Alma Mater Studiorum · Università di Bologna · Campus di Cesena

SCUOLA DI INGEGNERIA

Corso di Laurea in Ingegneria Elettronica e Telecomunicazioni per l'Energia

Design and Analysis of a Trellis-Based Syndrome Distribution Matching Algorithm

Elaborato in:

Progetto Di Elettronica E Telecomunicazioni LM

Relatore:
Prof. Ing.
Enrico Paolini

Presentata da:
Alessandro Cirino

Correlatori:
Dr.-Ing. habil.
Georg Böcherer

Chiar.mo Prof. Ing.
Marco Chiani

Sessione III
Anno Accademico 2017/2018

Acronyms

BPSK binary phase-shift keying

BSC binary symmetric channel

CCDM constant composition distribution matcher

CC convolutional code

DM distribution matcher

FEC forward error correction

FIR finite impulse response

IID independent and identically distributed

IUD independent and uniformly distributed

LBC linear block code

LDD low density diagonal

LUT look-up table

ML maximum likelihood

MNS maximum number of states

NRFO number of rows with flexible ones

PAS probabilistic amplitude shaping

PMF probability mass function

PS probabilistic shaping

RV random variable

SDM syndrome distribution matcher

SR systematic random

VDM Viterbi distribution matcher

Introduction

In the last years, probabilistic shaping (PS) has become a topic of interest from both academia and industry perspective [1]. When bits are modulated and sent through the channel as symbols, we have to take into account that the latter is often characterized by non-uniform capacity-achieving distributions. Therefore, PS is necessary, in order to use the maximum achievable rate of the channel during the transmission, operating closer to Shannon limit in the bandwidth-limited regime where higher order modulation is required. One of the most famous PS architectures is probabilistic amplitude shaping (PAS) [2], which concatenates a distribution matcher (DM) and a systematic forward error correction (FEC) encoder. We can therefore identify the DM as an enabling device for PS; in the specific PAS structure, DM and FEC encoder are separated and the transmission rate can be adapted by changing the distribution while using one single FEC engine [2].

Several distribution matching schemes have been proposed in the literature. Some of the most important ones are the constant composition distribution matcher (CCDM), discussed in detail in [3], which can be implemented by arithmetic coding of *m-out-of-n* codes [4], shell mapping, introduced in [1, 5], where the transmitter selects a low energy set of input sequences, called shell, and next one of them is chosen for transmission; in addiction, we may talk about trellis shaping, proposed in [6], where a sequence with minimum energy for transmission is usually selected within an ensemble. In this Thesis, a different approach based on trellis shaping is proposed, in order to decrease the DM complexity under certain conditions.

Moreover, in specific scenarios the results obtained with the proposed scheme are better or at least in line with CCDM used as benchmark.

In Chapter 1, an overview of a generic digital communication system is provided, some key quantities are introduced and some aspects of specific channel codes are shown. In Chapter 2 the DM is discussed and CCDM is presented. Chapter 3 presents trellis-based syndrome distribution matching. Results are shown in Chapter 4 and Chapter 5, where two scenarios are discussed and possible applications of the proposed DM algorithm are outlined. To conclude, final observations and possible further developments are presented.

Contents

List of Figures

List of Tables

Chapter 1

Digital Communication System

1.1 Communication System Model

This chapter introduces the communication system model, an example of which is shown in Figure 1.1. A data source generates bits according to a certain distribution and a source encoder is used to compress the stream of generated bits in a lossless way. We assume ideal data compression, thus we have independent and uniformly distributed (IUD) bits at the source encoder output. Then, a FEC encoder processes the compressed data to protect it against impairments caused by the noisy channel. After the FEC encoder, data are processed by a modulator and transmitted over the channel. As matter of fact, from modulator output to demodulator input we have a continuous-time system, with signals depending on the channel characteristics; the remaining portions of the system work in the discrete-time domain.

On the receiver side, we can observe the reverse chain compared to the transmitter side. The data sink must receive the information generated by the data source, according to some fidelity criterion. As an example, we can describe the passage from the discrete-time domain to the continuous one by a modulator performing the binary phase-shift keying (BPSK) modulation as shown in Figure 1.2.

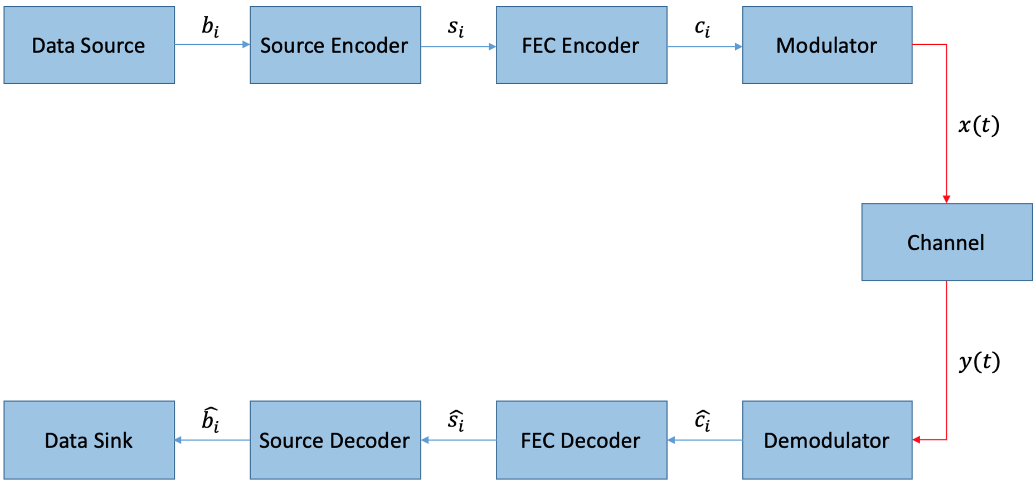

Figure 1.1: TX-RX system.

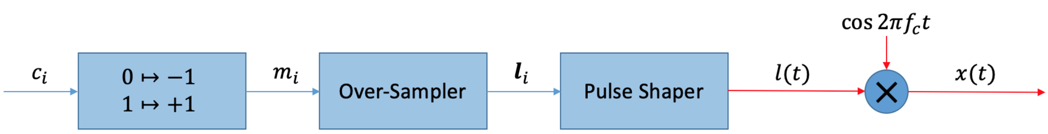

Figure 1.2: BPSK modulation scheme.

1.2 Preliminary Definitions

In this section, we describe some key quantities used in the Thesis, referring to [7]; they are necessary to introduce the distribution matching algorithms.

1.2.1 Entropy

Considering a discrete random variable (RV) X, with probability mass function (PMF) $P_X(x) = \Pr\{X = x\}$, for all $x \in \mathcal{X}$. In case $P_X(x) = 0$, let $0 \cdot \log_2 0 = 0$. The *entropy* of X is defined as:

$$\mathbb{H}(X) = -\sum_{x \in \mathcal{X}} P_X(x) \log_2 P_X(x) \tag{1.1}$$

where this value is expressed in bits [7]. This quantity is intrinsically linked to the measure of information; indeed, it corresponds to the uncertainty associated with the RV. Next, the *conditional entropy* is given by

$$\mathbb{H}(Y|X) = \sum_{x \in \mathcal{X}} P_X(x)\ \mathbb{H}(Y|X = x). \tag{1.2}$$

1.2.2 Mutual Information

Another fundamental quantity is the *mutual information* defined by

$$\mathbb{I}(X;Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} P_{X,Y}(x,y) \log_2 \frac{P_{X,Y}(x,y)}{P_X(x)P_Y(y)}. \tag{1.3}$$

The mutual information can also be expressed as

$$\mathbb{I}(X;Y) = \mathbb{H}(X) - \mathbb{H}(X|Y) = \mathbb{H}(Y) - \mathbb{H}(Y|X). \tag{1.4}$$

1.2.3 Transmission Rate

Since source coding is assumed to be ideal, each binary digit after the source encoding provides 1 bit of information. We therefore refer to the bits after the source encoder as information bits. Let us first focus the channel coding rate. Let k be the number of input bits to the channel encoder and n the corresponding output code word length. Then, the *code rate* can be defined as follows:

$$R_{\mathrm{C}} = \frac{\text{number of information bits}}{\text{number of coded bits}} = \frac{k}{n}. \tag{1.5}$$

Next, we can also consider that the mapper turns FEC bits into channel symbols; another rate can be defined, called *symbol rate*, that is described by:

$$R_{\mathrm{S}} = \frac{n}{\text{number of symbols}}. \tag{1.6}$$

Consequently, the *transmission rate* can be identified as we can see below:

$$R_{\mathrm{T}} = R_{\mathrm{C}} R_{\mathrm{S}} = \frac{k}{\text{number of symbols}}. \tag{1.7}$$

1.2.4 Capacity

For the transmission on a noisy channel, we define the *capacity* as the largest transmission rate that a system can achieve with an error probability that is arbitrarily small (Shannon channel capacity). If we consider X and Y as two random variables representing the input and output of a memoryless channel, we may characterize the channel in terms of the conditional distribution $P_{Y|X}$. By setting an input distribution P_X, the *joint distribution* can be obtained as $P_{Y,X} = P_{Y|X}P_X$. Furthermore, the *capacity* is defined based on the mutual information as

$$C = \underset{P_X}{\operatorname{argmax}}\, \mathbb{I}(X;Y). \tag{1.8}$$

There may be one or multiple distributions P_X yielding the maximum; these distributions are called capacity-achieving.

1.3 Linear Block Codes

Hereafter we focus on linear block codes (LBCs) defined over the Galois field $\mathbb{F}_2$. For any LBC we have

$$\boldsymbol{c} = \boldsymbol{x}\boldsymbol{G} \tag{1.9}$$

where $\boldsymbol{x} \in \mathbb{F}_2^k$ is the *information word*, $\boldsymbol{c} \in \mathbb{F}_2^n$ is the *code word* and $\boldsymbol{G} \in \mathbb{F}_2^{k\times n}$ is the *generator matrix*. We can also define the *parity-check matrix* $\boldsymbol{H} \in \mathbb{F}_2^{(n-k)\times n}$ through the fundamental relation

$$\boldsymbol{c}\boldsymbol{H}^{\mathrm{T}} = \boldsymbol{0}. \tag{1.10}$$

If the information word is composed by k bits, a total of 2^k code words exist. Due to the bijective mapping performed by the encoder, if the code has dimension k, then the matrix $\boldsymbol{G}$ must have k linearly independent rows; indeed, code words come in the same number of possible input sequences. The two relations (1.9) and (1.10) must be fulfilled by each of the 2^k output sequences $\boldsymbol{c}$. The code book $\mathcal{C}$ is represented by

$$\mathcal{C} = \{\boldsymbol{c} \in \mathbb{F}_2^n : \boldsymbol{c}\boldsymbol{H}^{\mathrm{T}} = \mathbf{0}\} \tag{1.11}$$

with the cardinality

$$|\mathcal{C}| = 2^k \tag{1.12}$$

and the rate of the FEC code is equivalent to (1.5). To sum up, the code book is formed by 2^k n-tuples, thus $\mathcal{C}$ is a subspace of the vector space $\mathbb{F}_2^n$. In this discussion, the parity-check matrix $\boldsymbol{H}$ is of major importance; further details on LBCs are provided in Chapter 3, where this type of code takes a main role.

1.4 Coset and Syndrome

Let $\mathcal{A}$ be a group and $\mathcal{S} \subseteq \mathcal{A}$ be a subgroup of $\mathcal{A}$; the *cosets* of $\mathcal{S}$ in $\mathcal{A}$ can be defined by $\{a + \mathcal{S} : a \in \mathcal{A}\}$. We next state some properties of the cosets:

- the number of cosets is given by

$$n_{\mathrm{c}} = \frac{|\mathcal{A}|}{|\mathcal{S}|}; \tag{1.13}$$

- considering $\mathcal{C}$ as an (n, k) LBC, defined over $\mathbb{F}_2$, we may affirm that $\mathcal{C}$ is a subgroup of $\mathbb{F}_2^n$; for each $\boldsymbol{r} \in \mathbb{F}_2^n$, the coset $\{\boldsymbol{r} + \mathcal{C}\}$ has cardinality 2^k and the number of cosets is

$$\frac{|\mathbb{F}_2^n|}{|\mathcal{C}|} = \frac{2^n}{2^k} = 2^{n-k}. \tag{1.14}$$

Proofs of the above properties and further details on coset theory may be found in [8, Chap. 2]. To summarize, the n dimensional $\mathbb{F}_2^n$ vector space is partitioned into 2^{n-k} cosets and each of these has size 2^k; furthermore, one of these cosets is the code $\mathcal{C}$. We may link each coset to an $(n-k)$-tuple $\boldsymbol{s}$

called *syndrome*, the length of which is determined by the number of cosets 2^{n-k}. Given a syndrome, the following relation must be fulfilled

$$\boldsymbol{s} = \boldsymbol{r}\boldsymbol{H}^{\mathrm{T}}. \tag{1.15}$$

We can affirm that for each syndrome an ensemble of 2^k sequences $\boldsymbol{r}$ which fulfil (1.15) exists and this corresponds to the coset size; this aspect is analysed in detail in Chapter 3.

1.5 Convolutional Codes

We introduce now convolutional codes (CCs) as linear codes with a particular algebraic structure; they are stream-oriented rather than block-oriented since the convolutional encoder is able to take a stream of bits as input and to generate output bits continuously; memory is introduced by the encoding process, as opposed to LBCs. The CCs may also be represented by states machines or trellis diagrams. In this section we want to introduce the design of a CC, analysing some of its key parameters, basing our description on [9, Chap. 4]. To describe, an example is exploited, using a representation via the schematic in Figure 1.3; the rate is $\frac{1}{2}$, due to the generation of two output bits for each data input bit and the states, corresponding to combinations of bits contained in the system memory implemented with the shift registers S1 and S2, are in this case four. Since the CC is constructed over $\mathbb{F}_2$, the adders are modulo-2 and we may identify the two discrete-time finite impulse response (FIR) filters related to the top and bottom branches of the encoder; two *generators* in $\mathbb{F}_2$ may be associated to these FIR filters, using the octal representation with the most significant bit on the left of the generator binary vector according to [10, Chap. 12] and described in Table 1.1. For example in the considered case study, even if we cannot appreciate the octal representation because it coincides with the decimal, we can represent

Decimal	Octal	Binary
0	0	000
1	1	001
2	2	010
3	3	011
4	4	100
5	5	101
6	6	110
7	7	111

Table 1.1: Octal Table.

the generators as

$$\begin{aligned} \boldsymbol{g}^1 &= [g_0^1 \; g_1^1 \; g_2^1] = [1\ 0\ 1]; \\ \boldsymbol{g}^2 &= [g_0^2 \; g_1^2 \; g_2^2] = [1\ 1\ 1]. \end{aligned} \tag{1.16}$$

Owing to them, the outputs may be represented by the convolution between the input and generators:

$$\begin{aligned} \boldsymbol{c}^1 &= \boldsymbol{u} * \boldsymbol{g}_1; \\ \boldsymbol{c}^2 &= \boldsymbol{u} * \boldsymbol{g}_2. \end{aligned} \tag{1.17}$$

Hence, using the example generators, we may represent the generic i-th output elements c_i^1 and c_i^2 by

$$\begin{aligned} c_i^1 &= u_i + u_{i-2}; \\ c_i^2 &= u_i + u_{i-1} + u_{i-2}. \end{aligned} \tag{1.18}$$

Hereafter, we deal with the semi-infinite representation of the CCs. Let $\boldsymbol{u} = (u_0 \; u_1 \ldots)$ be the *input data sequence* and let $\boldsymbol{c} = (c_0 \; c_1 \ldots)$ be the corresponding *output code sequence*. Hence, a *generator matrix* $\boldsymbol{G}$ may be defined as for LBCs, in fact, a similar encoder description is provided by

$$\boldsymbol{c} = \boldsymbol{u}\boldsymbol{G}. \tag{1.19}$$

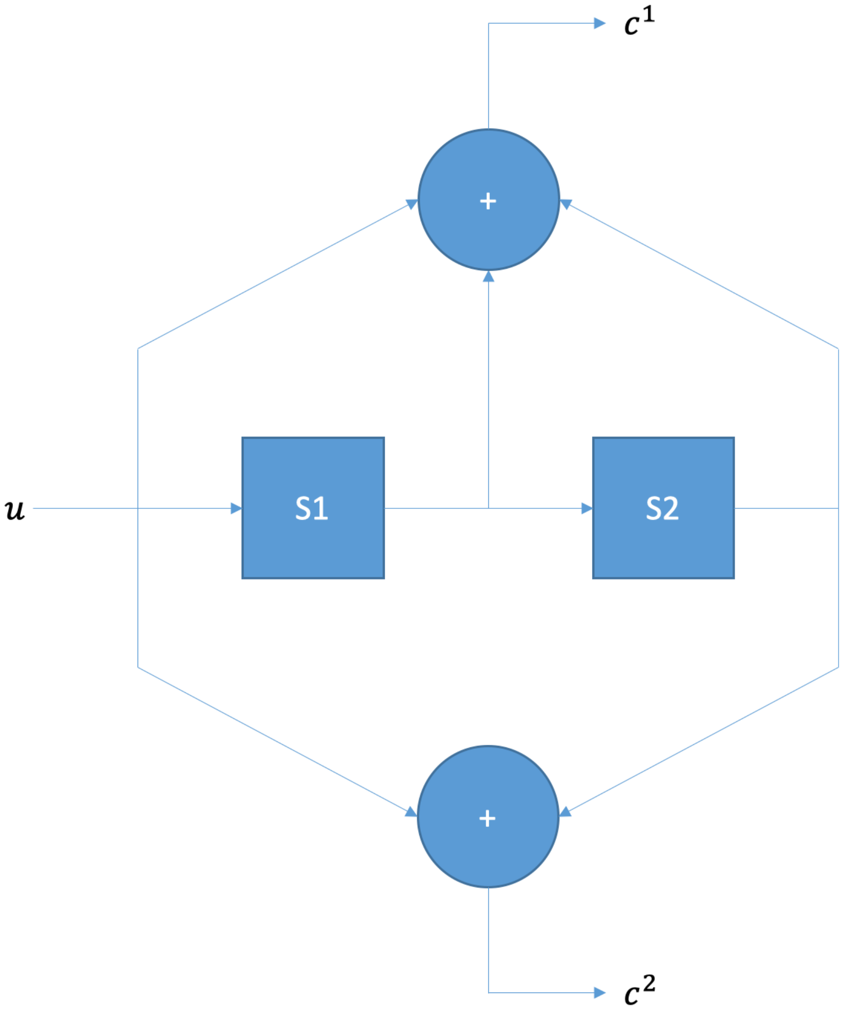

Figure 1.3: Rate $\frac{1}{2}$ convolutional code, generators [3,5].

The only difference is that the generator matrix for CCs has a semi-infinite structure, depending on how many data have to be encoded. Thus, the matrix $\boldsymbol{G}$ may be written as

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{G}_0 & \boldsymbol{G}_1 & \boldsymbol{G}_2 & \dots & \boldsymbol{G}_m & 0 & 0 & \dots \\ & \boldsymbol{G}_0 & \boldsymbol{G}_1 & \dots & \boldsymbol{G}_{m-1} & \boldsymbol{G}_m & 0 & \dots \\ & & \boldsymbol{G}_0 & \dots & \boldsymbol{G}_{m-2} & \boldsymbol{G}_{m-1} & \boldsymbol{G}_m & \dots \\ & & & \ddots & \vdots & \vdots & \vdots & \\ & & & & \boldsymbol{G}_0 & \boldsymbol{G}_1 & \boldsymbol{G}_2 & \dots \\ & & & & & \boldsymbol{G}_0 & \boldsymbol{G}_1 & \dots \\ & & & & & & \boldsymbol{G}_0 & \dots \\ & & & & & & & \ddots \end{bmatrix} \tag{1.20}$$

where

$$\boldsymbol{G}_l = \begin{bmatrix} g_{l1}^1 & g_{l1}^2 & \cdots & g_{l1}^n \\ g_{l2}^1 & g_{l2}^2 & \cdots & g_{l2}^n \\ \vdots & \vdots & & \vdots \\ g_{lk}^1 & g_{lk}^2 & \cdots & g_{lk}^n \end{bmatrix}, \quad \text{with } l = 0, 1, \dots, m. \tag{1.21}$$

Considering the case with input bits block size $k = 1$, number of output bits $n = 2$ (which corresponds to the number of generators), and the number of shift registers $m = 2$, we can recast (1.21) as

$$\boldsymbol{G}_l = [g_l^1 \; g_l^2], \quad \text{with } l = 0, 1, 2. \tag{1.22}$$

In this specific case study, due to all the considerations done so far, the generator matrix assumes the following form:

$$\boldsymbol{G} = \begin{bmatrix} 1\,1 & 0\,1 & 1\,1 & & & \\ & 1\,1 & 0\,1 & 1\,1 & & \\ & & 1\,1 & 0\,1 & 1\,1 & \\ & & & \ddots & \ddots & \ddots \end{bmatrix}. \tag{1.23}$$

Furthermore, the parity-check matrix can be obtained through (1.10) that, although introduced for LBCs, remains valid also for CC. Knowing that due to the m value only 3 input bits are useful to obtain a certain output element as we shown in (1.18), we can represent the input vector of interest as

$$\boldsymbol{u}^i = [u_i \ u_{i-1} \ u_{i-2}]. \tag{1.24}$$

Then, we can provide the generic two output bits at step i of the CC encoder by

$$[c_i^1 \ c_i^2] = [\boldsymbol{u}^i \boldsymbol{g}_1^{\mathrm{T}} \ \boldsymbol{u}^i \boldsymbol{g}_2^{\mathrm{T}}]. \tag{1.25}$$

Considering the specific case with $m = 3$, the $\boldsymbol{H}^{\mathrm{T}}$ can be expressed as

$$\boldsymbol{H}^{\mathrm{T}} = \begin{bmatrix} h_{1,1}^1 & h_{1,2}^1 & h_{1,3}^1 & 0 & \dots & & & & & & \\ h_{2,1}^1 & h_{2,2}^1 & h_{2,3}^1 & 0 & \dots & & & & & & \\ 0 & h_{1,2}^2 & h_{1,3}^2 & h_{1,4}^2 & 0 & \dots & & & & & \\ 0 & h_{2,2}^2 & h_{2,3}^2 & h_{2,4}^2 & 0 & \dots & & & & & \\ & & & & & \ddots & & & & & \\ & & & & & \dots & 0 & h_{1,i}^i & h_{1,i+1}^i & h_{1,i+2}^i & 0 & \dots \\ & & & & & \dots & 0 & h_{2,i}^i & h_{2,i+1}^i & h_{2,i+2}^i & 0 & \dots \\ & & & & & & & & & & & \ddots \end{bmatrix} \tag{1.26}$$

that is, we can define

$$\begin{aligned} \boldsymbol{h}_1^i &= [h_{1,i}^i \ h_{1,i+1}^i \ h_{1,i+2}^i]; \\ \boldsymbol{h}_2^i &= [h_{2,i}^i \ h_{2,i+1}^i \ h_{2,i+2}^i]. \end{aligned} \tag{1.27}$$

We can therefore define $\boldsymbol{H}_i^{\mathrm{T}}$ by

$$\boldsymbol{H_i}^{\mathrm{T}} = \begin{bmatrix} \boldsymbol{h}_1^i \\ \boldsymbol{h}_2^i. \end{bmatrix} \tag{1.28}$$

In order to fulfil (1.10), we can assure the modulo-2 sum equal to 0 between each of the consecutive two i-th elements of the code word c_i^1 and c_i^2; hence, we may determine the rows of the transposed parity-check matrix as

$$[c_i^1 \ c_i^2]\boldsymbol{H}_i^{\mathrm{T}} = [\boldsymbol{u}^i \boldsymbol{g}_1^{\mathrm{T}} \ \boldsymbol{u}^i \boldsymbol{g}_2^{\mathrm{T}}]\boldsymbol{H_i}^{\mathrm{T}} = [\boldsymbol{u}^i \boldsymbol{g}_1^{\mathrm{T}} \ \boldsymbol{u}^i \boldsymbol{g}_2^{\mathrm{T}}] \begin{bmatrix} \boldsymbol{h}_1^i \\ \boldsymbol{h}_2^i. \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}^i \boldsymbol{g}_1^{\mathrm{T}} \boldsymbol{h}_1^i \\ \boldsymbol{u}^i \boldsymbol{g}_2^{\mathrm{T}} \boldsymbol{h}_2^i \end{bmatrix} = 0 \tag{1.29}$$

that is, using $\overset{!}{=}$ to emphasise that the left-hand side *must be equal to* the right-hand side,

$$\begin{bmatrix} \boldsymbol{h}_1^i \\ \boldsymbol{h}_2^i \end{bmatrix} \overset{!}{=} \begin{bmatrix} \boldsymbol{g}_2 \\ \boldsymbol{g}_1 \end{bmatrix}. \tag{1.30}$$

Then, we can provide the representation of the transposed parity-check matrix

$$\boldsymbol{H}^{\mathrm{T}} = \begin{bmatrix} 1 & 1 & 1 & & & & \\ 1 & 0 & 1 & & & & \\ & 1 & 1 & 1 & & & \\ & 1 & 0 & 1 & & & \\ & & 1 & 1 & 1 & & \\ & & 1 & 0 & 1 & & \\ & & & & & \ddots & \end{bmatrix} \tag{1.31}$$

and infer the parity-check matrix

$$\boldsymbol{H} = \begin{bmatrix} 1\,1 & & & & & \\ 1\,0 & 1\,1 & & & & \\ 1\,1 & 1\,0 & 1\,1 & & & \\ & 1\,1 & 1\,0 & 1\,1 & & \\ & & 1\,1 & 1\,0 & 1\,1 & \\ & & & \ddots & \ddots & \ddots \end{bmatrix}. \tag{1.32}$$

To conclude, we want to emphasise the possibility to exploit the sliding nature of the CCs, which permits to decode information after the channel in a very efficient way with an acceptable complexity.

Chapter 2

Distribution Matching

To enable PS, there are different distribution matching approaches, based on a common structure, in attempting to provide precise PMFs and in this section after explaining the idea behind the DM we want to introduce one of the most famous methods called CCDM [3], analysing several important aspects of it. For a deeper analysis we refer the reader to [11, Chap. 9].

2.1 Distribution Matching Structure

Considering the system scheme proposed in Section 1.1, we can recast Figure 1.1 by the following representation in Figure 2.1, where the DM block is added in an end-to-end system and we neglect the source encoder (the same at the decoding side) to simplify the system. If we consider the PAS architecture, we can identify the PS system in the concatenation of DM and FEC encoder because they are decoupled [2]. We can therefore emphasise the role of the DM cutting it from the general chain and analysing it in detail, with regard to Figure 2.2. Given an input sequence of length k, the binary DM generates an output sequence of length n named $\boldsymbol{x}$ which follows the particular averaged binary distribution P_X. This latter can be identified, along all the possible output sequences related by the one-to-one relation

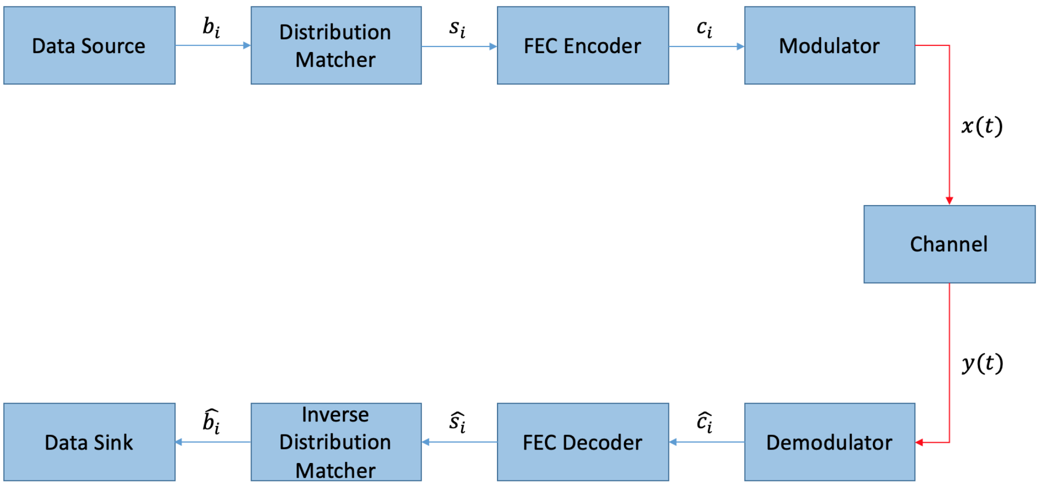

Figure 2.1: TX-RX system with DM.

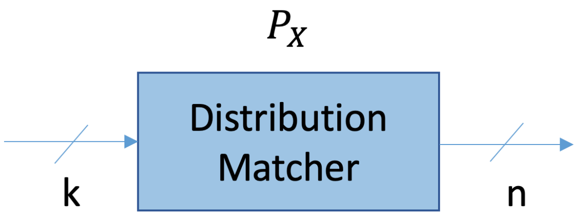

Figure 2.2: DM block.

with the 2^k input sequences, as

$$P_X(1) = \frac{\sum_{i=1}^{2^k} w_{\mathrm{H}}(\boldsymbol{x}_i)}{2^k \cdot n}, \tag{2.1}$$

$$P_X(0) = 1 - P_X(1) \tag{2.2}$$

and the rate of this block has the same expression of (1.5), so

$$R_{\mathrm{DM}} = \frac{k}{n}. \tag{2.3}$$

Remark. *Given a DM, we can evaluate the entropy of the averaged probability distribution P_X. Considering a receiver that assumes each element of $\boldsymbol{X}$ as independent and identically distributed (IID) with PMF P_X, evaluating* *the entropy as $\mathbb{H}(P_X)$ is worthwhile, keeping in mind that we can also use this particular DM suitable for short sequences on a stream of frames derived from a long sequence as well. Hence, in this specific scenario, we can compare the DM rate and the $\mathbb{H}(P_X)$.*

2.2 Types

To provide an exhaustive introduction to CCDM, we have to define the *types* [12, Sec. II] and their properties. Let $x^n = x_1 x_2 \dots x_n$ be a sequence with entries in a finite alphabet $\mathcal{X}$. If we define the number of times that letter $a \in \mathcal{X}$ occurs in the sequence x^n with

$$N(a|x^n) = |\{i \in \{1, 2, \dots, n\} : x_i = a\}|, \quad a \in \mathcal{X}; \tag{2.4}$$

then we may affirm that the sequence x^n has empirical (or sample) distribution defined by

$$P_{x^n}(a) = \frac{N(a|x^n)}{n}, \quad a \in \mathcal{X}. \tag{2.5}$$

It can be observed that each permutation of x^n has the same empirical distribution, so we may also define $n_a = N(a|x^n)$ and modify (2.5) writing

$$P_X(a) = \frac{n_a}{n}, \quad a \in \mathcal{X}. \tag{2.6}$$

Since every $P_X(a), a \in \mathcal{X}$, is an integer multiple of $\frac{1}{n}$, the distribution P_X is called an n-type. Let $\mathcal{T}^n(P_X)$ be the type class of the n-type P_X, corresponding to the set of all the n sequences with empirical distribution P_X.

2.3 CCDM

Having defined the types, we are now in a position to introduce the constant composition code $\mathcal{C} \subseteq \mathcal{T}^n(P_X)$; the latter is necessary to define CCDM, which is able to encode a k elements input sequence into n elements output sequence, implementing a fixed-to-fixed length mapping into $\mathcal{T}^n(P_X)$. Following the classical definition (1.5), we can define the rate of CCDM by

$$R_{\mathrm{CCDM}} = \frac{k}{n}. \tag{2.7}$$

Thinking about an end-to-end chain, we are typically interested in having the rate as large as possible to avoid large overhead; then, being x_i the generic i-th element of the alphabet $\mathcal{X}$ with $|\mathcal{X}| = M$, we choose an input length equal to

$$k = \lfloor \log_2 |\mathcal{T}^n(P_X)| \rfloor = \left\lfloor \log_2 \binom{n}{n_{x_1}, \dots, n_{x_M}} \right\rfloor \tag{2.8}$$

picking the maximum value which guarantees the mapping; the proof that the class cardinality is equal to the multinomial coefficient presented above is provided in [11, Sec. 9.2]. Furthermore, we may show two fundamental relations; the first, which gives an upper bound to the rate by the entropy, is

$$R_{\mathrm{CCDM}}(P_X, n) \leq \mathbb{H}(P_X); \tag{2.9}$$

the second describes the *rate loss* as

$$R_{\mathrm{loss}} = \mathbb{H}(P_X) - R_{\mathrm{CCDM}}(P_X, n). \tag{2.10}$$

Due to the proofs shown in [11, Chap. 9], we may affirm that, if the output length tends to infinity, the CCDM rate converges to the entropy, while the rate loss approaches to zero; so, we may write

$$R_{\mathrm{CCDM}}(P_X, n) \xrightarrow{n\to\infty} \mathbb{H}(P_X). \tag{2.11}$$

Example 2.3.1. To proceed, an example of non-binary CCDM is provided; let the alphabet be $\mathcal{X} = \{1, 5\}$ and output length $n = 4$. Given the following desired distribution

$$P_X(1) = \frac{1}{4}, \quad P_X(5) = \frac{3}{4} \tag{2.12}$$

where the probabilities are a multiple of $\frac{1}{4}$, P_X must be a 4-type because of the n value. If we want to represent the 4-type class we can use the following description

$$\mathcal{T}^4(P_X) = \Big\{(1,5,5,5),(5,1,5,5),(5,5,1,5),(5,5,5,1)\Big\}; \tag{2.13}$$

from the latter expression we may say that $n_1 = 1$ and $n_5 = 3$. From (2.8), having $n = 4$, we can evaluate the input length k by

$$k = \lfloor \log_2 |\mathcal{T}^4(P_X)| \rfloor = 2 \tag{2.14}$$

that is, we can create the following look-up table (LUT) that defines CCDM as an invertible one-to-one mapping:

$$00 \mapsto (1,5,5,5), \tag{2.15}$$
$$01 \mapsto (5,1,5,5), \tag{2.16}$$
$$10 \mapsto (5,5,1,5), \tag{2.17}$$
$$11 \mapsto (5,5,5,1). \tag{2.18}$$

We can even evaluate the CCDM rate from (2.7) as

$$R_{\text{CCDM}} = \frac{\log_2 4}{4} = \frac{1}{2} \tag{2.19}$$

and the entropy from (1.1) as

$$\mathbb{H}(P_X) = 0.8113. \tag{2.20}$$

Therefore, we can estimate the rate loss from (2.10) by

$$R_{\text{loss}} = 0.3113. \tag{2.21}$$

To see empirically that for large n the rate loss vanishes and the CCDM rate approaches the entropy, we can set $n = 10000$, using

$$n_1 = \frac{10000}{4} = 2500, \tag{2.22}$$
$$n_5 = 3\frac{10000}{4} = 7500; \tag{2.23}$$

in this case k becomes

$$k = \lfloor \log_2 |\mathcal{T}^{10000}(P_X)| \rfloor = 8106 \tag{2.24}$$

so that the CCDM rate becomes

$$R_{\text{CCDM}} = \frac{8106}{10000} = 0.8106 \tag{2.25}$$

a value which is much closer to the entropy $\mathbb{H}(P_X)$.

To conclude, a brief introduction to CCDM was provided, showing its rate and other characteristics, also providing an example. In real systems, we might be interested in using CCDM with a distribution which is not an n-type; then, approximation via distribution quantization exists [11, Sec 9.3].

2.4 CCDM Algorithm

In this brief section, we want to point out a particular algorithm which implements binary CCDM encoding-decoding system, deeply analysed in [4]. Binary constant composition codes correspond to *m-out-of-n* codes, which are in general a special binary case of non-binary CCDM codes [3]. Furthermore, a necessary inequality in order to find the minimum m value that guarantees the one-to-one correspondence between input and output is represented by

$$m_{\min} = \min \left\{ m : \left\lfloor \log_2 \binom{n}{m} \right\rfloor \geq k \right\}. \tag{2.26}$$

This algorithm is useful to realize comparison between binary CCDM and binary syndrome distribution matching algorithm which is provided in the next chapter, to understand which are the limits of this type of distribution matching i.e., on short sequences.

Chapter 3

Syndrome Distribution Matcher

In this chapter, a description of the main algorithm is provided; the idea is inspired by [13], where a trellis decoder for LBCs was presented. The purpose is to implement a trellis-based algorithm acting as an efficient syndrome distribution matcher (SDM); to pursue this goal we first introduce the trellis representation of the parity-check matrix, and then we give a brief description of minimum distance and syndrome decoding. To conclude, we introduce the syndrome distribution matching and formalize the main algorithm of the Thesis, providing a deep description of it.

3.1 Trellis Representation of the LBC Cosets

Consider a LBC defined by its parity-check matrix $\boldsymbol{H}$ of size $(n-k) \times n$. Referring to (1.4), we can identify 2^{n-k} length $n-k$ vectors called syndromes, along with their corresponding 2^{n-k} cosets, each formed by 2^k length n vectors. The fundamental connection between syndrome and *noisy received word* is

$$\boldsymbol{s} = \boldsymbol{r}\boldsymbol{H}^{\mathrm{T}} = (\boldsymbol{c} + \boldsymbol{e})\boldsymbol{H}^{\mathrm{T}} = \boldsymbol{e}\boldsymbol{H}^{\mathrm{T}} \tag{3.1}$$

where $\boldsymbol{s}$ is the length $n-k$ syndrome and $\boldsymbol{r}$ is the length n noisy received word, which is the sum of the code word $\boldsymbol{c}$ and the *noise pattern* $\boldsymbol{e}$ added by the channel. Our goal is to represent the 2^{n-k} cosets by a trellis based on the matrix $\boldsymbol{H}$. To this aim, we write the parity-check matrix as a concatenation of n column vectors of length $n-k$, i.e.,

$$\boldsymbol{H} = [\boldsymbol{h}_1\ \boldsymbol{h}_2\ \boldsymbol{h}_3 \ldots \boldsymbol{h}_n]. \tag{3.2}$$

We define the *partial noisy received word* $\boldsymbol{r}(t)$ as

$$\boldsymbol{r}(t) = [r_1\ r_2 \ldots r_t] \tag{3.3}$$

which consists of the first t entries of the noisy received word $\boldsymbol{r}$. Next, we define the *partial syndrome* $\boldsymbol{s}(t)$ by

$$\boldsymbol{s}(t) = \sum_{i=1}^{t} r_i \boldsymbol{h}_i^{\mathrm{T}} = \sum_{i=1}^{t-1} r_i \boldsymbol{h}_i^{\mathrm{T}} + r_t \boldsymbol{h}_t^{\mathrm{T}} = \boldsymbol{s}(t-1) + r_t \boldsymbol{h}_t^{\mathrm{T}}. \tag{3.4}$$

Note that

$$\boldsymbol{s}(n) = \sum_{i=1}^{n} r_i \boldsymbol{h}_i^{\mathrm{T}} = \boldsymbol{r}\boldsymbol{H}^{\mathrm{T}} = \boldsymbol{s} \tag{3.5}$$

that is, we can calculate the syndrome $\boldsymbol{s}$ recursively via (3.4). Now we define the trellis, considering its evolution along the dimension of n-tuples thought as time T; we can identify each step with t, which takes values $t = 1, 2, \ldots n$. Let $\boldsymbol{s}(t)$ be the generic state; this can take 2^{n-k} values due to the length of each $\boldsymbol{h}_t$ column. Transitions are defined by $r_t \boldsymbol{h}_t^{\mathrm{T}}$, which take two values according to $r_t = 0, 1$; so, in the binary case, from each active state at step t, two edges start and reach two distinct states at step $t+1$. For completeness, we say that at step $t = 0$ we have only one state, corresponding to the all-zeros $(n-k)$-tuple. Along the algorithm, all the steps except the last have states which represent partial syndromes $\boldsymbol{s}(t)$, as follow from (3.4), while at step n we can identify all *final syndromes* $\boldsymbol{s}(n)$, by (3.5). The paths ending in the same state form a coset and for each final state there are 2^k distinct paths reaching it, by Section 1.4. Moreover, if we are interested only in the

LBC code book, we can remove all paths that do not end in the all-zeros state, corresponding to reach the all-zeros syndrome; this particular choice reflects (1.10).

Example 3.1.1. Consider the binary $(5, 3)$ LBC from [13] with parity-check matrix:

$$\boldsymbol{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix} = [\boldsymbol{h}_1 \ \boldsymbol{h}_2 \ \boldsymbol{h}_3 \ \boldsymbol{h}_4 \ \boldsymbol{h}_5]. \tag{3.6}$$

Observing the matrix, we know that the number of steps is equal to $n = 5$ and the number of states at each step is $2^{n-k} = 2^2 = 4$. So we can immediately draw all states. We know that up to step n, each state represents a partial syndrome and we can connect the states by the recursive formula (3.4). In step 5, each final syndrome $\boldsymbol{s}(n)$ is reached by 2^k paths $\boldsymbol{r}$, which fulfil (3.1) i.e.,

$$\boldsymbol{r}\boldsymbol{H}^{\mathrm{T}} = \boldsymbol{s}(n); \tag{3.7}$$

the total number of paths considering all the cosets is equal to $2^k \times 2^{n-k} = 2^n = 2^5 = 32$, which is in line with the previous description. In Figure 3.1 we provide the complete trellis, considering all the cosets paths where red arrows identify $r_i = 1$, while blue arrows $r_i = 0$; Furthermore, in Figure 3.2 we provide its expurgated version, representing the LBC code book with parity-check matrix $\boldsymbol{H}$.

3.2 Minimum Distance and Syndrome Decoding

Consider a code word $\boldsymbol{c}$ sent through a noisy channel; its noisy version $\boldsymbol{r}$ is received at the decoder. Consider a binary symmetric channel (BSC), where all the elements of $\boldsymbol{r}$ are binary. The goal is to make a decision $\hat{\boldsymbol{c}}$ equal to $\boldsymbol{c}$ by *minimum distance decoding*, which minimizes the Hamming distance between the noisy vector $\boldsymbol{r}$ at the decoder and the code words $\boldsymbol{c} \in \mathcal{C}$. We assume that

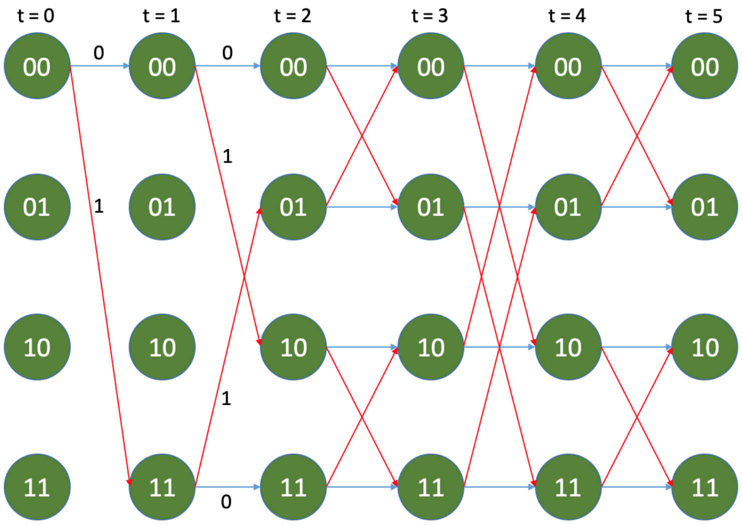

Figure 3.1: Complete trellis, representing the LBC cosets.

the BSC *crossover probability* p, which is the error probability through the channel, is strictly less than $\frac{1}{2}$. The objective is to minimize the probability of decision error after the channel; we can do this by maximum likelihood (ML) decoding [8, Sec. 1.4.3], which in this specific case $\left(\text{with } p < \frac{1}{2}\right)$ is equivalent to minimum distance decoding as shown next. Let us define the input X, output Y and noise Z, where $Y = X + Z$; the channel is described by the *likelihood*

$$P_{Y|X}(y|x) = P_Z(y - x) \tag{3.8}$$

with $P_Z(0) = 1 - p$ and $P_Z(1) = p$. Therefore, defined the *Hamming weight* $w_\mathrm{H}(\boldsymbol{x})$ as the number of 1s of a generic tuple $\boldsymbol{x}$, the *Hamming distance* $d_\mathrm{H}(\boldsymbol{c}, \boldsymbol{r})$ between $\boldsymbol{c}$ and $\boldsymbol{r}$ is described by

$$d_\mathrm{H}(\boldsymbol{c}, \boldsymbol{r}) = w_\mathrm{H}(\boldsymbol{c} - \boldsymbol{r}) = \sum_{i=1}^{n} \mathbb{1}\left(c_i - r_i \neq 0\right). \tag{3.9}$$

Over a memory-less channel, we define the likelihood $P^n_{Y|X}(\boldsymbol{r}|\boldsymbol{c})$ by

$$P^n_{Y|X}(\boldsymbol{r}|\boldsymbol{c}) = \prod_{i=1}^{n} P_{Y|X}(r_i|c_i) = \prod_{i=1}^{n} P_Z(r_i - c_i). \tag{3.10}$$

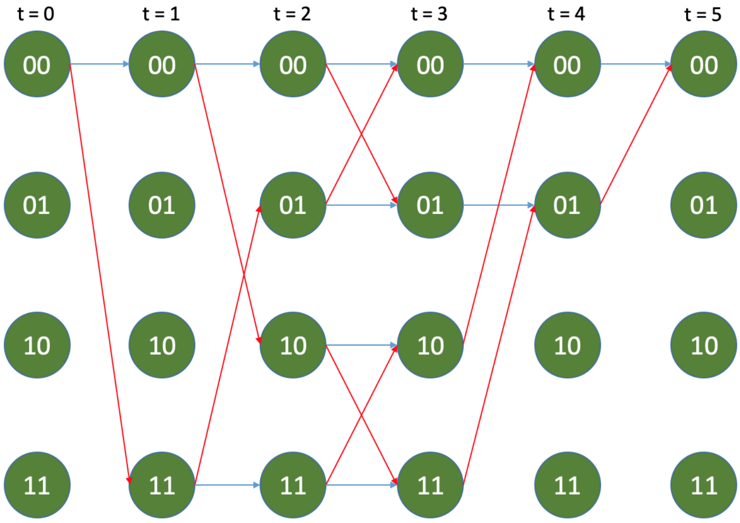

Figure 3.2: Code book trellis, representing the linear (5,3) code by $\boldsymbol{H}$.

In this specific case of a BSC (p) with $p < \frac{1}{2}$, ML decision $\hat{\boldsymbol{c}}$ at the decoder consists of

$$\begin{aligned}\hat{\boldsymbol{c}} &= \operatorname*{argmax}_{\boldsymbol{c}\in\mathcal{C}} P^n_{Y|X}(\boldsymbol{r}|\boldsymbol{c}) = \operatorname*{argmax}_{\boldsymbol{c}\in\mathcal{C}} (1-p)^{n-d_{\mathrm{H}}(\boldsymbol{c},\boldsymbol{r})} p^{d_{\mathrm{H}}(\boldsymbol{c},\boldsymbol{r})} \\ &= \operatorname*{argmax}_{\boldsymbol{c}\in\mathcal{C}} (1-p)^n \left(\frac{p}{1-p}\right)^{d_{\mathrm{H}}(\boldsymbol{c},\boldsymbol{r})} = \operatorname*{argmax}_{\boldsymbol{c}\in\mathcal{C}} \left(\frac{p}{1-p}\right)^{d_{\mathrm{H}}(\boldsymbol{c},\boldsymbol{r})}.\end{aligned} \tag{3.11}$$

Hence, we can see that, with $p < \frac{1}{2}$, maximizing the likelihood is equivalent to minimizing the Hamming distance d_{H}. Then, we can represent the minimum distance decoding by

$$\hat{\boldsymbol{c}} = \operatorname*{argmin}_{\boldsymbol{c}\in\mathcal{C}} d_{\mathrm{H}}(\boldsymbol{c},\boldsymbol{r}) = \operatorname*{argmin}_{\boldsymbol{c}\in\mathcal{C}} w_{\mathrm{H}}(\boldsymbol{r}-\boldsymbol{c}). \tag{3.12}$$

To proceed, we exploit syndromes to make ML decisions of the transmitted code word $\boldsymbol{c}$. In fact, we can adopt syndrome decoding, which exploits (3.1). Given a parity-check matrix, for each syndrome $\boldsymbol{s}$, 2^k sequences $\boldsymbol{r}'$ fulfil $\boldsymbol{r}'\boldsymbol{H}^{\mathrm{T}} = \boldsymbol{s}$ and within this ensemble we can always identify the minimum Hamming weight error pattern $\boldsymbol{e}$, called *coset leader*, by

$$\boldsymbol{e}(\boldsymbol{s}) = \operatorname*{argmin}_{\boldsymbol{r}':\, \boldsymbol{r}'\boldsymbol{H}^{\mathrm{T}}=\boldsymbol{s}} w_{\mathrm{H}}(\boldsymbol{r}'). \tag{3.13}$$

Then, we are able to construct a LUT with the 2^{n-k} syndromes and each corresponding coset leader $\boldsymbol{e}$. Next, knowledge of the syndrome $\boldsymbol{s}$ associated with received $\boldsymbol{r}$, leads us to the decoding rule

$$\hat{\boldsymbol{c}} = \boldsymbol{r} + \boldsymbol{e}\left(\boldsymbol{r}\boldsymbol{H}^{\mathrm{T}}\right) \tag{3.14}$$

that is, the LUT has memory a number of $(n-k)$-tuples equal to 2^{n-k}.

3.3 Syndrome Distribution Matching

Consider the syndrome decoding scheme in Figure 3.3. If we isolate the system formed by the LUT, which accepts the syndrome as input and produces an error pattern as output, this could be regarded as an SDM. In syndrome decoding, the output $\boldsymbol{e}$ is always the minimum Hamming weight sequence linked to the specific syndrome coset as shown in (3.13); nevertheless, a generalization based on a cost function is possible, which changes the one-to-one relation between syndromes and cosets leaders, keeping the LUT structure unaltered.

Performing distribution matching with a LUT works in principle for any binary code, but the table with 2^{n-k} entries may be too large depending on the case, leading to complexity issues. Moreover, we can affirm that SDM on trellis described in Section 3.2 is not much less complex than the LUT implementation since it has 2^{n-k} states at each step. We therefore consider a different approach, which allows us achieving a gain in specific scenarios.

3.4 SDM Algorithm

The purpose of SDM is to map the input bits onto a sequence that minimizes a cost function. Considering the syndrome $\boldsymbol{s} \in \mathbb{F}_2^{n-k}$ and the parity-check matrix $\boldsymbol{H} \in \mathbb{F}_2^{(n-k)\times n}$, let the shaped sequence be $\boldsymbol{r} \in \mathbb{F}_2^n$. We can create the trellis by successively accounting for the row constraints of the parity-check matrix, which is a complementary approach compared to [13];

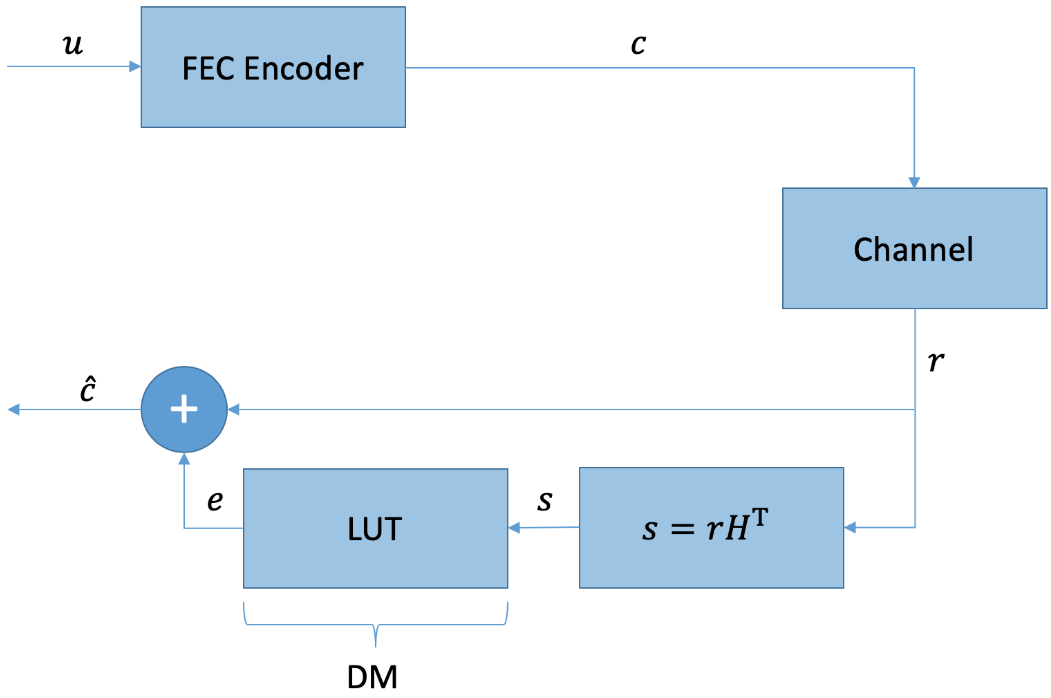

Figure 3.3: Syndrome decoding system with emphasis on DM.

the generic row constraint imposed by $\boldsymbol{H}$ may be named $\boldsymbol{h}_i$. The mathematical model of the problem is provided below for $\boldsymbol{H}$, $\boldsymbol{s}$ and a generic cost function $f(\cdot)$:

$$\boldsymbol{r} = \operatorname*{argmin}_{\boldsymbol{r}' \in \mathbb{F}_2^n} f(\boldsymbol{r}');$$
$$\text{Subject to } \; s_i = \boldsymbol{r}'\boldsymbol{h}_i^{\mathrm{T}}, \quad i = 1, 2, \ldots, n-k.$$

The number of steps is $n-k$ because of the syndrome length, and the number of states at each step is 2^n. However, we can reduce the actual number of states significantly by observing that a row $\boldsymbol{h}_i$ of $\boldsymbol{H}$ only constrains the entries of $\boldsymbol{r}$ where $\boldsymbol{h}_i$ is non-zero. Importantly, each state reached at a certain step is a *pre-shaped sequence*, fulfilling only the constraints considered up to that step. Let the initial state $t = 0$ be the all-zeros sequence; the main idea is going through the trellis exploiting the non-zeros entries in the rows of the parity-check matrix at the respective step of the algorithm, until all $n-k$

entries of the syndrome have been accounted for. Let *flexible ones* $n_1(t)$ be the number of 1*s* in the t-th row of $\boldsymbol{H}$ that have not yet been visited by one of the previous iterations. During each step, we define a matrix $\boldsymbol{X}(t)$ of size $2^{n_1(t)} \times n$

$$\boldsymbol{X}(t) = \begin{bmatrix} \boldsymbol{x}_1(t) \\ \boldsymbol{x}_2(t) \\ \vdots \\ \boldsymbol{x}_{2^{n_1(t)}}(t) \end{bmatrix} \tag{3.15}$$

formed by all the possible n-tuples as row vectors, which have all-zero elements except in the positions corresponding to the current flexible ones, where we can observe in each row of $\boldsymbol{X}(t)$ one of the $2^{n_1(t)}$ binary $n_1(t)$-tuples. Let the generic state at step $t-1$ be called $\boldsymbol{r}(t-1)$. Let s_t be the value that the input syndrome assumes at the step t; in order to find all valid subsequent states, we can exploit the following check relation on any $\boldsymbol{x}_i(t)$, $i = 1,\ 2, \ldots, 2^{n_1(t)}$,

$$\boldsymbol{X}_{\text{valid}}(t) = \left\{\boldsymbol{x}_i(t) \middle| \left[\boldsymbol{r}(t-1) + \boldsymbol{x}_i(t)\right] \boldsymbol{h}_t^T = s_t,\ i = 1,\ 2, \ldots, 2^{n_1(t)}\right\}. \tag{3.16}$$

We can create the branches between the state at step $t-1$ and all the following $2^{n_1(t)-1}$ states at step t. In fact, named the elements corresponding to the $n_1(t)$ positions by *flexible bits*, if we want to fulfil a constraint with the modulo-2 sum of binary elements, we have always half of the possible combinations of the considered number of flexible bits. Hence, the number of states at step t reached by the previous one is always $\frac{2^{n_1(t)}}{2} = 2^{n_1(t)-1}$; then the successors can be evaluated by

$$\boldsymbol{r}(t) = \left\{\boldsymbol{r}(t-1) + \boldsymbol{x}_i(t) | \boldsymbol{x}_i(t) \in \boldsymbol{X}_{\text{valid}}(t)\right\}. \tag{3.17}$$

Let the flexible ones of the previous constraint respecting the current (t-th) constraint be named *fixed ones* and the corresponding elements be *fixed bits*. After a certain number of iterations, depending on how the parity-check matrix is formed, the number of flexible ones becomes equal to zero

($n_1(t) = 0$), and the new step of the algorithm starts with all elements being fixed bits. Then two phases can be identified: in the first phase the iterative rule (3.16) is used, while in the second phase, states $\boldsymbol{r}(t-1)$ that fulfil constraint t are copied as states of the following step. The transition from the first phase to the second phase occurs according to the considered parity-check matrix. Between the two phases, the maximum number of states (MNS) can be evaluated; let the number of row constraints considered so far be called number of rows with flexible ones (NRFO). We define

$$\mathrm{MNS} = \prod_{i=1}^{\mathrm{NRFO}} 2^{n_1(i)-1} = \frac{1}{2^{\mathrm{NRFO}}} \prod_{i=1}^{\mathrm{NRFO}} 2^{n_1(i)} \tag{3.18}$$

which gives us the number of states that fulfil the first NRFO constraints; from that, we use all the remaining rows that have not yet been considered in order to expurgate the paths. At each further step t, we keep only the previous states $\boldsymbol{r}(t-1)$ with respect to (3.16), considering, due to $n_1(t) = 0$, the only $\boldsymbol{x}_1(t)$ as the all-zeros sequence; in the end, the algorithm always gives us 2^k paths, which is consistent with Section 1.4. In order to exploit the real gain of this algorithm, we can construct the parity-check matrix $\boldsymbol{H}$ introducing flexible $1s$ until its last row, then, we can end the SDM without any need to expurgate paths, avoiding the second phase; this is the only way to have MNS equal to 2^k; an example is provided below.

Example 3.4.1. Consider $k = 3$ and $n = 9$, let the parity-check matrix be

$$\boldsymbol{H} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \tag{3.19}$$

that is, we want to show that in this case we can obtain an SDM with a trellis formed by $2^k = 8$ states at each of the $n-k$ steps compared to the canonical trellis shaping with $2^{n-k} = 64$ states at each of the n steps. Nevertheless, this represents a gain since $k < n - k$. Without analysing the exact expression of the states, it is straightforward to evaluate the number of states involved in each iteration, counting the number of flexible ones introduced by each row. Let $\boldsymbol{g}$ be the vector containing the $n - k$ values $n_1(t)$, for all $1 \leq t \leq 6$ i.e.,

$$\boldsymbol{g} = [n_1(1)\ n_1(2)\ n_1(3)\ n_1(4)\ n_1(5)\ n_1(6)] = [2\ 2\ 2\ 1\ 1\ 1]. \tag{3.20}$$

Following (3.16) and (3.17) and since at each step the number of reached states by a previous one is equal to $2^{n_1(t)-1}$, we can evaluate the number of states, named $\mathrm{ns}(t)$ at each step t, starting from the first single state at $t = 0$ with $\mathrm{ns}(0) = 1$:

$$t = 1 : \mathrm{ns}(1) = \mathrm{ns}(0)2^{n_1(1)-1} = 2; \tag{3.21}$$

$$t = 2 : \mathrm{ns}(2) = \mathrm{ns}(1)2^{n_1(2)-1} = 4; \tag{3.22}$$

$$t = 3 : \mathrm{ns}(3) = \mathrm{ns}(2)2^{n_1(3)-1} = 8; \tag{3.23}$$

$$t = 4 : \mathrm{ns}(4) = \mathrm{ns}(3)2^{n_1(4)-1} = 8; \tag{3.24}$$

$$t = 5 : \mathrm{ns}(5) = \mathrm{ns}(4)2^{n_1(5)-1} = 8; \tag{3.25}$$

$$t = 6 : \mathrm{ns}(6) = \mathrm{ns}(5)2^{n_1(6)-1} = 8. \tag{3.26}$$

Then, we proved that the MNS involved along the algorithm steps is exactly $2^k = 2^3 = 8$.

An important issue is that the solution of a certain step is composed of the sequences that fulfil all the constraints up to that step; this gives us the successive estimations idea of the algorithm. Next, an additional example is provided to clarify the states structure.

Example 3.4.2. Considering the parity-check matrix with $k = 3$ and $n = 8$ provided in Figure 3.4, we can represent the partial evolution of the trellis ending in one of the possible shaped sequences, considering only one branch at each step. This is useful to understand how the algorithm affects the states and the transitions along the graph. To pursue the goal, we use the

$$\boldsymbol{H} = \left.\overbrace{\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & \underline{1} & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \underline{1} & 0 & 0 & 0 & 1 & 0 \\ \underline{1} & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}^{\boldsymbol{r}}\right\} \boldsymbol{s}$$

Figure 3.4: Parity-check matrix, $k = 3$, $n = 8$.

syndrome

$$\boldsymbol{s} = [1\ 0\ 0\ 1\ 1]. \tag{3.27}$$

Following the SDM description provided in this section, we can evaluate branches and states analysing sequentially the row constraints of the parity-check matrix. First, we identify the flexible ones in each row by blue circles; these 1s become fixed ones as from the consecutive row constraint and they are underlined with a red line below each of them. This representation is reflected directly on the structure of the trellis; as it is shown in Figure 3.5, at each step we use all possible combinations of bits corresponding to the positions of the blue circles in the current constraint of $\boldsymbol{H}$, keeping fixed all bits corresponding to the positions already analysed due to the flexible ones of the previous constraints, identified by the red line below them. Let the generic state be represented by $\boldsymbol{r}'$; to emphasise the difference between flexible and fixed bits in each state, we use a line below each binary element, blue for flexible bits and red for fixed bits. Hence, the trellis representation is provided below. Starting from the all-zeros state, if we analyse the first row constraint we can observe that the flexible ones are the second and fourth

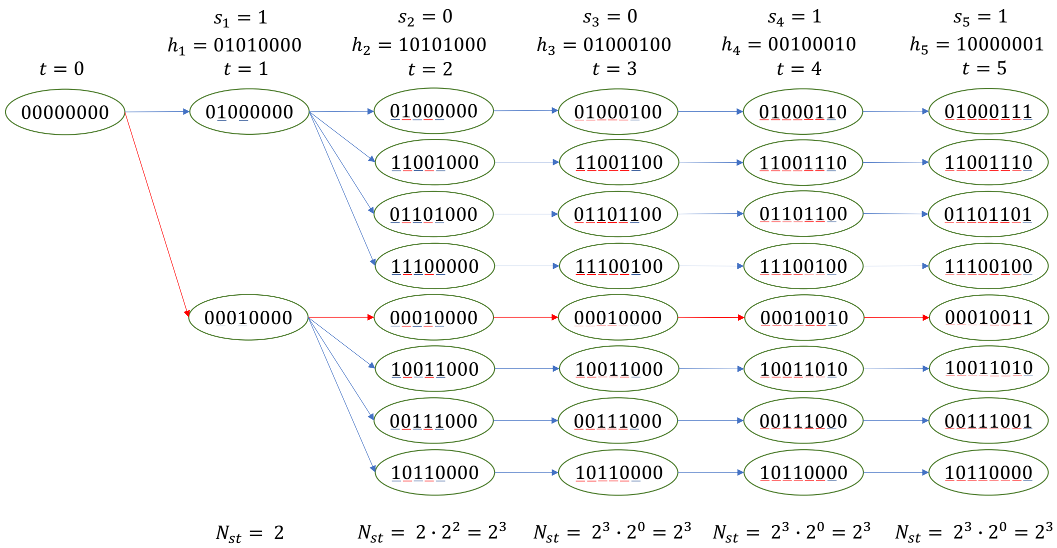

Figure 3.5: Trellis evolution.

elements; the only combinations of these two bits that provides states $\boldsymbol{r}'$ which fulfil

$$s_1 = \boldsymbol{r}'\boldsymbol{h}_1^{\mathrm{T}} \tag{3.28}$$

are the two states shown in Figure 3.5 at step $t = 1$. Following this idea, we can derive all the other sections of the trellis reaching all the final states. In the end, we must choose 1 out of 8 sequences; in fact, the preferred sequence is then selected among the 8 candidates following a specific cost function and in this case we try to minimize the number of 1s. We can identify between the two final candidates with minimum Hamming weight equal to 3 (fifth and eight states at step $t = 5$) the sequence obtained following the red arrows path represented by

$$\boldsymbol{r}' = [0\ 0\ 0\ 1\ 0\ 0\ 1\ 1] \tag{3.29}$$

which fulfils all the $\boldsymbol{H}$ constraints, solving the DM problem. In this specific example, the maximum number of states at each step is equal to

$$2^3 = 2^k < 2^{n-k} = 2^5; \tag{3.30}$$

hence, if $k < n - k$, we have a gain in terms of number of states, due to the particular structure of the parity-check matrix which introduces flexible ones until the last row.

A flow chart of the SDM is provided in Figure 3.6. It explains the main steps emphasising the two phases of the algorithm. Next, another example is discussed, showing a worse situation with respect to the previous one. In the new example, the number of states that are involved along operation is larger than 2^k because the matrix does not introduce flexible ones until the $(n-k)$-th constraint.

Example 3.4.3. Given $k = 5$ and $n = 10$, a further example is introduced. Let the following $\boldsymbol{H}$ and $\boldsymbol{s}$ be possible inputs of the SDM:

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}; \tag{3.31}$$

$$\boldsymbol{s} = [1\ 0\ 1\ 0\ 1]. \tag{3.32}$$

If we start with the first row, permutations of seven $1s$ can be observed; then, we can find the first $2^{n_1(1)} = 2^7$ pre-shaped n-tuples. Within this, only half of the ensemble fulfils the current row constraint in the binary case, because the sum of elements of a state must be equal to the corresponding bit of the syndrome. Following the idea of the flexible ones at each iteration, after 3 rows we can observe that a 1 has already been analysed in all possible n positions; in this case, being NRFO = 3, we have

$$\text{MNS} = \frac{1}{2^3} \prod_{i=1}^{3} 2^{n_1(i)} = \frac{2^7 \cdot 2^0 \cdot 2^3}{2^3} = 2^7 = 128. \tag{3.33}$$

With regard to the paths, we have to consider that only the first NRFO rows have already been fulfilled; indeed, the analysis of the remaining rows is needed. It can be shown that with the following rows a certain number of paths is expurgated to respect all the parity-check matrix constraints. In the end, the number of possible output sequences (paths), which correspond to states that are involved in the last iteration, is always equal to 2^k.

3.5 Considerations

To sum up, we can compare the SDMs based on the LUT reviewed in Section 3.3 with the algorithm described in Section 3.4. The first one is memory expensive due to the memory storage of all the possible output sequences, in particular 2^{n-k} length n-tuples; but, owing to the use of a LUT, the SDM is very time efficient. On the other hand, the second approach is less memory expensive, because we store only 2^k length n pre-shaped sequences at each step overwriting the previous, but much more time expensive because we need to evaluate runtime at each SDM use the output sequence given a specific input.

To conclude, we can say that through the trellis-based SDM algorithm, a certain number of shaped sequences can be obtained as outputs; the choice of the suitable path depends on the cost function which gives the desired distribution. Under certain conditions, we can obtain a relevant gain in terms of memory reduction, decreasing the number of states involved along the algorithm.

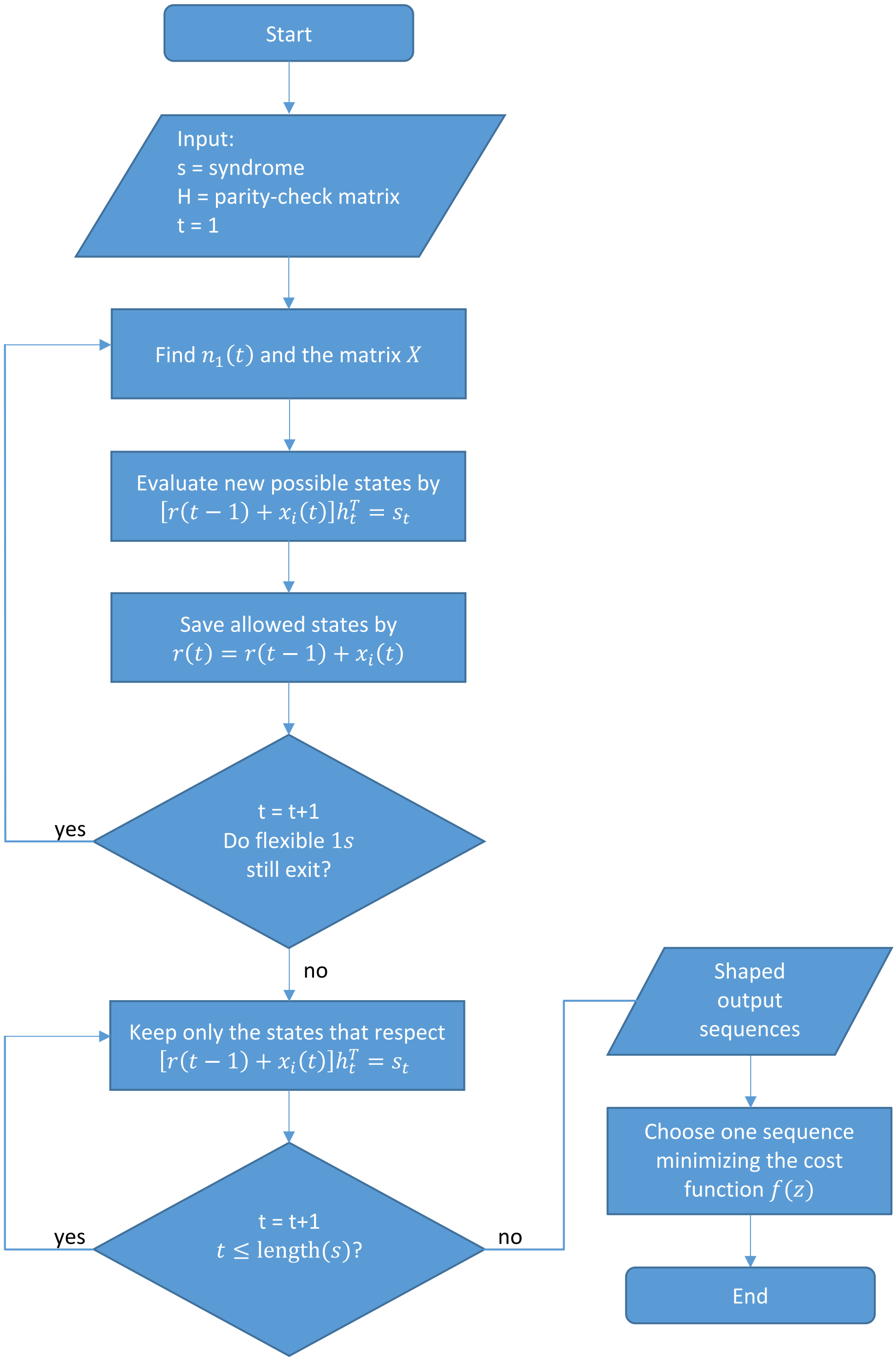

Figure 3.6: SDM algorithm flow-chart.

Chapter 4

SDM based on Systematic Random Matrix

In this chapter, an analysis of the SDM algorithm based on a systematic random (SR) matrix is provided. First of all, we have to explain the choice of the matrix used to implement the SDM and then move on to the resolution and results of the problem, using the algorithm described in Section 3.4.

4.1 SR Matrix

In order to realize the matrix $\boldsymbol{H}$, which is necessary to implement the SDM, we analyse the parity-check matrices recovered by a particular criterion explained hereafter. It is possible to create a matrix type which reduces the maximum number of states of the algorithm to the fixed value 2^k, regardless of the matrix and equal to the number of total paths related to the coset. As emphasised in Section 3.4, this gain is reachable only if we introduce flexible ones until the last row constraint. What we realized is creating a k dimensional identity matrix on the right and other k columns on the left by random generation with IUD bits. Hence, given for example $k = 4$ and $n = 8$, a possible parity-check matrix in systematic form is shown below.

$$\boldsymbol{H} = \begin{array}{r} \text{Constraint} \\ \\ \\ \\ \\ \end{array} \left[\underbrace{\begin{matrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{matrix}}_{\text{IUD uniformly distributed}} \middle| \underbrace{\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}}_{\text{Identity Matrix}} \right]$$

Figure 4.1: SR parity-check matrix, $k = 4$, $n = 8$.

Remark. *We point out the analysis and results of a particular SDM which tries to reach entropy equal to $\frac{1}{2}$; then, $n - k = k$ due to the chosen rate. Therefore, we do not focus on the number of states, as in Section 3.4 (because $2^{n-k} = 2^k$). Rather, we emphasise an advantage in particular scenarios in term of reachable output sequence entropy.*

4.2 SR-based SDM Results with Rate $\frac{1}{2}$

To provide results, the analysis has been done using SR matrices with rate $\frac{1}{2}$; we consider different matrix sizes, constructing in each case an appropriate parity-check matrix in the way described in the previous section. The choice of the matrix is not trivial, because the results are related to the constraints used along the rows. What we have done is, until input length is lower than or equal to 13, choosing the best in terms of output sequence entropy within an ensemble of 10 parity-check matrices; in all the other cases, we used the first generated matrix which follows the criterion above, without a specific research within an ensemble because of the higher cost in terms of time. Indeed, from an operative point of view, given the matrix $\boldsymbol{H}$, the algorithm performs the distribution matching on all possible sequences with a number of elements equal to the input length k; then, the cardinality of this ensemble is equal to 2^k. Comparing the results of this new approach using as benchmark the best obtainable CCDM entropy, both shown in Table 4.1, is relevant. Let m_{CCDM} be the number of $1s$ contained in the CCDM output sequence, referring to the empirical distribution. In our case, as we are interested in entropy that tends to $\frac{1}{2}$, we obtain the CCDM results using the lowest value of m_{CCDM} which fulfil the inequality(2.26), assuring the one-to-one correspondence between the DM input and output. We can observe from the tables that in these cases the SDM values are always better or at least comparable with respect to the CCDM. This means that, for very short output length, SR-based SDM is in line with or even better than CCDM. Plots containing the comparison between the table results (Figure 4.2) and the weighted spectra of the worst and best treated cases (Figure 4.3 and Figure 4.4) are provided below. From the first plot, we can observe the results represented by two entropy curves, which point out the good behaviour of the SR-based SDM on short sequences compared to CCDM. Besides, the weight spectra remark the shift to the left of the occurrences distribution with the increase of the sequences length; this means the decrease of the entropy on averaged PMF. To sum up, whenever there was interest in short sequences of length for which SDM gives a better result compared to CCDM, SR approach would be a reasonable choice. It is important to remind that the results are values obtained on an average PMF while CCDM gives an entropy based on a fixed empirical distribution. At the same time with the latter we have to take into account the distribution quantization problem of real systems [11, Sec. 9.3]; this aspect occurs when we are interested in an empirical distribution which is not n-type, hence we have to make an approximation to use the CCDM. A last observation is that with CCDM, we do not have an observable dependence between input/output length and entropy; with regard to SDM, the increase of the sequence length causes a

k_{SDM}	n_{SDM}	$\mathbb{H}(P_X)$	m_{CCDM}	$\mathbb{H}$
4	8	0.6873	2	0.8113
5	10	0.6525	2	0.7219
6	12	0.6313	2	0.6500
7	14	0.6269	3	0.7496
8	16	0.6202	3	0.6962
9	18	0.6145	3	0.6500
10	20	0.6026	3	0.6098
11	22	0.5988	4	0.6840
12	24	0.5949	4	0.6500
13	26	0.5911	3	0.6194
14	28	0.5883	4	0.5917
15	30	0.5865	5	0.6500
16	32	0.5846	5	0.6253

Table 4.1: SR-based results, rate $\frac{1}{2}$.

decrease of the entropy evaluated on the averaged PMF.

To conclude, SDM based on SR matrices is useful if we deal with short length sequences. For long ones, we encounter severe complexity issues that prevent us from obtaining results due to the prohibitive computation time and memory expense.

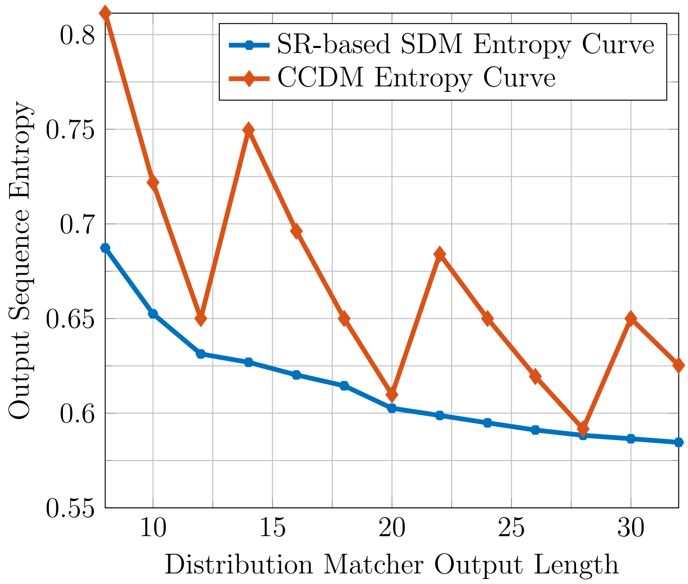

Figure 4.2: Entropy curves comparison, rate $\frac{1}{2}$.

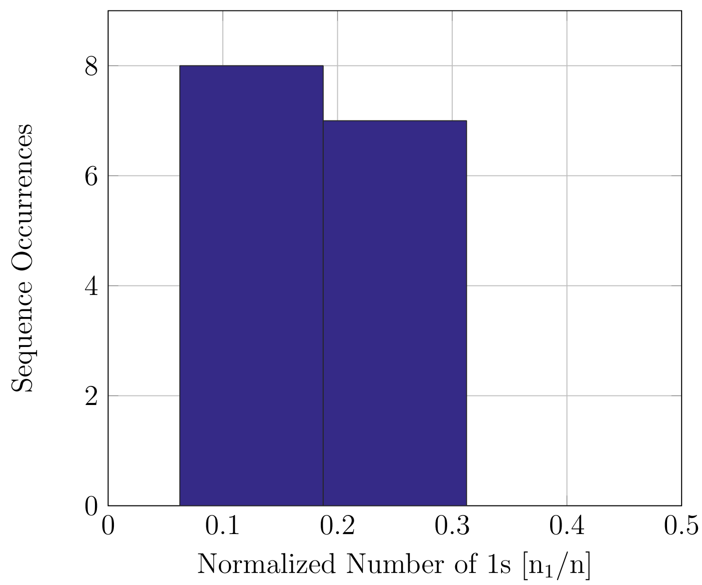

Figure 4.3: SR-based SDM, $n = 8$.

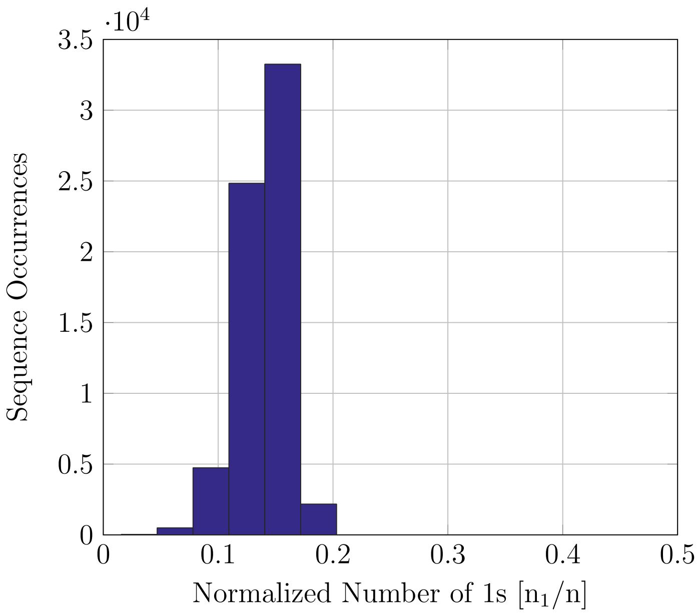

Figure 4.4: SR-based SDM, $n = 32$.

Chapter 5

SDM based on Low Density Diagonal Matrix

In this chapter, we try to tackle the distribution matching problem using the SDM based on a low density diagonal (LDD) matrix. After having clarified the matrix choice, a brief introduction to the modified version of the SDM described in Chapter 3 is provided; whereupon, we show several results, obtained using the best CCs generators shown in [10, Sec. 12.3] as base for the syndrome distribution matching, in order to create a precise parity-check matrix. Along this chapter the analysis and results are provided by $\frac{1}{2}$-rate SDM, having an output sequence entropy that tends to the rate.

5.1 LDD Matrix

We tried to find out the best matrix in term of output entropy using different approaches. By exhaustive search we recovered the best construction of $\boldsymbol{H}$ by following the same structure of CCs described in Section 1.5, being the one that gives the best results. To pursue the goal, we take two generators expressed by octal representation, for example $[\boldsymbol{g}_1, \boldsymbol{g}_2] = [5, 7]$, and we use these to construct the parity-check matrix such as (1.32). The only difference is that in this case we use a short CC to recover the $\boldsymbol{H}$, which has finite dimension, while the one of CC in Section 1.5 has a semi-infinite dimension; this is necessary to treat the syndrome problem as in Chapter 4. We then have

$$\boldsymbol{H} = \begin{bmatrix} 1 & 1 & & & & & & & & & & \\ 1 & 0 & 1 & 1 & & & & & & & & \\ 1 & 1 & 1 & 0 & 1 & 1 & & & & & & \\ & & 1 & 1 & 1 & 0 & & & & & & \\ & & & & 1 & 1 & & & & & & \\ & & & & & & \ddots & & & & & \\ & & & & & & & & 1 & 1 & & \\ & & & & & & & & 1 & 0 & 1 & 1 \\ & & & & & & & & 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}. \tag{5.1}$$

Let the order of a generator be the position of the rightmost 1 in its binary vector representation. That is, let n1 be the order of $\boldsymbol{g}_1$ and let n2 be the order of $\boldsymbol{g}_2$. Considering a specific index equal to

$$\mathrm{I} = \max\left(\mathrm{n1}, \mathrm{n2}\right) \tag{5.2}$$

that is, given

$$\boldsymbol{g}_1 = [1\ 0\ 1], \quad \boldsymbol{g}_2 = [1\ 1\ 1] \tag{5.3}$$

so that n1 $= 3$ and n2 $= 3$, we have

$$\mathrm{I} = 3; \tag{5.4}$$

we can define a vector called *rule* of length Rl equal to

$$\mathrm{Rl} = 2\mathrm{I} = 6 \tag{5.5}$$

provided by the I-th row of the parity-check matrix; in this case we can affirm that

$$\mathrm{R} = [1\ 1\ 1\ 0\ 1\ 1]. \tag{5.6}$$

This rule permits us to optimize the algorithm of Section 3.4 on this diagonal matrix, obtaining a reasonable complexity during the resolution of the problem; furthermore, the rule length is directly related to the number of trellis states at each iteration, maximised by $2^{\mathrm{Rl}-1}$. To conclude, we call this type of $\boldsymbol{H}$ LDD matrix, low density because we have the majority of the elements equal to 0 and diagonal due to the structure described above.

5.2 LDD-based SDM Algorithm

Building on the algorithm described in 3.4, a new version of SDM has been realized, trying to exploit the LDD structure of the matrix used for syndrome distribution matching. First of all, as we emphasised in the previous section, a rule that occurs cyclically shifted at each row may be found; hence, it is possible to predict all the transitions between two following steps of the SDM, which are always the same along the entire the algorithm depending on the binary value of the syndrome element. Therefore, storing all the possible transitions in a LUT may be feasible under a practical complexity limit provided below, avoiding the estimation of the entire branches ensemble at each new step of the SDM. Furthermore, the LDD structure is fundamental because each step of the trellis, due to the short Rl, involves only a limited number of the output sequence elements. Otherwise, this problem with parity-check matrix size used below would be infeasible due to the huge number of states involved at each step of the trellis.

In this precise type of optimization, we may affirm that the SDM is considerable a Viterbi distribution matcher (VDM), since we store weight information in metric functions associated with each state and the *survivor idea* is needed to reduce the complexity of the algorithm [14]. Referring to the latter, if two or more starting points at number of steps t reach the same final state at step $t + 1$ (representation of Section 3.1), following a specific cost function we can be able to maintain only one of them; in the analysed case, the only survivor of each state is always the one with the minimum associated weight, due to the fact that we are still interested in sequences of which entropy tends to $\frac{1}{2}$. Passing through the algorithm, two different phases may be recovered, given by the fact that a duplication of the number of states occurs at each iteration. We start from the all-zeros state, having a maximum number of states at a certain step equal to $2^{\mathrm{Rl}-1}$, due to the same reason provided in Chapter 3 that only half of the possible combinations fulfil the constraint in the binary algebra. Along the first 2I steps we deal with the *transient phase*, without reaching the upper bound of the number of states; after that, the *steady phase* is reached and assured until the end, using the survivor idea, dealing with $2^{\mathrm{Rl}-1}$ states at each iteration.

To sum up, we can represent all the stored trellis information by the following analysis. Looking at the matrix construction in (5.1), an important observation is that the first two elements of the first row must be $1s$, to ensure a correct determination of the first two output sequence bits; in fact, to assure a CC rate equal to $\frac{1}{2}$, in each row we introduce two flexible bits on the right part of the rule, as we can see in (5.6), identified by n_{flex}. Hence, given a matrix $\boldsymbol{Y}$ (at the t-th step), the rows of which are n_{flex}-tuples coinciding with all the possible combinations of $n_{\mathrm{flex}} = 2$ bits as

$$\boldsymbol{Y}(t) = \begin{bmatrix} \boldsymbol{y}_1(t) \\ \boldsymbol{y}_2(t) \\ \boldsymbol{y}_3(t) \\ \boldsymbol{y}_4(t) \end{bmatrix}, \tag{5.7}$$

defined the generic states at step $t-1$ by $\boldsymbol{x}_{t-1}$ and at step t by $\boldsymbol{x}_t$, we can evaluate for each possible state at step $t-1$ the n_{flex} parts of the LUT, in this case 2, fulfilling, according to the t-th value of the syndrome s_t (1 or 0),

$$\boldsymbol{Y}_{\mathrm{valid}}(t) = \left\{\boldsymbol{y}_i(t) | [\boldsymbol{x}^{(3:\mathrm{end})}(t-1);\ \boldsymbol{y}_i(t)]\mathrm{R} = s_t,\ i = 1,2,3,4\right\}. \tag{5.8}$$

In fact, each possible state, due to diagonal form of the matrix, is composed of the last $\mathrm{Rl} - n_{\mathrm{flex}} = 4$ bits of the previous state concatenated to one within the $n_{\mathrm{flex}} = 2$ possible combinations. So, to define the reachable states based on the current syndrome value, the following expression can be used

$$\boldsymbol{x}(t) = \left\{[\boldsymbol{x}^{(3:\text{end})}(t-1);\ \boldsymbol{y}_i(t)]|\boldsymbol{y}_i(t) \in \boldsymbol{Y}_{\text{valid}}(t)\right\}. \tag{5.9}$$

As already discussed, the shifted rule along the diagonal always guarantees the same possible branches between two steps and we can recast (5.8) and (5.9) using the generic 4-tuple $\boldsymbol{f}$ within the ensemble of $2^{\text{Rl}-n_{\text{flex}}} = 2^4$ possibilities instead of the generic $\boldsymbol{x}^{(3:\text{end})}(t-1)$ by

$$\boldsymbol{Y}_{\text{valid}}(t) = \{\boldsymbol{y}_i(t)|[\boldsymbol{f};\ \boldsymbol{y}_i(t)]\text{R} = s_t,\ i = 1,\ 2, \ldots, 2^{n_{\text{flex}}}\}. \tag{5.10}$$

and obtaining the state as

$$\boldsymbol{x}(t) = \{[\boldsymbol{f};\ \boldsymbol{y}_i(t)]|\boldsymbol{y}_i(t) \in \boldsymbol{Y}_{\text{valid}}(t)\}. \tag{5.11}$$

To proceed, stored the trellis branches, to perform distribution matching we have to exploit each element of the syndrome choosing the right connections between the states, using the survivor idea after the transient phase. The latter means that the expensive part of this algorithm is saving the trellis a priori, not the output sequence evaluation along the pre-stored graph. To conclude, after the resolution of the problem, we obtain as result $2^{\text{Rl}-1}$ paths with minimum weights within the ensemble formed by all 2^n possibilities.

5.3 LDD-based SDM Results with Rate $\frac{1}{2}$

In this section, all the results of the LDD-based SDM algorithm are provided, trying also in this case to reach an entropy as close as possible to $\frac{1}{2}$. First of all, a table with all the entropy values related to each pair of generators (in octal representation described in 1.1) of all the best CCs, with rate equal to $\frac{1}{2}$ is summed up [10, Section 12.3]. Successively, a plot representing a trade-off between number of states (memory) and entropy (performance) is shown in Figure 5.1; to conclude, the weight spectra of the worst case with $[\boldsymbol{g}_1, \boldsymbol{g}_2]$ = [3,1], the case study of Section 5.1 with $[\boldsymbol{g}_1, \boldsymbol{g}_2]$ = [5,7] and the best case with $[\boldsymbol{g}_1, \boldsymbol{g}_2]$ = [10627, 16765] are observable in Figure 5.2, Figure 5.3 and Figure 5.4.

Generators	Rl	$\mathbb{H}(P_X)$
$[\boldsymbol{g}_1, \boldsymbol{g}_2]$ = [3,1]	4	0.649664
$[\boldsymbol{g}_1, \boldsymbol{g}_2]$ = [5,7]	6	0.581192
$[\boldsymbol{g}_1, \boldsymbol{g}_2]$ = [13,17]	8	0.569718
$[\boldsymbol{g}_1, \boldsymbol{g}_2]$ = [27,31]	10	0.557369
$[\boldsymbol{g}_1, \boldsymbol{g}_2]$ = [53,75]	12	0.549736
$[\boldsymbol{g}_1, \boldsymbol{g}_2]$ = [117,155]	14	0.543494
$[\boldsymbol{g}_1, \boldsymbol{g}_2]$ = [247,371]	16	0.537001
$[\boldsymbol{g}_1, \boldsymbol{g}_2]$ = [561,753]	18	0.532301
$[\boldsymbol{g}_1, \boldsymbol{g}_2]$ = [1131,1537]	20	0.526552
$[\boldsymbol{g}_1, \boldsymbol{g}_2]$ = [2473,3217]	22	0.524961
$[\boldsymbol{g}_1, \boldsymbol{g}_2]$ = [4325,6747]	24	0.520311
$[\boldsymbol{g}_1, \boldsymbol{g}_2]$ = [10627,16765]	26	0.519873

Table 5.1: LDD-based results, rate $\frac{1}{2}$.

After these results, we may say that, increasing the generators length used to create $\boldsymbol{H}$, the entropy gets closer to the desired value of $\frac{1}{2}$ without never reaching it, as we can observe in the trade-off plot and weight spectra. We must say that the approach described in Section 5.2 is feasible until the Rl is lower than or equal to 18, due to the fact that after this value the number of states become unmanageable by the simulator, in terms of both the memory and the time expenses. We used another implementation, not described in this Thesis, to obtain the results of the other four CCs generators couples provided in the table above reaching Rl = 26, but the last one provided in [10, Sec. 12.3] remains untreatable even with this optimized version of the algorithm because of the huge number of states which is required by each SDM iteration. All these results are recovered by multiple proofs; more specifically we have done an average of 200 different input sequences up to $[\boldsymbol{g}_1, \boldsymbol{g}_2] = [53, 75]$, passing after to an amount of 20 iterations until the last case $[\boldsymbol{g}_1, \boldsymbol{g}_2] = [10627, 16765]$; nevertheless, being the

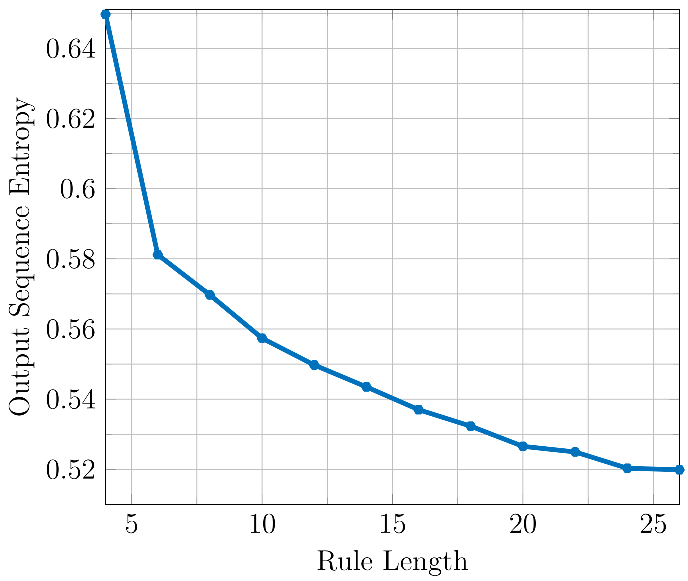

Figure 5.1: Memory-Performance trade-off.

syndrome length equal to 1000, the stability of the results is observed during the practical analysis. From Table 5.1, the best choice of generators is $[\boldsymbol{g}_1, \boldsymbol{g}_2] = [10627, 16765]$, which gives $\mathbb{H}(P_X) = 0.519873$; the intuition is that this particular $\boldsymbol{H}$ exploits at best the memory given by the states, guaranteeing the lowest entropy value presented. To conclude, the LDD-based SDM may be a good alternative to other DM; it cannot reach the entropy values of CCDM, which as we know from (2.11), with high input length, goes to a the rate; in fact, being the rate fixed, the entropy goes to this latter value. On the other hand, SDM exhibits the remarkable advantage to be decodable in a very easy way by syndrome coset; only a vector-matrix multiplication has to be done following (1.15). Hence, depending on the application, SDM can be more suitable compared to the others and this choice is a designer responsibility.

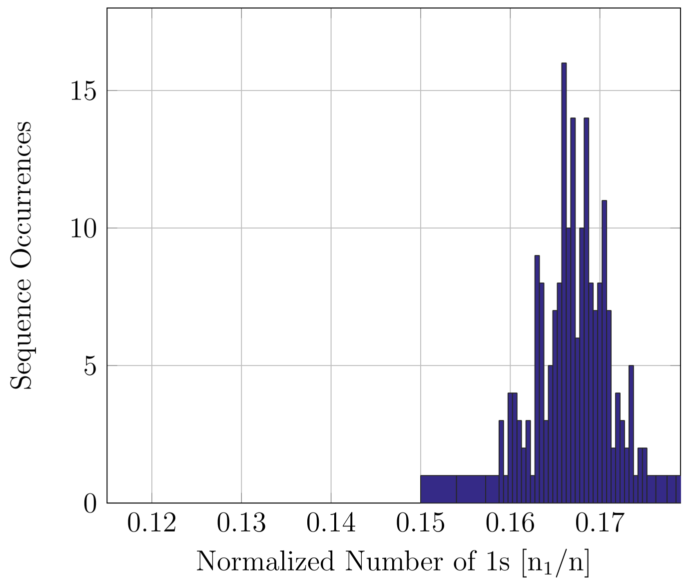

Figure 5.2: LDD-based SDM, generators [3,1].

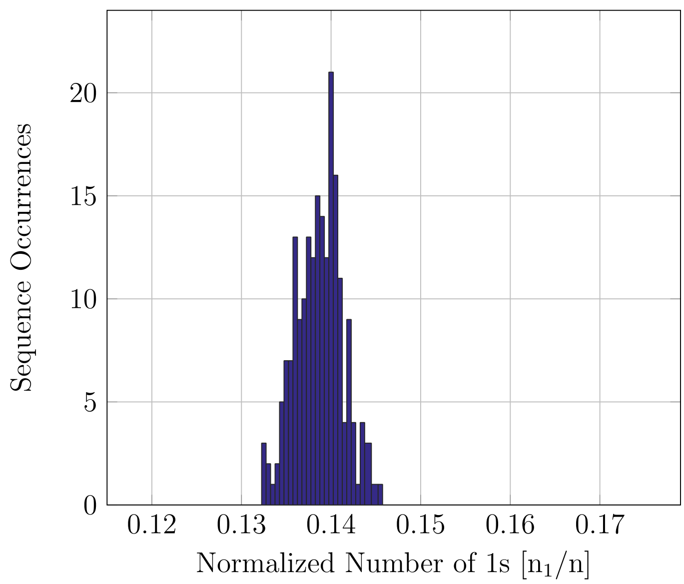

Figure 5.3: LDD-based SDM, generators [5,7].

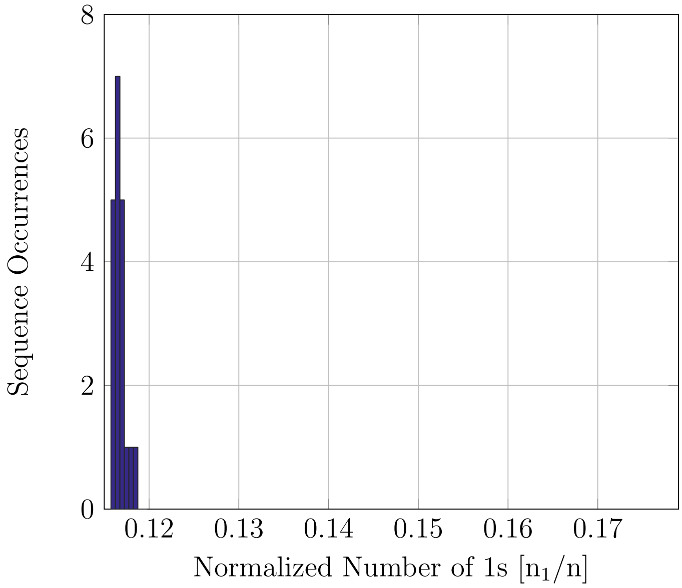

Figure 5.4: LDD-based SDM, generators [10627, 16765].

Conclusions

To sum up, in the Thesis an alternative distribution matching algorithm was provided, trying to analyse and explain its usefulness in certain types of problems. Only the binary case was treated; hence, a possible extension to the non-binary case would be reasonable and interesting. We found out that SR-based SDM is suitable to solve syndrome distribution matching on short input sequences, obtaining results in line or better compared to one of the most important competitor, the CCDM. Moreover, also LDD-based SDM gives interesting results, in particular an acceptable entropy, higher that CCDM, but with the possible implementation of a simpler decoder after due to the decoding property of SDM. Furthermore, put this DM in an end-to-end chain with a particular structure would be relevant.

Bibliography

[1] G. Böocherer, P. Schulte, and F. R. Steiner, "Probabilistic shaping and forward error correction for fiber-optic communication systems," *Journal of Lightwave Technology*, vol. 37, no. 2, Jan. 2019.

[2] G. Böcherer, F. Steiner, and P. Schulte, "Bandwidth efficient and rate-matched low-density parity-check coded modulation," *IEEE Transactions on Communications*, vol. 63, no. 12, pp. 4651–4665, Oct. 2015.

[3] P. Schulte and G. Böcherer, "Constant composition distribution matching," *IEEE Transactions on Information Theory*, vol. 62, no. 1, pp. 430–434, Nov. 2016.

[4] T. V. Ramabadran, "A coding scheme for m-out-of-n codes," *IEEE Transactions on Communications*, vol. 38, no. 8, pp. 1156–1163, Aug. 1990.

[5] A. K. Khandani and P. Kabal, "Shaping multidimensional signal spaces. I. optimum shaping, shell mapping," *IEEE Transactions on Information Theory*, vol. 39, no. 6, pp. 1799–1808, Nov. 1993.

[6] G. Forney, "Trellis shaping," *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 281–300, Mar. 1992.

[7] T. M. Cover and J. A. Thomas, "Elements of information theory 2nd edition," 2006.

[8] G. Böcherer, "Lecture notes on channel coding." [Online]. Available: https://arxiv.org/abs/1607.00974

[9] W. Ryan and S. Lin, *Channel codes: classical and modern.* Cambridge University Press, 2009.

[10] L. Shu and D. J. Costello, *Error control coding 2nd Edition.* Prentice-Hall, 2004.

[11] G. Böcherer, "Principles of coded modulation," habilitation thesis, Technical University of Munich, 2018. [Online]. Available: http://www.georg-boecherer.de/bocherer2018principles.pdf

[12] I. Csiszár, "The method of types [information theory]," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2505–2523, Oct. 1998.

[13] J. Wolf, "Efficient maximum likelihood decoding of linear block codes using a trellis," *IEEE Transactions on Information Theory*, vol. 24, no. 1, pp. 76–80, Jan. 1978.

[14] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, Apr. 1967.

Acknowledgements

I wish to thank a little bunch of people that have always sustained me along this path even in difficult moments:

- Moni e Ciro;
- Tutto il resto della mia Famiglia;
- Prof. E. Paolini, Dr. G. Böcherer and Prof. M. Chiani;
- Diego (Diagonal) e Davide (Divide);
- I miei amici storici;
- I miei amici e compagni universitari.

Each person in the list has contributed providing me the strength to give always my best, enduring me even when I did not deserve it. Hence, thank you all.