

$$\Delta k = \max_{\boldsymbol{c} \in \mathcal{T}_{\boldsymbol{\gamma}}} \sum_{i=0}^{n-1} \log_2 \left( 1 + \frac{2^{-w}}{P_{C_{i+1}|\boldsymbol{C}_1^i}(c_{i+1}|\boldsymbol{c}_1^i)} \right). \tag{A.24}$$

*Proof.* We are interested in the solution of

$$\max_{\boldsymbol{c} \in \mathcal{T}_{\boldsymbol{\gamma}}} \sum_{i=1}^{n} \log_2 \left( 1 + \frac{2^{-w}}{P_{C_{i+1}|\boldsymbol{C}_1^i}(c_{i+1}|\boldsymbol{c}_1^i)} \right) = \max_{\boldsymbol{c} \in \mathcal{T}_{\boldsymbol{\gamma}}} \sum_{i=1}^{n} \log_2 \left( 1 + 2^{-w} \frac{n+1-i}{n_{c_i} - n_{c_i}(\boldsymbol{c}_1^{i-1})} \right). \tag{A.25}$$

A finite-precision implementation of the CCDM requires $2^w \geq n$, since in Section 2.2 we require $2^w \geq \Theta$ and for CCDM we use $\Theta = n - i$. This implies $\frac{1}{n} \geq \delta = 2^{-w} > 0$. From Lemma A.3, it follows that a greedy optimizer to the above problem is a global optimizer. We observe that the sequence (A.23) is a greedy optimizer. □

## A.2. Proof of Theorem 2.2

**Lemma A.4: $\overline{W}(\alpha)$ is bijective**

Consider the output alphabet $\mathcal{A} = \{a_1, \ldots, a_m\}$, a non-constant weight function $w : \mathcal{A} \to \mathbb{N}_0^+$ such that $w(a_1) \leq w(a_2) \leq \ldots \leq w(a_m)$, a family of PMFs parametrized by $\alpha > 0$:

$$Q_A(a) = \frac{2^{-\alpha w(a)}}{\sum\limits_{b \in \mathcal{A}} 2^{-\alpha w(b)}} \tag{A.26}$$

and the expected weight achieved by the distribution $Q_A$:

$$\overline{W}(\alpha) = \mathbb{E}_{Q_A}\left[w(A)\right] = \sum_{a \in \mathcal{A}} Q_A(a) w(a). \tag{A.27}$$

**Claim:** The function $\overline{W}(\alpha)$ is bijective on $\alpha \in [0, +\infty)$ with image $\Big(w(a_1), \frac{1}{|\mathcal{A}|}\sum_{a \in \mathcal{A}} w(a)\Big]$.

*Proof.* The proof follows by showing that the derivative is strictly negative for $\alpha \in [0, +\infty)$. We start with

$$\begin{aligned}
&\frac{\partial}{\partial \alpha} \sum_{a \in \mathcal{A}} w(a) \frac{2^{-\alpha w(a)}}{\sum_{b \in \mathcal{A}} 2^{-\alpha w(b)}} = \\
&= \sum_{a \in \mathcal{A}} w(a) \frac{-w(a)2^{-\alpha w(a)} \left(\sum_{b \in \mathcal{A}} 2^{-\alpha w(b)}\right) - 2^{-\alpha w(a)} \left(\sum_{b \in \mathcal{A}} -w(b)2^{-\alpha w(b)}\right)}{\left(\sum_{b \in \mathcal{A}} 2^{-\alpha w(b)}\right)^2} \\
&\overset{\text{sign}}{\propto} \left(-\left(\sum_{a \in \mathcal{A}} w^2(a)2^{-\alpha w(a)}\right)\left(\sum_{b \in \mathcal{A}} 2^{-\alpha w(b)}\right) + \left(\sum_{a \in \mathcal{A}} -w(a)2^{-\alpha w(a)}\right)\left(\sum_{b \in \mathcal{A}} -w(b)2^{-\alpha w(b)}\right)\right) \\
&= \left(\left(\sum_{a \in \mathcal{A}} w(a)2^{-\alpha w(a)}\right)^2 - \left(\sum_{a \in \mathcal{A}} w^2(a)2^{-\alpha w(a)}\right)\left(\sum_{a \in \mathcal{A}} 2^{-\alpha w(a)}\right)\right) \\
&= \left(|\boldsymbol{u}\boldsymbol{v}|^2 - \|\boldsymbol{u}\|^2 \|\boldsymbol{v}\|^2\right) < 0
\end{aligned} \tag{A.28}$$

where we introduced two vectors

$$\boldsymbol{u} = \left[w(a_1)2^{-0.5\alpha w(a_1)}, \ldots, w(a_m)2^{-0.5\alpha w(a_m)}\right] \tag{A.29}$$

$$\boldsymbol{v} = \left[2^{-0.5\alpha w(a_1)}, \ldots, 2^{-0.5\alpha w(a_m)}\right]. \tag{A.30}$$

The final step follows from the Cauchy-Schwartz inequality. Note that, since the weight

function is non-constant, the vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ can not be parallel. Thus, we have the strict inequality in (A.28). Since the function is monotonic, the image of $[0, +\infty)$ can be obtained by evaluating $\overline{W}(\alpha)$ at the boundaries, which proves the Lemma. □

---

**Lemma A.5: Marginal distribution of the entropy typical sets**
Consider a probability distribution $P_A$ on the alphabet $\mathcal{A} = \{a_1, \ldots, a_m\}$. For $\epsilon > 0$ we define the set of length-$n$ letter typical (strongly typical) sequences as

$$\mathcal{L}_\epsilon^n(P_A) = \left\{ \boldsymbol{a} \in \mathcal{A}^n \colon \left| \frac{n_{a_i}(\boldsymbol{a})}{n} - P_A(a_i) \right| < \epsilon \text{ for } i = 1, \ldots, m \right\}. \tag{A.31}$$

We also define the set of length-$n$ entropy typical (weakly typical) sequences as

$$\mathcal{H}_\epsilon^n(P_A) = \left\{ \boldsymbol{a} \in \mathcal{A}^n \colon \left| \frac{1}{n} \sum_{i=1}^{n} - \log_2 P_A(a_i) - \mathbb{H}(P_A) \right| < \epsilon \right\} \tag{A.32}$$

and the one-sided entropy typical set as

$$\mathcal{H}_{\epsilon+}^n(P_A) = \left\{ \boldsymbol{a} \in \mathcal{A}^n \colon \frac{1}{n} \sum_{i=1}^{n} - \log_2 P_A(a_i) < \mathbb{H}(P_A) + \epsilon \right\}. \tag{A.33}$$

**Claim:** The size of the sets approaches $2^{n\mathbb{H}(P_A)}$ as $n \to \infty$ and $\epsilon \to 0$, i.e., we have

$$\lim_{\epsilon \to 0} \lim_{n \to \infty} |\mathcal{L}_\epsilon^n(P_A)| = 2^{n\mathbb{H}(P_A)} \tag{A.34}$$

$$\lim_{\epsilon \to 0} \lim_{n \to \infty} |\mathcal{H}_\epsilon^n(P_A)| = 2^{n\mathbb{H}(P_A)} \tag{A.35}$$

$$\lim_{\epsilon \to 0} \lim_{n \to \infty} |\mathcal{H}_{\epsilon+}^n(P_A)| = 2^{n\mathbb{H}(P_A)}. \tag{A.36}$$

Consider a vector RV $\tilde{\boldsymbol{A}}$ uniformly distributed $\mathcal{H}_{\epsilon+}^n(P_A)$, i.e., $\tilde{\boldsymbol{A}} \sim \mathbb{U}\left[\mathcal{H}_{\epsilon+}^n(P_A)\right]$. The marginal distribution of the first symbol from $\tilde{\boldsymbol{A}}$ satisfies

$$\lim_{\epsilon \to 0} \lim_{n \to \infty} P_{\tilde{A}_1} = P_A. \tag{A.37}$$

---

*Proof.* (A.34) and (A.35) are well-known properties of typical sets and can be found in [1, 47]. We follow [1] to prove (A.36). By standard arguments, we have

$$(1 - o(n)) 2^{n(\mathbb{H}(P_A) - \epsilon)} < |\mathcal{H}_\epsilon^n(P_A)| < 2^{n(\mathbb{H}(P_A) + \epsilon)} \tag{A.38}$$

where $o(n)$ denotes a function such that $o(n) \to 0$ as $n \to \infty$. Consider a RV $\boldsymbol{A} \sim P_A^n$. We have

$$1 \geq \Pr\left[\boldsymbol{A} \in \mathcal{H}_{\epsilon+}^n(P_A)\right] = \sum_{\boldsymbol{a} \in \mathcal{H}_{\epsilon+}^n(P_A)} P_A(\boldsymbol{a}) > |\mathcal{H}_{\epsilon+}^n(P_A)| 2^{-n(\mathbb{H}(P_A)+\epsilon)} \tag{A.39}$$

where the last inequality follows because $P_A(\boldsymbol{a}) > 2^{-n(\mathbb{H}(P_A)+\epsilon)}$ for $\boldsymbol{a} \in \mathcal{H}_{\epsilon+}^n(P_A)$, see the definition of $\mathcal{H}_{\epsilon+}^n(P_A)$. From the last inequality we have

$$|\mathcal{H}_{\epsilon+}^n(P_A)| < 2^{n(\mathbb{H}(P_A)+\epsilon)} \tag{A.40}$$

and from $\mathcal{H}_{\epsilon}^n(P_A) \subseteq \mathcal{H}_{\epsilon+}^n(P_A)$ we have

$$(1 - o(n)) 2^{n(\mathbb{H}(P_A)-\epsilon)} < |\mathcal{H}_{\epsilon+}^n(P_A)| \tag{A.41}$$

which completes the proof of (A.36).
Now consider $\tilde{\boldsymbol{A}} \sim \mathbb{U}\left[\mathcal{H}_{\epsilon+}^n(P_A)\right]$. For brevity we drop the parameter $P_A$ of each of the sets. The marginal distribution of the first symbol from $\tilde{\boldsymbol{A}}$ for $\epsilon \to 0, n \to \infty$ is

$$\begin{aligned}
P_{\tilde{A}_1}(a) &= \Pr\left[\tilde{\boldsymbol{A}} \in \mathcal{L}_{\epsilon}^n\right] \Pr\left[\tilde{A}_1 = a | \tilde{\boldsymbol{A}} \in \mathcal{L}_{\epsilon}^n\right] + \Pr\left[\tilde{\boldsymbol{A}} \in \mathcal{H}_{\epsilon+}^n \backslash \mathcal{L}_{\epsilon}^n\right] \Pr\left[\tilde{A}_1 = a | \tilde{\boldsymbol{A}} \in \mathcal{H}_{\epsilon+}^n \backslash \mathcal{L}_{\epsilon}^n\right] \\
&= \frac{|\mathcal{L}_{\epsilon}^n|}{|\mathcal{H}_{\epsilon+}^n|} \Pr\left[\tilde{A}_1 = a | \tilde{\boldsymbol{A}} \in \mathcal{L}_{\epsilon}^n\right] + \frac{|\mathcal{H}_{\epsilon+}^n| - |\mathcal{L}_{\epsilon}^n|}{|\mathcal{H}_{\epsilon+}^n|} \Pr\left[\tilde{A}_1 = a | \tilde{\boldsymbol{A}} \in \mathcal{H}_{\epsilon+}^n \backslash \mathcal{L}_{\epsilon}^n\right] \\
&\overset{(a)}{=} \Pr\left[\tilde{A}_1 = a | \tilde{\boldsymbol{A}} \in \mathcal{L}_{\epsilon}^n\right] \\
&\overset{(b)}{=} \frac{1}{n} \sum_{i=1}^{n} \Pr\left[\tilde{A}_i = a | \tilde{\boldsymbol{A}} \in \mathcal{L}_{\epsilon}^n\right] \\
&= \frac{1}{n} \sum_{i=1}^{n} \sum_{\boldsymbol{a} \in \mathcal{L}_{\epsilon}^n} \frac{\mathbb{1}(a_i = a)}{|\mathcal{L}_{\epsilon}^n|} \\
&= \frac{1}{|\mathcal{L}_{\epsilon}^n|} \sum_{\boldsymbol{a} \in \mathcal{L}_{\epsilon}^n} \frac{n_{a_i}(\boldsymbol{a})}{n} \\
&\overset{(c)}{\to} P_A(a)
\end{aligned}$$

where (a) follows by (A.34) and (A.36), (b) follows by symmetry, and (c) follows from the definition of the letter typical set and vanishing $\epsilon$. □

**Theorem A.2: Asymptotic Branching Probabilities**
Consider the output alphabet $\mathcal{A} = \{a_1, \ldots, a_m\}$, a non-constant weight function $w\colon \mathcal{A} \to \mathbb{N}_0^+$ such that $w(a_1) \leq w(a_2) \leq \ldots \leq w(a_m)$, and a probability distribution on the codewords $P_{\boldsymbol{C}}$ as in (2.71) with the maximum allowed weight $n\overline{W_0}$, i.e., $\boldsymbol{C} \sim \mathbb{U}\left[\mathcal{T}^n_{n\overline{W_0}}\right]$. We assume that $w(a_1) < \overline{W_0}$. For $n \to \infty$, the probability distribution of the first symbol $C_1$ from $\boldsymbol{C}$ converges, i.e., we have

$$P_{C_1}(a_j) \xrightarrow{n\to\infty} Q_A(a_j) = \begin{cases} \dfrac{2^{-\alpha w(a_j)}}{\sum\limits_{a\in\mathcal{A}} 2^{-\alpha w(a)}}, & \text{if } \overline{W_0} < \dfrac{1}{|\mathcal{A}|}\sum\limits_{a\in\mathcal{A}} w(a) \\ \dfrac{1}{|\mathcal{A}|}, & \text{if } \overline{W_0} \geq \dfrac{1}{|\mathcal{A}|}\sum\limits_{a\in\mathcal{A}} w(a) \end{cases} \tag{A.42}$$

where $\alpha$ is chosen such that $\sum_{a\in\mathcal{A}} Q_A(a)w(a) = \overline{W_0}$.

*Proof.* We define a relaxed weight-constrained set for a weight function $w$ and average per-symbol weight $\overline{W_0}$:

$$\mathcal{W}^n_\epsilon(w, \overline{W_0}) = \left\{ \boldsymbol{a} \in \mathcal{A}^n \colon \frac{1}{n}\sum_{i=1}^{n} w(a_i) < \overline{W_0} + \epsilon \right\} \tag{A.43}$$

which for $\epsilon \to 0$ becomes the support set $\mathcal{T}^n_{n\overline{W_0}}$ of the the model $P_{\boldsymbol{C}}$.

We first consider $w(a_1) < \overline{W_0} < \frac{1}{|\mathcal{A}|}\sum_{a\in\mathcal{A}} w(a)$. Consider a PMF $R_A$ parametrized by some $\alpha > 0$

$$R_A(a) = \frac{2^{-\alpha w(a_j)}}{\sum_{a\in\mathcal{A}} 2^{-\alpha w(a)}} = \frac{2^{-\alpha w(a_j)}}{Z_\alpha}. \tag{A.44}$$

The condition defining the set (A.43) can be rewritten such that the set reads as

$$\mathcal{W}^n_\epsilon(w, \overline{W_0}) = \left\{ \boldsymbol{a} \in \mathcal{A}^n \colon \frac{1}{n}\sum_{i=1}^{n} -\log_2 R_A(a_i) < \alpha\overline{W_0} + \log_2 Z_\alpha + \alpha\epsilon \right\}. \tag{A.45}$$

We choose $\alpha$ such that the set becomes a one-sided entropy typical set for the PMF $R_A$, i.e., we require

$$\mathbb{H}(R_A) = \alpha\overline{W_0} + \log_2 Z_\alpha \tag{A.46}$$

which corresponds to choosing $\alpha$ such that

$$\sum_{a\in\mathcal{A}} R_A(a)w(a) = \overline{W_0}. \tag{A.47}$$

Lemma A.4 states that there exists a unique $\alpha$ satisfying the condition. Thus, we have

$$\mathcal{W}_\epsilon^n(w, \overline{W_0}) = \mathcal{H}_{\alpha\epsilon+}^n(R_A) \tag{A.48}$$

and from Lemma A.5 for $\epsilon \to 0$ and $n \to \infty$ the PMF of the first symbol of a vector RV $\sim \mathbb{U}(\mathcal{W}_\epsilon^n(w, \overline{W_0}))$ is $R_A$. Since $\boldsymbol{C} \sim \mathbb{U}\left[\mathcal{T}_{n\overline{W_0}}^n\right]$ and $\mathcal{W}_\epsilon^n(w, \overline{W_0}) = \mathcal{T}_{n\overline{W_0}}^n$ for $\epsilon$ small enough, it follows that $P_{C_1} = R_A$.

For $\overline{W_0} \geq \frac{1}{|\mathcal{A}|}\sum_{a\in\mathcal{A}} w(a)$ we cannot follow the same steps, since it may not by possible to find a unique $\alpha$ satisfying (A.47). Instead, we first show that the set $\mathcal{W}_\epsilon^n(w, \overline{W_0})$ contains a letter typical set with uniform PMF $R_A$. Next we show that the letter typical set dominates the marginal distribution of the first symbol. We introduce

$$\Delta\overline{W_0} = \overline{W_0} - \sum_{a\in\mathcal{A}} \frac{1}{|\mathcal{A}|} w(a) \geq 0$$

and a uniform PMF

$$R_A(a) = \frac{1}{|\mathcal{A}|} \text{ fot } a \in \mathcal{A}$$

and rewrite the set (A.43) as

$$\begin{aligned}\mathcal{W}_\epsilon^n(w, \overline{W_0}) &= \left\{\boldsymbol{a} \in \mathcal{A}^n \colon \frac{1}{n}\sum_{i=1}^n w(a_i) < \sum_{a\in\mathcal{A}} R_A(a)w(a) + \Delta\overline{W_0} + \epsilon\right\} \\ &= \left\{\boldsymbol{a} \in \mathcal{A}^n \colon \sum_{a\in\mathcal{A}} w(a)\left(\frac{n_a(\boldsymbol{a})}{n} - R_A(a)\right) < \Delta\overline{W_0} + \epsilon\right\}.\end{aligned}$$

Now consider the letter typical set with $\epsilon_1 = \frac{\epsilon}{w_{\text{MAX}}}$ and $w_{\text{MAX}} = \max_{a\in\mathcal{A}} w(a)$:

$$\mathcal{L}_{\epsilon_1}^n(R_A) = \left\{\boldsymbol{a} \in \mathcal{A}^n \colon \left|\frac{n_a(\boldsymbol{a})}{n} - R_A(a)\right| < \epsilon_1 = \frac{\epsilon}{w_{\text{MAX}}} \text{ for } a \in \mathcal{A}\right\} \tag{A.49}$$

and observe that $\mathcal{L}_{\epsilon_1}^n(R_A) \subseteq \mathcal{W}_\epsilon^n(w, \overline{W_0})$. The asymptotic size of the set $\mathcal{L}_{\epsilon_1}^n(R_A)$ is $|\mathcal{A}|^n$ (see Lemma A.5), and is the same as the upper-bound on the size of $\mathcal{W}_\epsilon^n(w, \overline{W_0})$. Thus, the sequences from the set $\mathcal{L}_{\epsilon_1}^n$ dominate the marginal distribution of the first symbol (see below). Now consider $\boldsymbol{C} \sim \mathbb{U}\left[\mathcal{W}_\epsilon^n(w, \overline{W_0})\right]$. For brevity, we skip the parameters of each of the sets. The marginal distribution of the first symbol from $\boldsymbol{C}$ for $\epsilon \to 0, n \to \infty$ is

$$P_{C_1}(a) = \Pr\left[\boldsymbol{C} \in \mathcal{L}_{\epsilon_1}^n\right] \Pr\left[C_1 = a | \boldsymbol{C} \in \mathcal{L}_{\epsilon_1}^n\right] + \Pr\left[\boldsymbol{C} \in \mathcal{W}_\epsilon^n \backslash \mathcal{L}_{\epsilon_1}^n\right] \Pr\left[C_1 = a | \boldsymbol{C} \in \mathcal{W}_\epsilon^n \backslash \mathcal{L}_{\epsilon_1}^n\right]$$

$$
\begin{aligned}
&= \frac{|\mathcal{L}^n_{\epsilon_1}|}{|\mathcal{W}^n_\epsilon|}\Pr\Big[C_1 = a|\boldsymbol{C} \in \mathcal{L}^n_{\epsilon_1}\Big] + \frac{|\mathcal{W}^n_\epsilon| - |\mathcal{L}^n_{\epsilon_1}|}{|\mathcal{W}^n_\epsilon|}\Pr\Big[C_1 = a|\boldsymbol{C} \in \mathcal{W}^n_\epsilon \backslash \mathcal{L}^n_{\epsilon_1}\Big] \\
&= \Pr\Big[C_1 = a|\boldsymbol{C} \in \mathcal{L}^n_{\epsilon_1}\Big] \\
&\overset{\text{(a)}}{=} \frac{1}{n}\sum_{i=1}^{n}\Pr\Big[C_i = a|\boldsymbol{C} \in \mathcal{L}^n_{\epsilon_1}\Big] \\
&= \frac{1}{n}\sum_{i=1}^{n}\sum_{\boldsymbol{a}\in\mathcal{L}^n_{\epsilon_1}}\frac{\mathbb{1}(a_i = a)}{|\mathcal{L}^n_{\epsilon_1}|} \\
&= \frac{1}{|\mathcal{L}^n_{\epsilon_1}|}\sum_{\boldsymbol{a}\in\mathcal{L}^n_{\epsilon_1}}\frac{n_{a_i}(\boldsymbol{a})}{n} \\
&\overset{\text{(b)}}{\to} R_A(a)
\end{aligned}
$$

where (a) follows by symmetry, and (b) follows since $\epsilon \to 0 \implies \epsilon_1 \to 0$. □

# B

# Proofs for Chapter 3

## B.1. Proof of Theorem 3.3

**Theorem B.1: Probabilistically Ordered NBC Mapping**
Consider $L$ independent bits $B_1, \ldots, B_L$ with corresponding probabilities $P_{B_1}(0), \ldots, P_{B_L}(0)$, and the joint PMF $P_{\boldsymbol{B}} = \prod_{l=1}^{L} P_{B_l}$. The bit labels $\boldsymbol{b}_i,\ i = 1, \ldots, 2^L$ ordered according to NBC with the most significant bit first, i.e., $[\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_{2^L}]$ are $[0\ldots00, 0\ldots01, \ldots, 1\ldots11]$. These vectors have non-increasing probabilities, i.e., $P_{\boldsymbol{B}}(\boldsymbol{b}_i) \geq P_{\boldsymbol{B}}(\boldsymbol{b}_{i+1})$ if and only if

$$\forall l \in \{1, \ldots, L\} \quad \mathrm{LLR}(B_l) \geq \sum_{i=l+1}^{L} \mathrm{LLR}(B_i) \tag{B.1}$$

where $\mathrm{LLR}(B_l) := \log_2\left(\frac{P_{B_l}(0)}{P_{B_l}(1)}\right)$ is the log-likelihood ratio (LLR) of the bit $B_l$.

*Proof.* The inequalities in (B.1) imply that

$$\forall l \in \{1, \ldots, L\} \quad P_{B_l}(0) \geq P_{B_l}(1). \tag{B.2}$$

Consider two bit sequences $\boldsymbol{a} = [a_1, \ldots, a_L]$ and $\boldsymbol{b} = [b_1, \ldots, b_L]$, where $\boldsymbol{a}$ has a larger

NBC number. Suppose $i$ is the first position where the sequences differ:

$$\begin{aligned}\boldsymbol{a} &= (a_1, \ldots, a_{i-1}, 1, a_{i+1}, \ldots, a_L)\\ \boldsymbol{b} &= (a_1, \ldots, a_{i-1}, 0, b_{i+1}, \ldots, b_L).\end{aligned}$$

Then we have

$$\begin{aligned}\frac{P_{\boldsymbol{B}}(\boldsymbol{a})}{P_{\boldsymbol{B}}(\boldsymbol{b})} &= \frac{P_{B_i}(1)}{P_{B_i}(0)} \prod_{l=i+1}^{L} \frac{P_{B_l}(a_l)}{P_{B_l}(b_l)}\\ &\overset{\text{(B.2)}}{\leq} \frac{P_{B_i}(1)}{P_{B_i}(0)} \prod_{l=i+1}^{L} \frac{P_{B_l}(0)}{P_{B_l}(1)}\\ &\overset{\text{(B.1)}}{\leq} 1\end{aligned}$$

which proves that the labels ordered according to the NBC have non-increasing probabilities.

We prove the converse by contradiction. Consider the $2^L - 1$ inequalities for the NBC ordered bit labels:

$$P_{\boldsymbol{B}}(\boldsymbol{b}_i) \geq P_{\boldsymbol{B}}(\boldsymbol{b}_{i+1}), i = 1, \ldots, 2^L - 1.$$

Suppose that the inequality (B.1) does not hold for the $i$-th bit-level. We have

$$\begin{aligned}\mathrm{LLR}(B_i) &< \sum_{l=i+1}^{m} \mathrm{LLR}(B_l),\\ P_{B_i}(0) \prod_{l=i+1}^{m} P_{B_l}(1) &< P_{B_i}(1) \prod_{l=i+1}^{m} P_{B_l}(0)\end{aligned}$$

which results in the contradiction

$$P_{\boldsymbol{B}}(\boldsymbol{b}_{2^{L-i}-1}) < P_{\boldsymbol{B}}(\boldsymbol{b}_{2^{L-i}})$$

for

$$\begin{aligned}\boldsymbol{b}_{2^{L-i}-1} &= [\underbrace{0 \ldots 0}_{i-1} 0 \underbrace{1 \ldots 1}_{L-i}]\\ \boldsymbol{b}_{2^{L-i}} &= [\underbrace{0 \ldots 0}_{i-1} 1 \underbrace{0 \ldots 0}_{L-i}].\end{aligned}$$

□

# C

# Notation and Abbreviations

## Mathematical Notation

| | |
|---|---|
| $\mathbb{R}$ | the set of real numbers |
| $\mathbb{N}_0$ | the set of natural number (with 0 included) |
| $\mathcal{X}$ | a set |
| $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ | the floor and ceiling function |
| $p_X$ | a probability density function of a random variable $X$ |
| $P_X$ | a probability mass function of a random variable $X$ |
| $\mathbb{U}[\mathcal{X}]$ | a random variable uniformly distributed on the set $\mathcal{X}$ |
| $\mathbb{E}[X]$ | expectation of a random variable $X$ |
| $\mathbb{H}(P_X)$ or $\mathbb{H}(X)$ | entropy of a random variable $X$ with PMF $P_X$ |
| $h(p_X)$ or $h(X)$ | differential entropy of a random variable $X$ with PDF $p_X$ |
| $\mathbb{D}(P_X\|Q_X)$ | Kullback–Leibler divergence between two PMFs |
| $\mathbb{I}(X;Y)$ | mutual information between random variables $X$ and $Y$ |
| $\boldsymbol{x}$ | a row vector (finite sequence) |
| $l(\boldsymbol{x})$ | the number of entries in $\boldsymbol{x}$ |
| $x_i$ or $[\boldsymbol{x}]_i$ | the $i$-th entry of $\boldsymbol{x}$ |
| $\boldsymbol{x}_i^j$ or $[\boldsymbol{x}]_i^j$ | a subvector (subsequence) of $\boldsymbol{x}$ |

| | |
|---|---|
| $[\boldsymbol{x}, \boldsymbol{y}]$ | a concatenation of two sequences $\boldsymbol{x}$ and $\boldsymbol{y}$ |
| $n_a(\boldsymbol{x})$ | the number of occurrences of $a$ in $\boldsymbol{x}$ |
| $\boldsymbol{\gamma}(\boldsymbol{x})$ | a composition of $\boldsymbol{x}$ |
| $P_{X,\boldsymbol{x}}$ | the empirical PMF of a sequence $\boldsymbol{x}$ |
| $\mathcal{C}^M$ | a model codebook of a distribution matcher |
| $\mathcal{C}^I$ | an implementation codebook of a distribution matcher |

# List of Abbreviations

| | |
|---|---|
| ACDM | arithmetic coding based distribution matcher |
| ASK | amplitude shift keying |
| ATSC | Advanced Television Systems Committee |
| AWGN | additive white Gaussian noise |
| b/CU | bits per channel use |
| b2b | block-to-block |
| BICM | bit-interleaved coded modulation |
| BLDM | bit-level distribution matcher |
| BMD | bit-metric decoding |
| CCDM | constant-composition distribution matcher |
| CSI | channel state information |
| DM | distribution matcher |
| DSP | digital signal processor |
| DVB | Digital Video Broadcasting |
| EDM | enumerative distribution matcher |
| ENOB | effective number of bits |
| FER | frame error rate |
| GS | geometric shaping |
| IID | independent and identically distributed |
| KL | Kullback–Leibler |
| LDPC | low-density parity-check |
| LTE | Long-Term Evolution |
| MAC | multiply–accumulate |
| MAP | maximum a posteriori |

| | |
|---|---|
| MC | Monte-Carlo |
| MCDM | multi-composition distribution matcher |
| ML | maximum likelihood |
| MSDM | multi-stream distribution matcher |
| NBC | natural binary code |
| PAS | probabilistic amplitude shaping |
| PDF | probability density function |
| PMF | probability mass function |
| PSCM | probabilistically shaped coded modulation |
| QAM | quadrature amplitude modulation |
| QPSK | quadrature phase shift keying |
| RBFC | Rayleigh block fading channel |
| RF | radio frequency |
| RV | random variable |
| SISO | single-input single-output |
| SMD | symbol-metric decoding |
| SMDM | shell mapping distribution matcher |
| SNR | signal-to-noise ratio |
| SPDM | split-parallelized distribution matcher |
| TMCDM | truncated multi-composition distribution matcher |
| USTM | unitary space-time modulation |
| WiMAX | Worldwide Interoperability for Microwave Access |

# Bibliography

[1] G. Kramer, "Topics in Multi-User Information Theory," *Foundations and Trends in Communications and Information Theory*, vol. 4, no. 4-5, pp. 265–444, Apr. 2008. [Online]. Available: http://dx.doi.org/10.1561/0100000028

[2] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, July 1948.

[3] R. G. Gallager, *Principles of Digital Communication*. Cambridge University Press, 2008.

[4] G. Forney, R. Gallager, G. Lang, F. Longstaff, and S. Qureshi, "Efficient modulation for band-limited channels," *IEEE J. Sel. Areas Commun.*, vol. 2, no. 5, pp. 632–647, Sept. 1984.

[5] G. D. Forney and L. . Wei, "Multidimensional constellations. i. introduction, figures of merit, and generalized cross constellations," *IEEE J. Sel. Areas Commun.*, vol. 7, no. 6, pp. 877–892, Aug. 1989.

[6] G. Caire, G. Taricco, and E. Biglieri, "Bit-interleaved coded modulation," *IEEE Trans. Inf. Theory*, vol. 44, no. 3, pp. 927–946, May 1998.

[7] A. Martinez, A. G. i Fabregas, G. Caire, and F. M. J. Willems, "Bit-interleaved coded modulation revisited: A mismatched decoding perspective," *IEEE Trans. Inf. Theory*, vol. 55, no. 6, pp. 2756–2765, June 2009.

[8] H. Imai and S. Hirakawa, "A new multilevel coding method using error-correcting codes," *IEEE Trans. Inf. Theory*, vol. 23, no. 3, pp. 371–377, May 1977.

[9] U. Wachsmann, R. F. H. Fischer, and J. B. Huber, "Multilevel codes: theoretical concepts and practical design rules," *IEEE Trans. Inf. Theory*, vol. 45, no. 5, pp. 1361–1391, July 1999.

[10] R1-1700679, "Non-uniform constellations in NR," *Sony, 3GPP TSG RAN1 NR Ad Hoc Meeting*, 16-20 Jan. 2017.

[11] J. Zoellner and N. Loghin, "Optimization of high-order non-uniform QAM constellations," in *2013 IEEE Int. Symp. on Broadband Multimedia Systems and Broadcasting*, June 2013.

[12] M. F. Barsoum, C. Jones, and M. Fitz, "Constellation design via capacity maximization," in *2007 IEEE Int. Symp. Inf. Theory*, June 2007.

[13] "NATSC Proposed Standard: Physical Layer Protocol (A/322)," Tech. Rep. S34, June 2016.

[14] "Digital Video Broadcasting (DVB);Next Generation broadcasting system to Handheld, physical layer specification (DVB-NGH)," Tech. Rep. A160, Nov. 2013.

[15] F. Steiner and G. Böcherer, "Comparison of geometric and probabilistic shaping with application to ATSC 3.0," in *11th Int. ITG Conf. on Systems, Commun. and Coding 2017*, Feb. 2017.

[16] G. Böcherer, F. Steiner, and P. Schulte, "Bandwidth efficient and rate-matched low-density parity-check coded modulation," *IEEE Trans. Commun.*, vol. 63, no. 12, pp. 4651–4665, Dec 2015.

[17] F. Buchali, G. Böcherer, W. Idler, L. Schmalen, P. Schulte, and F. Steiner, "Experimental demonstration of capacity increase and rate-adaptation by probabilistically shaped 64-QAM," in *2015 Eur. Conf. Opt. Commun. (ECOC)*, Sept 2015, pp. 1–3.

[18] R1-1700076, "Signal shaping for QAM constellations," *Huawei, HiSilicon, 3GPP TSG RAN1 NR Ad Hoc Meeting*, Jan. 2017.

[19] G. Böcherer and B. C. Geiger, "Optimal m-type quantizations of distributions," *CoRR*, vol. abs/1307.6843, 2013. [Online]. Available: http://arxiv.org/abs/1307.6843

[20] G. Böcherer and R. A. Amjad, "Block-to-block distribution matching," *CoRR*, vol. abs/1302.1020, 2013. [Online]. Available: http://arxiv.org/abs/1302.1020

[21] N. Abramson, *Information Theory and Coding*. McGraw-Hill, 1963.

[22] F. Rubin, "Arithmetic stream coding using fixed precision registers," *IEEE Trans. Inf. Theory*, vol. 25, no. 6, pp. 672–675, Nov. 1979.

[23] T. V. Ramabadran, "A coding scheme for m-out-of-n codes," *IEEE Trans. Commun.*, vol. 38, no. 8, pp. 1156–1163, Aug. 1990.

[24] D. J. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press 2003, 2003.

[25] K. Sayood, *Introduction to Data Compression*. Elsevier, 2006.

[26] J. Sayir, "On coding by probability transformation," Ph.D. dissertation, ETH, 1999.

[27] J. J. Rissanen, "Generalized Kraft inequality and arithmetic coding," *IBM J. Research and Development*, vol. 20, no. 3, May 1976.

[28] R. Pasco, "Source coding algorithms for fast data compression," Ph.D. dissertation, Stanford University, 1976.

[29] G. G. Langdon, "An introduction to arithmetic coding," *IBM J. Research and Development*, vol. 28, no. 2, pp. 135–149, Mar. 1984.

[30] I. H. Witten, R. M. Neal, and J. G. Clearly, "Arithmetic coding for data compression," *Commun. ACM*, 1987.

[31] M. Rabbani, R. L. Joshi, and P. W. Jones, *The JPEG 2000 Suite*. Wiley, 2009, ch. JPEG 2000 core coding system (Part 1) ., pp. 1–69.

[32] "Compression techniques | webp |." [Online]. Available: https://developers.google.com/speed/webp/docs/compression

[33] ITU-T, *Audiovisual and Multimedia Systems: Advanced video coding for generic audiovisual services (H.264)*, ITU-T Std., 2017.

[34] ——, *Audiovisual and Multimedia Systems: High efficiency video coding (H.265)*, ITU-T Std., 2018.

[35] J. Sayir, "Arithmetic coding for noisy channels," in *Proc. 1999 IEEE Inf. Theory and Commun. Workshop*, 1999, pp. 69–71.

[36] P. Schulte and G. Böcherer, "Constant composition distribution matching," *IEEE Trans. Inf. Theory*, vol. 62, no. 1, pp. 430–434, Jan 2016.

[37] C. Jones, "An efficient coding system for long source sequences," *IEEE Trans. Inf. Theory*, vol. 27, no. 3, pp. 280–291, May 1981.

[38] P. Schulte and F. Steiner, "Shell mapping for distribution matching," *CoRR*, vol. abs/1803.03614, 2018.

[39] R. Laroia, N. Farvardin, and S. A. Tretter, "On optimal shaping of multidimensional constellations," *IEEE Trans. Inf. Theory*, vol. 40, no. 4, pp. 1044–1056, July 1994.

[40] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.

[41] G. Böcherer and R. A. Amjad, "Informational divergence and entropy rate on rooted trees with probabilities," in *2014 IEEE Int. Symp. Inf. Theory*, June 2014, pp. 176–180.

[42] M. Pikus and W. Xu, "Bit-level probabilistically shaped coded modulation," *IEEE Commun. Lett.*, vol. 21, no. 9, pp. 1929–1932, Sept. 2017.

[43] G. Böcherer, P. Schulte, and F. Steiner, "High throughput probabilistic shaping with product distribution matching," *CoRR*, vol. abs/1702.07510, 2017. [Online]. Available: http://arxiv.org/abs/1702.07510

[44] M. Pikus and W. Xu, "Applying bit-level probabilistically shaped coded modulation for high-throughput communications," in *Proc. 28th IEEE Int. Symp. Personal, Indoor and Mobile Radio Commun. (PIMRC)*, Feb. 2018, pp. 1–6.

[45] Y. Gultekin, W. van Houtum, S. Serbetli, and F. Willems, "Constellation shaping for IEEE 802.11," in *Proc. 28th IEEE Int. Symp. Personal, Indoor and Mobile Radio Commun. (PIMRC)*, Feb. 2018.

[46] Y. Gultekin, F. Willems, W. van Houtum, and S. Serbetli, "Approximate enumerative sphere shaping," in *2018 IEEE Int. Symp. Inf. Theory*, Aug 2018, pp. 676–680.

[47] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York, NY, USA: Wiley-Interscience, 2006.

[48] O. İşcan and W. Xu, "Polar codes with integrated probabilistic shaping for 5G new radio," *CoRR*, vol. abs/1808.09360, 2018. [Online]. Available: https://arxiv.org/abs/1808.09360

[49] M. Pikus, G. Kramer, and G. Böcherer, "Discrete signaling for non-coherent, single-antenna, Rayleigh block-fading channels," *IEEE Commun. Lett.*, vol. 20, no. 4, pp. 764–767, April 2016.

[50] T. Marzetta and B. Hochwald, "Capacity of a mobile multiple-antenna communication link in Rayleigh flat fading," *IEEE Trans. Inf. Theory*, vol. 45, no. 1, pp. 139–157, Jan 1999.

[51] B. Hassibi and B. Hochwald, "How much training is needed in multiple-antenna wireless links?" *IEEE Trans. Inf. Theory*, vol. 49, no. 4, pp. 951–963, April 2003.

[52] B. Hochwald and T. Marzetta, "Unitary space-time modulation for multiple-antenna communications in Rayleigh flat fading," *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 543–564, Mar 2000.

[53] F. Rusek, A. Lozano, and N. Jindal, "Mutual information of IID complex gaussian signals on block Rayleigh-faded channels," *IEEE Trans. Inf. Theory*, vol. 58, no. 1, pp. 331–340, Jan 2012.

[54] B. Hassibi and T. Marzetta, "Multiple-antennas and isotropically random unitary inputs: the received signal density in closed form," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1473–1484, Jun 2002.

[55] G. Alfano, C.-F. Chiasserini, A. Nordio, and S. Zhou, "Closed-form output statistics of MIMO block-fading channels," *IEEE Trans. Inf. Theory*, vol. 60, no. 12, pp. 7782–7797, Dec 2014.

[56] S. Verdu, "Spectral efficiency in the wideband regime," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1319–1343, Jun 2002.

[57] I. Telatar and D. Tse, "Capacity and mutual information of wideband multipath fading channels," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1384–1400, Jul 2000.