Finite-Precision Implementation of Arithmetic Coding Based Distribution Matchers

Marcin Pikus*[†], Wen Xu*, and Gerhard Kramer[†]

*Huawei Technologies Duesseldorf GmbH, Munich Research Center, Riesstr. 25, 80992 Munich, Germany

[†]Institute for Communications Engineering, Technical University of Munich, Arcisstr. 21, 80290 Munich, Germany

marcin.pikus@tum.de, wen.dr.xu@huawei.com, gerhard.kramer@tum.de

Abstract—A distribution matcher (DM) encodes a binary input data sequence into a sequence of symbols with a desired target probability distribution. Several DMs, including shell mapping and constant-composition distribution matcher (CCDM), have been successfully employed for signal shaping, e.g., in optical-fiber or 5G. The CCDM, like many other DMs, is typically implemented by arithmetic coding (AC). In this work we implement AC based DMs using finite-precision arithmetic (FPA). An analysis of the implementation shows that FPA results in a rate-loss that shrinks exponentially with the number of precision bits. Moreover, a relationship between the CCDM rate and the number of precision bits is derived.

I. INTRODUCTION

A distribution matcher (DM) reversibly maps a sequence $\boldsymbol{U}$ of independent and uniformly distributed bits into a sequence $\boldsymbol{A}$ of symbols to emulate a memoryless source P_A, i.e., the output of the DM approximates a sequence of independent and identically distributed (IID) symbols, each distributed according to P_A. The accuracy of the approximation is measured by the Kullback–Leibler (KL) divergence between the probability distribution of the DM's output and the probability distribution of the IID sequence. An inverse distribution matcher (DM^{-1}) performs the inverse operation recovering $\boldsymbol{U}$ from $\boldsymbol{A}$.

DMs can be used in communication systems, such as probabilistic amplitude shaping (PAS) [1], to adjust the distribution of transmitted symbols to a distribution beneficial for a certain channel, e.g., a distribution achieving the capacity. PAS with a constant-composition distribution matcher (CCDM) [2] was recently used for optical-fiber communication [3] and proposed for the 5G mobile system [4]. DMs can also be interesting for secrecy in communications.

We refer to CCDM and other DMs implemented by arithmetic coding (AC), e.g., the multi-composition DM [5] and the multiset-partition DM [6], as AC based distribution matchers (AC-DMs). AC is applied in a reverse order for distribution matching, i.e., the AC-DM is implemented by an AC decompresser and the inverse AC-DM by an AC compresser. A direct implementation of AC requires infinite-precision arithmetic (IPA) operations. A large body of research was dedicated to efficient finite-precision arithmetic (FPA) implementation[1] of AC, see e.g. [7]–[11]. An AC-DM for CCDM was first proposed by Ramabadran in [12] for binary inputs and outputs, and by Schulte & Böcherer in [2] for larger output alphabets. CCDM has good-performance and low-complexity for long

output sequences, whereas other solutions, e.g., [5], [13], [14] are usually too complex[2] to be used with very long sequences.

In this work, we show how to adapt the technique from [12] to efficiently implement non-binary AC-DMs. We show that the FPA implementation decreases the input sequence length, and that the decrease shrinks exponentially with the number of precision bits. We also derive necessary conditions to ensure that an FPA implementation of a CCDM is one-to-one. The one-to-one property is needed for error-free decoding, but proving it is challenging because of many rounding operations performed by the encoding and decoding algorithms. Moreover, we show that the CCDM is not asymptotically optimal in terms of KL divergence when implemented in FPA. This however does not significantly affect performance in practice.

The work is organized as follows. Sec. II introduces distribution matching. Sec. III and Sec. IV describe how AC-DMs are implemented if one could use IPA, and when one uses FPA. Sec. V shows how to guarantee that an AC-DM is one-to-one and Sec. VI gives specific results for the CCDM. Sec. VII applies the ideas to several DMs.

We denote random variables (RVs) by uppercase letters, such as A, and realizations by lowercase letters, such as a. A row vector is denoted by a bold symbol, e.g., $\boldsymbol{a}$. The i-th entry in the vector $\boldsymbol{a}$ is denoted by a_i, and a subvector $[a_i, a_{i+1}, \cdots, a_j]$ of $\boldsymbol{a}$ is denoted by $\boldsymbol{a}_i^j$. The length (dimension) of a vector is denoted by $l(\boldsymbol{a})$, e.g., we have $\boldsymbol{a} = \boldsymbol{a}_1^{l(\boldsymbol{a})}$. A RV uniformly distributed on a set $\mathcal{A}$ is denoted by $\mathbb{U}_{\mathcal{A}}$, i.e., $S \sim \mathbb{U}_{\mathcal{A}}$ means that $P_S(s) = 1/|\mathcal{A}|$ for $s \in \mathcal{A}$.

II. DISTRIBUTION MATCHING

A one-to-one block-to-block DM is an injective function f_{DM} from binary input sequences $\boldsymbol{u} \in \{0,1\}^k$ to codewords $\boldsymbol{c}$ from the codebook $\mathcal{C}$:

$$f_{\text{DM}} \colon \{0,1\}^k \to \mathcal{C} \subseteq \mathcal{A}^n \tag{1}$$

where $\mathcal{A}$ is the output alphabet. The ratio $R = \frac{k}{n}$ is called the *matching rate*. A higher matching rate for a given output distribution results in a higher transmission rate. We assume that the input sequence $\boldsymbol{U}$ is a random vector consisting of k IID Bernoulli$(1/2)$ distributed bits. The output sequence of the DM is thus a random vector $\tilde{\boldsymbol{A}} = f_{\text{DM}}(\boldsymbol{U}) \sim \mathbb{U}_{\mathcal{C}}$ uniformly distributed on $\mathcal{C}$. The goal of the DM is to make

---

[1]Usually using integer operations.

[2]The memory complexity increases at least linearly with the length of the sequence.

its output "look" as if it was a sequence of IID RVs, each distributed according to a target probability distribution P_A. This is usually performed by minimizing the normalized KL divergence between the DM's output $\tilde{A}$ and the IID sequence $A \sim P_A^n = \prod_{i=1}^n P_A$:

$$\frac{1}{n}\mathbb{D}(P_{\tilde{A}}\|P_A^n) = \frac{1}{n}\sum_{c \in \mathcal{C}}\frac{1}{|\mathcal{C}|}\log_2\frac{\frac{1}{|\mathcal{C}|}}{P_A^n(c)}. \qquad (2)$$

Different approaches for implementing low-complexity mappings f_{DM} that achieve low normalized divergence can be found, e.g., in [2], [5], [6], [13]–[17].

III. ARITHMETIC CODING BASED DM

An AC-DM maps binary[3] input sequences u of length k to non-binary sequences (codewords) c of length n. The sequences c are formed by symbols from the output alphabet $\mathcal{A} = \{a_1, \ldots, a_m\}$, i.e., $c \in \mathcal{A}^n$.

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

Definition 1 Natural m-ary code number. *Consider the alphabet $\mathcal{A} = \{a_1, \ldots, a_m\}$ and a sequence $x \in \mathcal{A}^n$. The function $\text{NC}_m(\cdot)$ returns a natural m-ary code number corresponding to the sequence x, i.e.,*

$$\text{NC}_m(x) = \sum_{j=1}^{n}(\text{id}(x_j) - 1)\, m^{n-j} \qquad (3)$$

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

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

$$d(u) = \frac{\text{NC}_2(u)}{2^k}. \qquad (4)$$

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 $c_1 \in \mathcal{A}^n$ and $c_2 \in \mathcal{A}^n$, if $\text{NC}_m(c_1) < \text{NC}_m(c_2)$, then $I(c_1)$ will be placed somewhere below $I(c_2)$. We describe $I(c)$ by the beginning $x(c)$ and the width $y(c)$, i.e., $I(c) = [x(c), x(c) + y(c))$. An interval $I(c)$ can be computed recursively using a chosen probability model P_C on the codeword's symbols. P_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., $P_{C_{i+1}|C_1^i}(\cdot|s)$, where s is a sequence denoting a prefix of the codeword. The conditional cumulative probability of a letter $c \in \mathcal{A}$ is defined as

$$F_{C_{i+1}|C_1^i}(c|s) = \sum_{a \leq c}P_{C_{i+1}|C_1^i}(a|s) \qquad (5)$$

where $a \leq c$ refers to the lexicographical ordering of the alphabet's symbols. Clearly, we have $F_{C_{i+1}|C_1^i}(a_m|s) = 1$ for any s. For notational convenience we also use $F_{C_{i+1}|C_1^i}(a_0|s) = 0$ if $a_0 \notin \mathcal{A}$. The computation of the codewords' intervals can be performed by iteratively applying equations (7) and (8) below for $i = 0, \ldots, n-1$:

$$x(\lambda) = 0, \ y(\lambda) = 1 \qquad (6)$$
$$x(sa_j) = x(s) + y(s)F_{C_{i+1}|C_1^i}(a_{j-1}|s), \ j = 1, \ldots, m \quad (7)$$
$$y(sa_j) = y(s)P_{C_{i+1}|C_1^i}(a_j|s), \ j = 1, \ldots, m \qquad (8)$$

where λ denotes an empty sequence, and sa_j denotes a concatenation of s and a_j. The recursive procedure (6)–(8) gives

$$x(c) = \sum_{c' \in \mathcal{A}^n:\ c' < c}P_C(c') \qquad (9)$$

$$y(c) = \prod_{i=0}^{n-1}P_{C_{i+1}|C_1^i}(c_{i+1}|c_1^i) = P_C(c) \qquad (10)$$

where $c' < c$ refers to the lexicographical ordering of the codewords.

A one-to-one mapping between data sequences and codewords can be established if each point $d(u_i), i = 1, \ldots, 2^k$, belongs to an interval and if each interval $I(c), c \in \mathcal{A}^n$, contains at most one point $d(u)$. The first condition follows because the intervals $I(c), c \in \mathcal{A}^n$, 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.,

$$\frac{1}{2^k} \geq \max_{c \in \mathcal{A}^n}|I(c)| = \max_{c \in \mathcal{A}^n}|y(c)| = \max_{c \in \mathcal{A}^n}|P_C(c)|. \qquad (11)$$

We are interested in maximizing k and thus we often choose

$$k = k_{\text{IPA}} = \left\lfloor -\log_2\left(\max_{c \in \mathcal{A}^n}|P_C(c)|\right)\right\rfloor. \qquad (12)$$

IV. FINITE-PRECISION ARITHMETIC IMPLEMENTATION

The above described procedure requires IPA operations in general, which is infeasible in practice. Ramabadran in [12] describes how to implement a binary CCDM using FPA. We adapt this technique to implement a non-binary AC-DM with an arbitrary model P_C. Instead of using the IPA models $P_{C_{i+1}|C_1^i}, F_{C_{i+1}|C_1^i}$, we use a finite-precision integer[4] representation $\hat{F}_C$ for the cumulative model F_C from (5):

$$\hat{F}_{C_{i+1}|C_1^i}(c|s) = \left\lfloor \Theta F_{C_{i+1}|C_1^i}(c|s) + \frac{1}{2}\right\rfloor \qquad (13)$$

---

[3]Extensions to larger input alphabet sizes are straightforward.

[4]Bounded integers can be represented using a finite number of bits.

with $\hat{F}_{C_{i+1}|C_1^i}(a_0|\boldsymbol{s}) = 0$, since $a_0 \notin \mathcal{A}$. The model $\hat{P}_C$ is defined as

$$\hat{P}_{C_{i+1}|C_1^i}(a_j|\boldsymbol{s}) = \hat{F}_{C_{i+1}|C_1^i}(a_j|\boldsymbol{s}) - \hat{F}_{C_{i+1}|C_1^i}(a_{j-1}|\boldsymbol{s}). \quad (14)$$

Θ is a scaling factor used for the integer representation. It effectively converts the probability models P_C, F_C into frequency-counts models $\hat{P}_C, \hat{F}_C$ per Θ symbols. Next, we represent the subsequent intervals appearing in (6)–(8) 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 that we choose as described in (21) below)

$$x(\boldsymbol{s}) = \frac{\hat{x}(\boldsymbol{s})}{2^{L(\boldsymbol{s})+w}}, \quad y(\boldsymbol{s}) = \frac{\hat{y}(\boldsymbol{s})}{2^{L(\boldsymbol{s})+w}}. \quad (15)$$

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

$$\hat{x}(\lambda) = 0, \ \hat{y}(\lambda) = 2^w, \ L(\lambda) = 0 \quad (16)$$

$$\hat{x}(\boldsymbol{s}a_j) = \left(\hat{x}(\boldsymbol{s}) + \left\lfloor \frac{\hat{y}(\boldsymbol{s})\hat{F}_{C_{i+1}|C_1^i}(a_{j-1}|\boldsymbol{s})}{\Theta} + \frac{1}{2} \right\rfloor \right) 2^v \quad (17)$$

$$\hat{y}(\boldsymbol{s}a_j) = \left(\left\lfloor \frac{\hat{y}(\boldsymbol{s})\hat{F}_{C_{i+1}|C_1^i}(a_j|\boldsymbol{s})}{\Theta} + \frac{1}{2} \right\rfloor + \right.$$
$$\left. - \left\lfloor \frac{\hat{y}(\boldsymbol{s})\hat{F}_{C_{i+1}|C_1^i}(a_{j-1}|\boldsymbol{s})}{\Theta} + \frac{1}{2} \right\rfloor \right) 2^v \quad (18)$$

$$L(\boldsymbol{s}a_j) = L(\boldsymbol{s}) + v \quad (19)$$

where v is chosen such that[5]

$$2^w \leq \hat{y}(\boldsymbol{s}a_j) < 2^{w+1}. \quad (20)$$

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 (17) and (18) guarantees that the mantissa $\hat{y}(\boldsymbol{s}a_j)$ is at least 2^w. This provides sufficient precision for further subdivisions in (17) and (18) for the next symbols. Note that (16)–(18) round the interval boundaries rather than the width. The interval width (18) is simply a difference between the two boundaries. This ensures that the original interval is partitioned during each step. Thus, the codeword 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

$$\hat{P}_{C_{i+1}|C_1^i}(a_j|\boldsymbol{s}) > 0 \implies \hat{y}(\boldsymbol{s}a_j) > 0$$

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

$$2^w \geq \Theta. \quad (21)$$

As in the IPA case (11), we guarantee error-free decoding by choosing

$$y(\boldsymbol{c}) = \frac{\hat{y}(\boldsymbol{c})}{2^{L(\boldsymbol{c})+w}} \leq \frac{1}{2^k}, \ \forall \boldsymbol{c} \in \mathcal{A}^n. \quad (22)$$

[5]If $\hat{P}_{C_{i+1}|C_1^i}(a_j|\boldsymbol{s}) = 0$ for certain $\boldsymbol{s}, a_j$, then we may have $\hat{y}(\boldsymbol{s}a_j) = 0$ and a v satisfying (20) can not be found. This does not lead to problems as the encoder will not produce codewords with the prefix $\boldsymbol{s}a_j$ (the codeword's interval has zero length), so further subdivisions are not needed.

V. BOUNDING THE DISCREPANCY

The above described FPA scheme implements a one-to-one mapping if (22) is satisfied and the codeword intervals partition the unit interval. The latter condition is inherently satisfied by rounding the interval boundaries in (17)–(18) rather than rounding the interval length. The FPA condition (22) does not follow from the IPA condition (11). This is because the intervals computed by the FPA implementation are approximations of the IPA intervals. The discrepancy is due to the model rounding (13)–(14) and the rounding operations during the computation of (16)–(18). To assess the FPA condition (22) we must bound the rounding error. From (18) we have

$$\frac{\hat{y}(\boldsymbol{s}a_j)}{2^v} = \left\lfloor \frac{\hat{y}(\boldsymbol{s})\hat{F}_{C_{i+1}|C_1^i}(a_j|\boldsymbol{s})}{\Theta} + \frac{1}{2} \right\rfloor$$
$$- \left\lfloor \frac{\hat{y}(\boldsymbol{s})\hat{F}_{C_{i+1}|C_1^i}(a_{j-1}|\boldsymbol{s})}{\Theta} + \frac{1}{2} \right\rfloor \quad (23)$$

$$\leq \frac{\hat{y}(\boldsymbol{s})\hat{P}_{C_{i+1}|C_1^i}(a_j|\boldsymbol{s})}{\Theta} + 1 \quad (24)$$

where the second line follows by $x - 1 < \lfloor x \rfloor \leq x$ and (14). Dividing both sides by $2^{L(\boldsymbol{s})+w}$ and using (15) we get

$$y(\boldsymbol{s}a_j) \leq y(\boldsymbol{s})\frac{\hat{P}_{C_{i+1}|C_1^i}(a_j|\boldsymbol{s})}{\Theta} + 2^{-L(\boldsymbol{s})-w} \quad (25)$$

$$\leq y(\boldsymbol{s}) \left(P_{C_{i+1}|C_1^i}(a_j|\boldsymbol{s}) + \epsilon \right) + 2^{-L(\boldsymbol{s})-w} \quad (26)$$

$$\leq y(\boldsymbol{s}) P_{C_{i+1}|C_1^i}(a_j|\boldsymbol{s}) \left(1 + \frac{\epsilon + 2^{-w}}{P_{C_{i+1}|C_1^i}(a_j|\boldsymbol{s})} \right) \quad (27)$$

where in the second line we introduced ϵ to denote the maximal absolute error between the IPA model $P_{C_{i+1}|C_1^i}$ and the rounded model $\frac{1}{\Theta}\hat{P}_{C_{i+1}|C_1^i}$, e.g., $\epsilon = \frac{1}{2}\Theta^{-1}$ if rounding is used. The third line follows because $y(\boldsymbol{s}) \geq 2^{-L(\boldsymbol{s})}$. Finally, the length of the interval can be bounded by applying (27) for all codeword symbols consecutively

$$y(\boldsymbol{c}) \leq P_C(\boldsymbol{c}) \prod_{i=0}^{n-1} \left(1 + \frac{\epsilon + 2^{-w}}{P_{C_{i+1}|C_1^i}(c_{i+1}|\boldsymbol{c}_1^i)} \right) \quad (28)$$

which results in the bound (22) on the input length becoming

$$k_{\text{FPA}} \leq -\log_2 \left(\max_{\boldsymbol{c} \in \mathcal{A}^n} P_C(\boldsymbol{c}) \prod_{i=0}^{n-1} \left(1 + \frac{\epsilon + 2^{-w}}{P_{C_{i+1}|C_1^i}(c_{i+1}|\boldsymbol{c}_1^i)} \right) \right). \quad (29)$$

Inequality (29) connects the length k_{FPA} obtained for the FPA implementation and the length k_{IPA} for the IPA implementation from (12). Comparing (28) and (10) we observe that the base intervals can dilate due to the model approximation and rounding operations. This will effectively reduce the input length. Using higher precision arithmetic, i.e., larger w and Θ, results in a lower dilatation.

We emphasize the importance of (29), since checking directly[6] if an AC-DM is one-to-one is not feasible for long

[6]For example by encoding and decoding all possible input sequences.

sequences. An alternative approach may involve evaluating the right-hand-side of (29) for some randomly selected codewords from the codebook of the AC-DM and selecting the lowest obtained k_{FPA}. This approach guarantees error free decoding with high probability. Interestingly, k_{FPA} for the CCDM has a closed form expression. See next section.

We can get a more-restrictive upper bound on k_{FPA} by decomposing the maximization, i.e.,

$$
k_{\text{FPA}} \leq - \log_2 \left(\max_{\boldsymbol{c} \in \text{supp}(P_{\boldsymbol{C}})} P_{\boldsymbol{C}}(\boldsymbol{c}) \right) +
$$
$$
\underbrace{- \max_{\boldsymbol{c} \in \text{supp}(P_{\boldsymbol{C}})} \sum_{i=0}^{n-1} \log_2 \left(1 + \frac{\epsilon + 2^{-w}}{P_{C_{i+1}|\boldsymbol{C}_1^i}(c_{i+1}|\boldsymbol{c}_1^i)} \right)}_{\Delta k}
$$

where $\text{supp}(P_{\boldsymbol{C}}) = \{\boldsymbol{c} \in \mathcal{A}^n \colon P_{\boldsymbol{C}}(\boldsymbol{c}) > 0\}$. The first term is the upper bound on k_{IPA} from (12). The latter term is the input length loss (also called rate-loss) Δk due to the FPA implementation of an AC-DM ($\Delta k = 0 \implies k_{\text{FPA}} = k_{\text{IPA}}$). By using the identity $\log_2(1 + x) \leq x \log_2 e$, we get

$$
\Delta k \leq (\epsilon + 2^{-w}) \left(\max_{\boldsymbol{c} \in \text{supp}(P_{\boldsymbol{C}})} \sum_{i=0}^{n-1} \frac{\log_2 e}{P_{C_{i+1}|\boldsymbol{C}_1^i}(c_{i+1}|\boldsymbol{c}_1^i)} \right)
$$

The rate-loss Δk shrinks exponentially[7] with w, which allows to keep the rate loss small with reasonable precision.

VI. EFFICIENT IMPLEMENTATION OF THE CCDM

We now turn to designing an FPA CCDM.

Definition 2 Composition and n-type. *A composition of a vector $\boldsymbol{c} \in \mathcal{A}^n$ is a vector containing the numbers of occurrences in $\boldsymbol{c}$ of each of the symbols from the alphabet $\mathcal{A}$. We denote a composition by*

$$
\gamma(\boldsymbol{c}) := [n_{a_1}(\boldsymbol{c}), \ldots, n_{a_m}(\boldsymbol{c})] \tag{30}
$$

where $n_a(\boldsymbol{c}) = |\{i \colon c_i = a\}|$ denotes the number of occurrences of the symbol a in the sequence $\boldsymbol{c}$. An n-type Q_A is a probability distribution corresponding to the composition $\gamma(\boldsymbol{c})$

$$
Q_A(a) = \frac{n_a(\boldsymbol{c})}{n}, \quad a \in \mathcal{A}. \tag{31}
$$

Example 1. $\mathcal{A} = \{0,1\}, \boldsymbol{c} = [1011], \gamma(\boldsymbol{c}) = [1,3]$ *and* $Q_A(0) = 0.25, Q_A(0) = 0.75$.

The CCDM chooses some composition $\gamma = [n_{a_1}, \ldots, n_{a_m}]$ and the following model for AC

$$
P_{\boldsymbol{C}}(\boldsymbol{c}) = \begin{cases} \frac{1}{|\mathcal{T}_\gamma|}, & \text{if } \boldsymbol{c} \in \mathcal{T}_\gamma \\ 0, & \text{otherwise} \end{cases} \tag{32}
$$

where $\mathcal{T}_\gamma$ is a set of length-n sequences with the composition γ, i.e., $\mathcal{T}_\gamma = \{\boldsymbol{c} \in \mathcal{A}^n \colon \gamma(\boldsymbol{c}) = \gamma\}$. For the IPA implementation the input length would be $k_{\text{IPA}} = \lfloor \log_2 |\mathcal{T}_\gamma| \rfloor$. The CCDM's conditional model is

$$
P_{C_{i+1}|\boldsymbol{C}_1^i}(a_j|\boldsymbol{s}) = \frac{n_{a_j} - n_{a_j}(\boldsymbol{s})}{n - i}, \quad \text{for } a_j \in \mathcal{A}. \tag{33}
$$

[7]If ϵ shrinks exponentially which is the case when rounding is used.

The CCDM is an AC-DM with the model (32) and can be implemented using the FPA implementation presented above. The CCDM's conditional probabilities (33) can be represented exactly[8] by choosing $\Theta = n - i$ in (13), i.e., Θ is decremented after each encoded symbol. This way the rounding is avoided and k_{FPA} from (29) can be bounded by

$$
k_{\text{FPA}} \leq \log_2 |\mathcal{T}_\gamma| - \max_{\boldsymbol{c} \in \mathcal{T}_\gamma} \sum_{i=0}^{n-1} \log_2 \left(1 + \frac{2^{-w}}{P_{C_{i+1}|\boldsymbol{C}_1^i}(c_{i+1}|\boldsymbol{c}_1^i)} \right)
$$
$$
= \log_2 |\mathcal{T}_\gamma| - \Delta k \overset{\text{choose}}{\implies} k_{\text{FPA}} = \lfloor \log_2 |\mathcal{T}_\gamma| - \Delta k \rfloor \tag{34}
$$

where γ is the composition used by the CCDM. Δk can be found analytically for the CCDM, see Theorem 1. This theorem simplifies the choice of k_{FPA} for an FPA CCDM and guarantees that the CCDM is one-to-one. We note that Theorem 1 applies to any composition, i.e., the alphabet's symbols can be relabeled such that the theorem's prerequisites are satisfied.

Theorem 1 The longest interval for FPA CCDM. *Consider an FPA CCDM using the composition $\gamma = [n_{a_1}, \ldots, n_{a_m}]$ with $n_{a_1} \leq n_{a_2} \leq \ldots \leq n_{a_m}$. A sequence*

$$
\boldsymbol{z} = [\underbrace{a_1 \ldots a_1}_{n_{a_1}} \underbrace{a_2 \ldots a_2}_{n_{a_2}} \ldots \underbrace{a_m \ldots a_m}_{n_{a_m}}], \tag{35}
$$

has the largest upper bound (28) on the interval length among all sequences in $\mathcal{T}_\gamma$, or equivalently the interval $I(\boldsymbol{z})$ can be the longest after dilution due to the rounding operations. Thus, the sequence $\boldsymbol{z}$ determines the FPA rate-loss

$$
\Delta k = \sum_{i=0}^{n-1} \log_2 \left(1 + \frac{2^{-w}}{P_{C_{i+1}|\boldsymbol{C}_1^i}(z_{i+1}|\boldsymbol{z}_1^i)} \right). \tag{36}
$$

Proof. See Appendix. $\square$

In [2] the authors consider a one-to-one IPA CCDM that is asymptotically optimal.[9] It turns out that the FPA rate-loss Δk prevents the asymptotic optimality of a one-to-one FPA CCDM implemented as above. See Corollary 1.

Corollary 1 One-to-one, FPA CCDM is not asymptotically optimal. *Consider an FPA CCDM with the precision parameter w.*

1) *Suppose CCDM uses the n-type Q_A. The matching rate $R = \frac{k_{\text{FPA}}}{n}$ of the CCDM satisfies*

$$
\lim_{n \to \infty} R < \mathbb{H}(Q_A) - \log_2 \left(1 + 2^{-w} \right). \tag{37}
$$

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*

$$
\lim_{n \to \infty} \frac{1}{n} \mathbb{D}(P_{\tilde{\boldsymbol{A}}} \| P_A^n) > \log_2 \left(1 + 2^{-w} \right). \tag{38}
$$

[8]From (14), (13), and (5), the rounded model $\frac{1}{\Theta} \hat{P}_{C_{i+1}|\boldsymbol{C}_1^i}$ with $\Theta = n - i$ is equal to the IPA model $P_{C_{i+1}|\boldsymbol{C}_1^i}$.

[9]In short, the optimality means that for a target distribution P_A and properly chosen composition, we have $\frac{k}{n} \to \mathbb{H}(P_A)$ and $\frac{1}{n} \mathbb{D}(P_{\tilde{\boldsymbol{A}}} \| P_A^n) \to 0$ as $n \to \infty$, see [2] for more detail.

Fig. 1. Δk for binary CCDM with composition $\gamma = [\frac{n}{2}, \frac{n}{2}]$ and $w = 14$ obtained by Theorem 1 and the bound from [12].

Proof. From (34) we have

$$R = \frac{k_{\text{FPA}}}{n} \leq \frac{1}{n} \log_2 |\mathcal{T}_\gamma| - \frac{\Delta k}{n} \quad (39)$$

$$< \mathbb{H}(Q_A) - \log_2 \left(1 + 2^{-w}\right) \quad (40)$$

where γ is the corresponding composition. The inequality $\frac{1}{n} \log_2 |\mathcal{T}_\gamma| < \mathbb{H}(Q_A)$ follows by the optimality of the IPA CCDM. Equation (40) follows by (34) with the branching probabilities bounded by one.

Next, the divergence for the CCDM with the n-type Q_A is

$$\frac{1}{n} \mathbb{D}(P_{\tilde{A}} \| P_A^n) = \mathbb{H}(Q_A) - \frac{k_{\text{FPA}}}{n} + \mathbb{D}(Q_A \| P_A)$$

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

where the last step follows by (40). $\square$

We remark that the bounds (37) and (38) are not tight and suffice only to show that the FPA CCDM is not asymptotically optimal. Tighter bounds can be obtained by evaluating Δk using Theorem 1.

VII. RESULTS

A. Optimal m-out-of-n Codes

The paper [12] provides an upper bound on the rate-loss Δk for a binary CCDM. For the composition $\gamma = [n_0, n_1]$, the bound from [12] is

$$\Delta k < \log_2 \left(1 + \log_e \frac{1}{1 - 2^{-(w+1)} T}\right) \text{ for } 2^{-(w+1)} T < 1$$

with

$$T = \sum_{b \in \{0,1\}} n_b \left(1.5772 + \log_e n_{1-b} + \frac{1}{2 n_{1-b}}\right).$$

The optimal codes $k = \lfloor \log_2 \binom{n}{m} \rfloor$, can be constructed by the FPA CCDM only when $\Delta k < 1$. Let n_{MAX} denote the maximum output sequence length for which $\Delta k < 1$. For example, [12] approximates $n_{\text{MAX}} \approx 2390$ for $m = \frac{n}{2}$ and $w = 14$. By using Theorem 1 we obtain a tighter bound, i.e., $n_{\text{MAX}} \approx 4440$, see Fig. 1. To verify the result, we built a CCDM with $\gamma = [1600, 1600]$ and $k = 3193$, and successfully verified the encoding and decoding for 10^{10} different input sequences.

B. Low Precision CCDM

We use the algorithm from Sec. IV to implement non-binary CCDMs with precision parameters $w \in \{6, 12, 18\}$. The target distribution is $P_A(a) \propto e^{-0.004 a^2}$ for $a \in \mathcal{A} = \{1, 3, \ldots, 31\}$, and the CCDM's composition is chosen to minimize per-symbol divergence as in [2]. To guarantee that the CCDMs are one-to-one, we use (34) and Theorem 1 to choose the maximal possible input length k_{FPA}. The results are presented in Fig. 2. Due to the low arithmetic precision, the input length of the CCDM with $w = 6$ does not approach the target entropy $\mathbb{H}(P_A)$. Consequently, the normalized divergence for the low-precision CCDM is bounded away from zero and the CCDM is not asymptotically optimal.

It is difficult to see the difference in matching rates of the CCDMs with $w = 12$ and $w = 18$ due to the scale, see Fig. 2a. However, for long sequences, the divergence of the CCDM with $w = 12$ differs from the divergence of the CCDM with $w = 18$. This happens due to a small (unobservable in Fig. 2a) difference in matching rates. Finally, the CCDM with $w = 18$ performs as well as an IPA CCDM in the range $n \leq 10^4$.

Theorem 1 is useful for choosing the lowest-complexity (minimal precision) CCDM for a given target probability and output length. E.g., for $n = 100$, the CCDMs with $w = 12$ and $w = 18$ perform as well as the IPA CCDM. For $n > 1000$, the CCDM with $w = 18$ has significantly lower divergence. We observed the trend that longer output sequences and larger output alphabets require higher precision w to achieve a performance on par with the IPA CCDM.

VIII. CONCLUSIONS

We showed how to efficiently implement an AC-DM in FPA, and how to choose the input length for the CCDM to guarantee error-free decoding. The required input length depends on the precision of the arithmetic operations performed by the CCDM implementation. We observe that the precision of 18 bits allows to achieve a performance on par with the IPA implementation for an alphabet of size 16 and output sequences of length up to 10^4 symbols.

APPENDIX A
PROOF OF THEOREM 1

We begin with some lemmas.

Lemma 1 Properties of $\log_2(1 + \delta x)$. *Consider the function $f(x) = \log_2 (1 + \delta x)$ for $\delta > 0, x \geq 0$. Then f has the following properties:*

$$x_2 > x_1 \geq 0 \implies f(x_2) - f(x_1) \leq f(x_2 - x_1), \quad (41)$$

$$x_2 \geq 0, x_1 \geq 0 \implies f(x_1 + x_2) \leq f(x_1) + f(x_2), \quad (42)$$

$$x_1, \ldots, x_k \geq 0 \implies f\left(\sum_{i=1}^{k} x_i\right) \leq \sum_{i=1}^{k} f(x_i). \quad (43)$$

Lemma 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*

(a) Matching rate.

(b) Normalized divergence.

Fig. 2. Matching rate and normalized divergence for CCDM with the target distribution $P_A(a) \propto e^{-0.004a^2}$ for $a \in \mathcal{A} = \{1, 3, \ldots, 31\}$, and precision parameter $w \in \{6, 12, 18\}$.

$s \in \{0,1\}^k$ *with composition* $\gamma(s) = [n_0, n_1]$ *where* $n_0 \leq n_1$. *Define the cost function*

$$c(\boldsymbol{s}) = \sum_{i=1}^{k} f\left(\frac{x_i}{n_{s_i} - n_{s_i}(\boldsymbol{s}_1^{i-1})}\right). \quad (44)$$

Then the maximizer of the cost function is

$$\boldsymbol{z} = [\underbrace{0 \ldots 0}_{n_0} \underbrace{1 \ldots 1}_{n_1}] = \underset{\boldsymbol{s} \in \{0,1\}^k : \gamma(\boldsymbol{s}) = [n_0, n_1]}{\mathrm{argmax}} c(\boldsymbol{s}). \quad (45)$$

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

Proof. The proof was removed due to the 6-page limit for submissions. □

Lemma 3 Optimality of a greedy maximizer. *Consider a sequence* $\boldsymbol{a} \in \{a_1, \ldots, a_m\}^n = \mathcal{A}^n$, *a composition* $\gamma = [n_{a_1}, \ldots, n_{a_m}]$ *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*

$$c(\boldsymbol{s}) = \sum_{i=1}^{n} f\left(\frac{n+1-i}{n_{s_i} - n_{s_i}(\boldsymbol{s}_1^{i-1})}\right) \quad (46)$$

and consider the optimization

$$\max_{\boldsymbol{s} \in \mathcal{A}^n : \gamma(\boldsymbol{s}) = \gamma} c(\boldsymbol{z}). \quad (47)$$

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

$$z_i = \underset{a \in \mathcal{A} : n_a > n_a(\boldsymbol{s}_1^{i-1})}{\mathrm{argmin}} n_a - n_a(\boldsymbol{s}_1^{i-1}) \quad (48)$$

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

Proof. The proof was removed due to the 6-page limit for submissions. □

Finally, we prove Theorem 1. We are interested in the solution of

$$\max_{\boldsymbol{c} \in \mathcal{T}_\gamma} \sum_{i=1}^{n} \log_2\left(1 + 2^{-w} \frac{n+1-i}{n_{c_i} - n_{c_i}(\boldsymbol{c}_1^{i-1})}\right). \quad (49)$$

An FPA implementation of the CCDM requires $2^w \geq n$, since we require $2^w \geq \Theta$ (see (21)) and for CCDM we use $\Theta = n - i$. This implies $\frac{1}{n} \geq \delta = 2^{-w} > 0$. From Lemma 3, it follows that a greedy optimizer to the above problem is a global optimizer. Observe that the sequence (35) is a greedy optimizer.

REFERENCES

[1] 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.
[2] P. Schulte and G. Böcherer, "Constant composition distribution matching," *IEEE Trans. Inf. Theory*, vol. 62, no. 1, pp. 430–434, Jan 2016.
[3] F. Buchali *et al.*, "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.
[4] R1-1700076, "Signal shaping for QAM constellations," *Huawei, HiSilicon, 3GPP TSG RAN1 NR Ad Hoc Meeting*, Jan 2017.
[5] M. Pikus and W. Xu, "Arithmetic coding based multi-composition codes for bit-level distribution matching," *CoRR*, vol. abs/2581920.
[6] T. Fehenberger, D. S. Millar, T. Koike-Akino, K. Kojima, and K. Parsons, "Multiset-partition distribution matching," *IEEE Trans. Commun.*, vol. 67, no. 3, pp. 1885–1893, Mar 2019.
[7] J. J. Rissanen, "Generalized Kraft inequality and arithmetic coding," *IBM J. Research and Development*, vol. 20, no. 3, May 1976.
[8] R. Pasco, "Source coding algorithms for fast data compression," Ph.D. dissertation, Stanford University, 1976.
[9] F. Rubin, "Arithmetic stream coding using fixed precision registers," *IEEE Trans. Inf. Theory*, vol. 25, no. 6, pp. 672–675, Nov 1979.
[10] G. G. Langdon, "An introduction to arithmetic coding," *IBM J. Research and Development*, vol. 28, no. 2, pp. 135–149, Mar. 1984.
[11] I. H. Witten, R. M. Neal, and J. G. Clearly, "Arithmetic coding for data compression," *Commun. ACM*, 1987.
[12] T. V. Ramabadran, "A coding scheme for m-out-of-n codes," *IEEE Trans. Commun.*, vol. 38, no. 8, pp. 1156–1163, Aug 1990.
[13] 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.
[14] P. Schulte and F. Steiner, "Shell mapping for distribution matching," *CoRR*, vol. abs/1803.03614, 2018.
[15] M. Pikus and W. Xu, "Bit-level probabilistically shaped coded modulation," *IEEE Commun. Lett.*, vol. 21, no. 9, pp. 1929–1932, Sept 2017.
[16] F. Steiner, P. Schulte, and G. Bocherer, "Approaching waterfilling capacity of parallel channels by higher order modulation and probabilistic amplitude shaping," in *2018 52nd A. Conf. on Inf. Sciences and Systems (CISS)*, Mar 2018, pp. 1–6.
[17] O. İşcan and W. Xu, "Polar codes with integrated probabilistic shaping for 5G new radio," in *2018 IEEE 88th Veh. Technol. Conf. (VTC-Fall)*, Aug 2018, pp. 1–5.