ABSTRACT

LOBO, RUBEN GERALD. On Locally Invertible Encoders and Multidimensional Convolutional Codes. (Under the direction of Dr. Mladen A. Vouk and Dr. Donald L. Bitzer).

Multidimensional (m-D) convolutional codes generalize the well known notion of a 1-D convolutional code defined over a univariate polynomial ring with coefficients in a finite field to multivariate polynomial rings. The more complicated structure of a multivariate polynomial ring when compared to a univariate one, however, makes the generalization nontrivial. While 1-D convolutional codes have been thoroughly understood and have wide applications in communication systems, the theory of m-D convolutional codes is still in its infancy, and these codes lack unified notation and practical implementation.

This dissertation develops a sequence space approach for realizing m-D convolutional codes. While most of the existing research is focused on algebraic aspects, fundamental issues regarding practical implementation that are well developed and fairly straightforward in the 1-D case have remained undefined for m-D convolutional codes. In this dissertation we address some of these issues.

We define a new notion of sequence space ordering and show that certain multivariate polynomial matrices which we call as locally invertible encoders, when transformed to the sequence space domain, have an invertible subsequence map between their input and output sequences. This subsequence map has a well defined structure that allows for the explicit construction of locally invertible encoders by performing elementary operations on the ground field without the use of any polynomial operations. We use the invertible subsequence map to introduce a novel method to encode and invert multidimensional sequences. We show that locally invertible encoders have good structural properties which make them a natural choice to generate multidimensional convolutional codes.

On Locally Invertible Encoders and Multidimensional Convolutional Codes

by

Ruben Gerald Lobo

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Computer Engineering

Raleigh

2006

Approved By:

Dr. Ernest Stitzinger

Dr. Brian L. Hughes

Dr. Alexandra Duel-Hallen

Dr. Mladen A. Vouk
Co-chair of Advisory Committee

Dr. Donald L. Bitzer
Co-chair of Advisory Committee

This dissertation and the work that went into it is
lovingly dedicated to my mother, Severine Lobo, who
has sacrificed so much and worked so hard
to give me a better life.

Biography

Ruben Lobo was born in Mangalore, India on May 24, 1976. He attended St. Aloysius College in his hometown where he received his elementary and secondary education. Ruben holds a Bachelor of Engineering degree in Electronics and Communication, awarded in 1998, from the Manipal Institute of Technology, Manipal, India. His interest in computer networks led him to North Carolina State University in August 2000, where he obtained his Master of Science degree in Computer Engineering in May 2002. His graduate advisor Dr. Mladen Vouk introduced him to Dr. Donald Bitzer who inspired him to pursue a doctoral degree. He is currently a doctoral candidate in the Department of Electrical and Computer Engineering, North Carolina State University. He has undertaken his doctoral research under the guidance of Dr. Mladen Vouk and Dr. Donald Bitzer.

Acknowledgements

I am indebted to a number of people who have helped me with my research and in the preparation of this dissertation. Foremost, I recognize my academic advisors Dr. Mladen Vouk and Dr. Donald Bitzer for their invaluable guidance, financial support and infinite patience throughout the course of my research. I truly could not have asked for better mentors. Their confidence in my abilities has meant a lot over the years. I am most grateful to Dr. Ernest Stitzinger for his wide-open door, who despite his busy schedule, always made time to help me with algebraic concepts. I also thank Dr. Brian Hughes and Dr. Alexandra Duel-Hallen for serving on my advisory committee.

My colleagues Jeff Ligon and Evan Ernst have helped me in many ways. Jeff has read (and re-read) many chapters and Evan has been an excellent sounding board for ideas. Their insightful suggestions have helped improve this work, for which I am very grateful. I owe a huge debt of gratitude to my advisors former students Dr. Ajay Dholakia and Dr. Havish Koorapaty. Their doctoral dissertations and research papers have provided a foundation for my work. I also thank Marhn Fullmer who has been consistently generous in providing me with equipment, software packages and technical support.

I am deeply indebted to Nisha Rajagopal for taking care of me, and for putting up with me and my dissertation during the days leading up to my defense. I could not have completed this project without her love and support. I thank Goktug Yorulmaz, Ashok Mallya, Vishwas Puttasubbappa and Raoul Jetley for being great friends and supporting me throughout my academic career. My family has been my biggest gift in life. Without their constant encouragement and support this work would not have been possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The fundamental principals governing modern digital communication were pioneered by Claude E. Shannon in 1948. With a series of landmark papers [51], starting with "A Mathematical Theory of Communication" [50], Shannon showed that by proper encoding of information at the source, errors introduced by noise in the channel can be reduced to any desired level without sacrificing the rate of transmission, as long as the transmission rate is kept less than the capacity of the channel. This fundamental and ground-breaking work gave rise to the twin disciplines of *information theory* and *coding theory*. While the former is the study of achievable bounds for communication, the latter attempts to realize these bounds by devising efficient channel coding techniques to achieve reliable communication over unreliable channels. Channel coding in this case refers to the systematic process of introducing redundancy into the message at the source, prior to transmission, so as to recover the original message at the output of the channel in the presence of errors.

In recent years there has been a dramatic increase in the volume of data that is being captured, stored, processed and transmitted as digital information. This has led to an increasing demand for efficient and reliable systems used in the transmission and storage of digital data. Channel coding is of central importance in the design of such digital communication systems. Block codes invented by Hamming [28] in 1950, and convolutional codes invented by Elias [17] in 1955, are two of the most common channel coding schemes in use today. This work is related to convolutional codes and its generalization to higher dimensions.

1.1 Multidimensional Convolutional Coding: An Overview

The theory of convolutional codes grew out and extended the theory of linear block codes in a new direction. The easiest way to contrast the two classes of codes is to compare their encoders. In block codes, the entire sequence of discrete information symbols (assumed to be from a finite field $\mathbb{F}_q$) is divided into blocks of k symbols each. These k-symbol blocks, denoted by $u = (u_0, u_1, \ldots, u_{k-1})$, are called messages. The encoder for a block code transforms each message u into an n-symbol block $v = (v_0, v_1, \ldots, v_{n-1})$ called a codeword. Since each codeword depends only on the corresponding message; that is, each message is encoded independently, the encoder is memoryless. The set of q^k codewords of length n corresponding to q^k different possible messages of length k forms the *block code*. The ratio $\frac{k}{n}$ is called the *code rate*. For the code to be useful, we require $k < n$ so that a different codeword is assigned to each message. Typically, k and n are large integers and $n - k$ redundant symbols are added to each message to form the codeword. It is these redundant bits that provide the block code with the capability to combat errors. For a fixed code rate, more redundant symbols can be added by increasing the length k of the message and the length n of the codeword while holding the rate $\frac{k}{n}$ constant. The exact technique of choosing these redundant symbols to achieve reliable communication over an unreliable channel is a topic of central importance in the design of block codes.

In the case of convolutional codes, the entire sequence of discrete information symbols, denoted by $u := (u^{(1)}, u^{(2)}, \ldots, u^{(k)})$, is considered to be made up of k interleaved sequences, where each sequence $u^{(y)}$, $y = 1, \ldots, k$ is given as $u^{(y)} = (u_0^{(y)}, u_1^{(y)}, u_2^{(y)}, \ldots)$. That is, $u = (u_0^{(1)}, u_0^{(2)}, \ldots u_0^{(k)},\ u_1^{(1)}, u_1^{(2)}, \ldots u_1^{(k)},\ u_2^{(1)}, u_2^{(2)}, \ldots u_2^{(k)},\ \ldots)$. A convolution encoder is implemented as a k-input, n-output linear sequential circuit. At each time instant i, the encoder transforms a k-symbol message block $u_i = (u_i^{(1)}, u_i^{(2)}, \ldots, u_i^{(k)})$ into an n-symbol encoded block $v_i = (v_i^{(1)}, v_i^{(2)}, \ldots, v_i^{(n)})$. The resulting encoded sequence $v := (v^{(1)}, v^{(2)}, \ldots, v^{(n)}) = (v_0^{(1)}, v_0^{(2)}, \ldots v_0^{(n)},\ v_1^{(1)}, v_1^{(2)}, \ldots v_1^{(n)},\ v_2^{(1)}, v_2^{(2)}, \ldots v_2^{(n)},\ \ldots)$ is called the codeword. The key difference here when compared to block codes is that at time instant i the encoded block v_i depends not only on the corresponding message block u_i, but also on M previous message blocks $u_{i-1}, \ldots, u_{i-M}$. Hence, the encoder has memory order M. The set of all possible encoded sequences produced by the encoder forms the *convolutional code*. The ratio $\frac{k}{n}$ is called the *code rate*. The parameters k, n and M, with $k < n$, determine the amount of redundancy that is introduced. Typically, k and n are small

integers and more redundancy is added by increasing the memory order M while holding the rate $\frac{k}{n}$ constant. The design of convolutional codes is focused on the use of the memory to achieve reliable communication while keeping the code rate fixed.

Multidimensional convolutional codes generalize the above notion of convolutional codes to m-dimensional (m-D) sequences. In this context the convolutional code described above is a 1-D convolutional code. A m-D sequence can be viewed as a set of symbols indexed on an m-dimensional integer lattice. In m-D convolutional codes, the entire input sequence of discrete information symbols, denoted by $u := (u^{(1)}, u^{(2)}, \ldots, u^{(k)})$, is considered to be made up of k interleaved m-D sequences $u^{(y)}$, $y = 1, \ldots, k$.

For example, a 2-D sequence $u = (u^{(1)}, u^{(2)}, u^{(3)})$ consisting of three interleaved 2-D sequences $u^{(y)} = (u^{(y)}_{(0,0,0)}, u^{(y)}_{(1,0,0)}, u^{(y)}_{(2,0,0)}, \ldots, u^{(y)}_{(0,1,0)}, \ldots, u^{(y)}_{(0,0,1)}, \ldots)$, is of the form $u = (u^{(1)}_{(0,0,0)}, u^{(2)}_{(0,0,0)}, u^{(3)}_{(0,0,0)}, \ u^{(1)}_{(1,0,0)}, u^{(2)}_{(0,0,0)}, u^{(3)}_{(0,0,0)}, \ \ldots)$. We will make this rather unwieldily representation more precise in Chapter 2.

A k-input, n-output, m-D convolutional encoder accepts at its input a m-D sequence $u = (u^{(1)}, \ldots, u^{(k)})$ and produces at its output an m-D encoded sequence $v = (v^{(1)}, \ldots, v^{(n)})$. At each time instant, the encoder transforms a k-symbol message block $u_{(i_1,\ldots,i_m)} = (u^{(1)}_{(i_1,\ldots,i_m)}, \ldots, u^{(k)}_{(i_1,\ldots,i_m)})$ from the input sequence u, into an n-symbol encoded block $v_{(i_1,\ldots,i_m)} = (v^{(1)}_{(i_1,\ldots,i_m)}, \ldots, v^{(n)}_{(i_1,\ldots,i_m)})$. The resulting m-D encoded sequence $v = (v^{(1)}, v^{(2)}, \ldots, v^{(n)})$ is called the codeword. At any time instant, the encoded block $v_{(i_1,\ldots,i_m)}$ is a function not only of the corresponding message block $u_{(i_1,\ldots,i_m)}$, but also a fixed number of "earlier" message blocks $u_{(i_1-1,\ldots,i_m-1)}, \ldots, u_{(i_1-M_1,\ldots,i_m-M_m)}$. The integers $M_1, \ldots, M_m$ are the memory orders of the encoder in each of the m dimensions. The set of all possible m-D encoded sequences produced by the encoder forms the *m-D convolutional code* and the ratio $\frac{k}{n}$ is called the *code rate*. The main attraction here when compared to 1-D convolutional codes is that the encoding process (which is essentially a discrete m-D convolution operation) is inherently recursive in nature. By virtue of the memory orders M_i, $i = 1, \ldots, m$, the encoder spreads the information contained in a single information symbol of the input sequence over several encoded symbols of the codeword along each of the m dimensions. As a result, multiple symbols of a codeword would have to be changed simultaneously in every dimension for errors to go undetected.

From the above description, one could view a block code as a 0-D convolutional code without memory. This would make block codes seem to be a degenerate special case of m-D convolutional codes. To adopt this point of view however, would be a gross

oversimplification of the theory of block codes. Block codes have a rich algebraic structure which allow for simple and efficient decoding algorithms. The book by Lin and Costello [35] gives an excellent treatment to both block codes and 1-D convolutional codes. From here on, when using the term "m-D" we will always assume that $m \geq 1$.

When encoding 1-D sequences, it natural to associate the sequential structure of the data flow with time ordering. This motivates the habit of considering information sequences that start at some finite time $i \leq 0$ and continue indefinitely. As a result, the 1-D sequence $u = (u^{(1)}, \ldots, u^{(k)})$ can be represented by the vector $u(z_1) \in \mathbb{F}((z_1))^k$, where $\mathbb{F}((z_1))$ is the field of formal Laurent series with left compact support. Therefore, a 1-D convolutional code is usually defined as a k-dimensional subspace of $\mathbb{F}((z_1))^n$, with a basis of rational vectors in $\mathbb{F}(z_1)^n$. The restriction that the basis vectors of the code have components in the field of rationals $\mathbb{F}(z_1)$ allows for the construction of feedback encoders. This definition goes back to the work of Forney [22, 26] and has been adopted in the books by Johannesson and Zigangirov [30], Dholakia [12] and Piret [45] which are devoted solely to 1-D convolutional codes.

For $m > 1$, there is no natural notion of causality inducing a particular ordering in a m-D space, and so the notions of past and future that are natural in a 1-D space do not generalize in a straightforward manner to m dimensions. Since the m-D encoding process (i.e. m-D convolution) is inherently recursive in nature, some a priori assumption has to be made on the direction of recursion. This might cause some difficulty in allowing for information sequences that extend indefinitely in all dimensions of the m-D space, because when encoding symbols of an information sequence along a particular dimension at some point the encoder has to consider symbols that extend in other dimensions as well. For this reason, the information sequences, and consequently codewords, are restricted to having finite support in m-D convolutional codes. With the finite support restriction one can associate m-D sequences with either polynomials or Laurent polynomials in m variables. This leads to a module-theoretic approach to convolutional codes. Fornasini and Valcher [18, 54, 19] were the first to describe 2-D convolutional codes with finite support using the Laurent polynomial ring $\mathbb{F}[z_1^{\pm}, z_2^{\pm}]$. They define a 2-D convolutional code as a submodule of $\mathbb{F}[z_1^{\pm}, z_2^{\pm}]^n$, where the code can be viewed as a set of 2-D sequences indexed on the integer plane $\mathbb{Z}^2$. The papers [20, 21] by the same authors on finite support multidimensional behaviors is also relevant to m-D convolutional codes, because behaviors (in the system theoretical sense) and convolutional codes are often considered as dual. This duality has

been well explored in [48, 27]. Weiner [56, 57] describes m-D convolutional codes with finite support over the polynomial ring $\mathbb{F}[z_1, \ldots, z_m]$ and defines the code as a submodule of $\mathbb{F}[z_1, \ldots, z_m]^n$. In this case the code can be viewed as a set of sequences indexed on the nonnegative integer lattice $\mathbb{N}^m$. Weiner's Ph.D. dissertation [56] covers many aspects of m-D convolutional codes in detail, especially highlighting the similarities and differences with the well known theory of 1-D convolutional codes, and is probably the most comprehensive treatment of the subject to date.

While the m-D encoding process ($m > 1$) is a straightforward generalization of the 1-D case, there are significant theoretical differences between 1-D and m-D convolutional codes. The complicated structure of a multivariate polynomial ring when compared to a univariate one is primarily due to the fact that it is not a principal ideal domain (PID). This makes the generalization nontrivial. Many of the decomposition techniques [22, 42] used in 1-D codes that rely on the ring being a PID no longer apply. However, substantial progress in the field of multidimensional systems theory (see for example [7]) has made the task of analyzing m-D convolutional codes more realizable. As a result, interest in m-D convolutional coding theory is growing [18, 54, 57, 56, 9, 10, 27, 32]. Since m-D convolutional codes involve a system of linear multivariate polynomial equations, the use of Gröbner basis as a computational technique becomes attractive. To this end, some results on finding the parity check matrix and reducing the number of delay elements in the encoder realization using Gröbner basis techniques have been reported in [9, 10].

1.2 Contributions

This dissertation develops a sequence space approach to m-D convolutional codes. While most of the current research is focused on the algebraic theory, parameters like encoder memory, constraint lengths, and sequence ordering that are essential for practical implementation of m-D convolutional coding are undefined. We address some of these issues that are well understood in the 1-D case but have not been dealt with in the algebraic treatment. The ordering of information in 1-D codes is a nonissue due to its singular dimension. However, in a m-D sequence space a priori assumptions on the ordering of information cannot be avoided during practical implementation. This necessitates a predefined ordering on the sequence space which we define. We build on this notion of sequence space ordering and show that certain multivariate polynomial matrices, when transformed to the

sequence space domain, have an invertible subsequence map between their input and output sequences. We show that these matrices have full row rank and define them as locally invertible encoders. We use the invertible subsequence map to introduce a novel method to encode and invert m-D sequences. We show that locally invertible encoders are easy to construct by defining their reduced encoding matrix which has a well defined structure. We then proceed to describe the structural properties of these encoders. We show that locally invertible encoders are zero prime and derive an encoder inverse. We also derive a set of pseudo-inverses (inverses with delay) for these encoders and show that they can be used for error detection. Finally, we use the invertible subsequence map to compute a set of parity check sequences that can be used to obtain error syndromes. A subset of these parity check sequences are used to construct a parity check matrix which describes the dual code. All computations involving these encoders are performed using elementary operations on the ground field without the use of any polynomial operations. Some of the material presented in this dissertation has been reported in [39, 40, 38].

1.3 Outline of Dissertation

Chapter 2 introduces a framework for representing multidimensional information in two transform-domains: algebraically as a vector of polynomials in m variables and geometrically as an m-D composite sequence. We define an isomorphism between the m-D finite sequence space and the multivariate polynomial ring. The polynomial representation is used to analyze the structure of m-D sequences and elucidate coding problems.

In Chapter 3 we review the algebraic aspects of m-D convolutional codes from a module-theoretic viewpoint. The nontrivial generalization of 1-D to m-D convolutional codes is described. We define m-D convolutional codes, their generator matrices, and their encoders. We look at three notions of primeness that arise when considering multivariate polynomial matrices as m-D convolutional encoders and discuss the effects they have on the existence of parity check matrices, dual codes and encoder inverses.

Chapter 4 provides a sequence domain analysis of the encoding process. Parameters like encoder memory and constraint lengths are defined. A new notion of sequence space ordering is described. This notion of sequence space ordering forms the foundation for our work in the remaining chapters.

Chapter 5 introduces the notion of local invertibility based on the concept of

subsequence mapping. We build on the sequence space ordering introduced in Chapter 4 and define locally invertible encoders. A novel method to encode and invert m-D sequences using the invertible subsequence map is discussed. Explicit construction of locally invertible encoders using the reduced encoding matrix is described. The structural properties of locally invertible encoders is explored. A method to obtain an encoder inverse and a set of pseudo-inverses from the inverse of the reduced encoding matrix is described.

In Chapter 6 we explore three approaches to detect errors and obtain error syndromes using locally invertible encoders. The first method is based on the subsequence overlap that arises when inverting received sequences using the invertible subsequence map. The second approach is an algebraic interpretation of the subsequence overlap using pseudo-inverses obtained from the reduced encoding matrix. The third approach describes a method to obtain a set of parity check sequences using an extended subsequence map. A subset of these parity check sequences is used to construct a parity check matrix which forms an encoder for the dual code.

Chapter 7 concludes the dissertation and points towards future research directions.

Chapter 2

Representation of Multidimensional Information

This chapter introduces a framework for representing multidimensional information. We begin by setting some notation to describe multidimensional information geometrically as multidimensional sequences. We then define a map for transforming these sequences into multivariate polynomials. The polynomial representation will provide us with the algebraic machinery required to analyze the structure of these sequences and elucidate the coding problems.

Let $\mathbb{F} = \mathbb{F}_q$ be a finite field with q elements. Let $\mathbb{N}$ be the set of nonnegative integers, and let $\mathbb{N}^m$ represent a m-D lattice with axes $i_1, \ldots, i_m \in \mathbb{N}$. Examples of such lattices are shown in Figure 2.1.

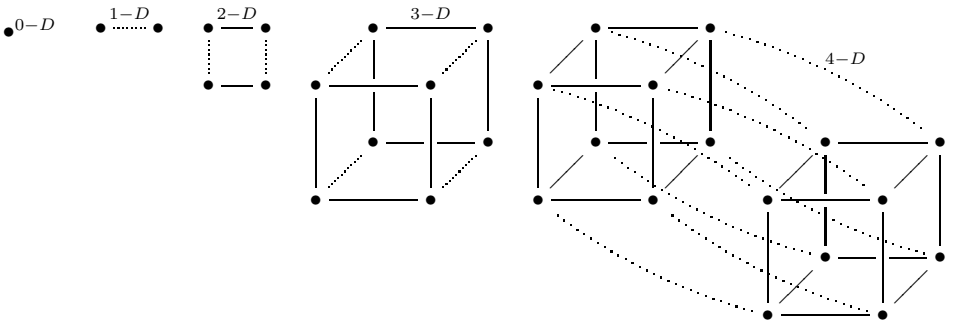


Figure 2.1: Multidimensional integer lattices.

Definition 2.0.1. *A m-D sequence over $\mathbb{F}$ is a map*

$$\begin{array}{rccc} u: & \mathbb{N}^m & \rightarrow & \mathbb{F} \\ & (i_1, \ldots, i_m) & \mapsto & u_{(i_1,\ldots,i_m)} \end{array} \tag{2.1}$$

that associates the coordinates $(i_1, \ldots, i_m)$ of $\mathbb{N}^m$ with the elements $u_{(i_1,\ldots,i_m)} \in \mathbb{F}$. The sequence u is said to have finite support if $u_{(i_1,\ldots,i_m)} = 0$ for all but finitely many $(i_1, \ldots, i_m) \in \mathbb{N}^m$.

Next, we consider the case when there is more than one element of the field attached to each coordinate of the lattice.

Definition 2.0.2. *If there are k elements of $\mathbb{F}$ attached to each coordinate of $\mathbb{N}^m$, then a m-D composite sequence over $\mathbb{F}$ is a map*

$$\begin{array}{rccc} u: & \mathbb{N}^m & \rightarrow & \mathbb{F}^k \\ & (i_1, \ldots, i_m) & \mapsto & u^{(1)}_{(i_1,\ldots,i_m)}, \ldots, u^{(k)}_{(i_1,\ldots,i_m)} \end{array} \tag{2.2}$$

that associates the coordinates $(i_1, \ldots, i_m)$ of $\mathbb{N}^m$ with the vectors $[u^{(1)}_{(i_1,\ldots,i_m)}, \ldots, u^{(k)}_{(i_1,\ldots,i_m)}]$ in the k-tuple vector space $\mathbb{F}^k$. The sequence $u := (u^{(1)}, \ldots, u^{(k)})$ is called a composite sequence, because it can be viewed as being made up of k interleaved m-D sequences $u^{(j)}$, $j = 1, \ldots, k$ with elements $u^{(j)}_{(i_1,\ldots,i_m)} \in \mathbb{F}$.

Note 2.0.1. For $k = 1$, a m-D composite sequence a m-D sequence are one and the same.

Consider the block diagram of a typical digital communication system shown in Figure 2.2. The *information source* produces a continuous waveform or a sequence of discrete symbols which needs to be communicated to the destination. The *source encoder* transforms the source output into a 1-D *information sequence* $u = (u^{(1)}, \ldots, u^{(k)})$ over $\mathbb{F}$ with coordinates in $\mathbb{N}^1$. In this case, the 1-D lattice $\mathbb{N}^1$ is usually associated with time ordering and is considered to be the time axis. The source encoder typically tries to minimize the number of symbols per time unit required to represent the source output in such a way that it can be reconstructed form the information sequence u without ambiguity. The *channel encoder* in this scenario is a k-input, n-output linear sequential circuit called a convolutional encoder. The convolutional encoder transforms the information sequence u into a 1-D encoded sequence $v = (v^{(1)}, \ldots, v^{(n)})$ called a *codeword*. We will postpone the precise definition of this 1-D convolutional encoder until Chapter 3. In general, the

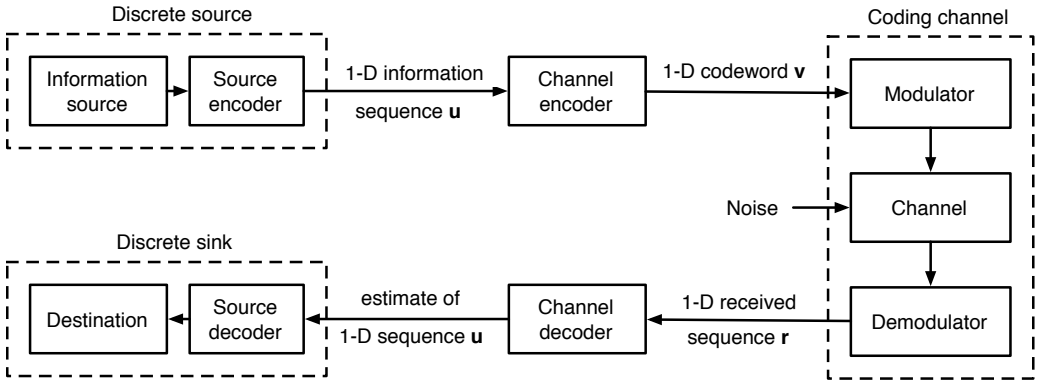


Figure 2.2: Block diagram of a typical digital communication system.

code resulting from such a 1-D encoding technique will be invariant to shifts only with respect to the time axis. The *modulator* transforms each output symbol of the channel encoder into a waveform that is transmitted over the channel, where it assumed to be corrupted by noise. The *demodulator* processes each received waveform and produces a discrete output. The 1-D sequence of demodulator outputs corresponding to the codeword v is called the received sequence $r = (r^{(1)}, \ldots, r^{(n)})$. The *channel decoder* attempts to recover the original information sequence from the received sequence r and produces at its output a 1-D estimate $\tilde{u} = (\tilde{u}^{(1)}, \ldots, \tilde{u}^{(k)})$ of the information sequence u. The *source decoder* transforms the estimate $\tilde{u}$ into an estimate of the information source output and delivers it to the *destination*.

Multidimensional convolutional encoders generalize the above encoding process to multidimensional sequences. The block diagram of a m-D convolutional coding system is

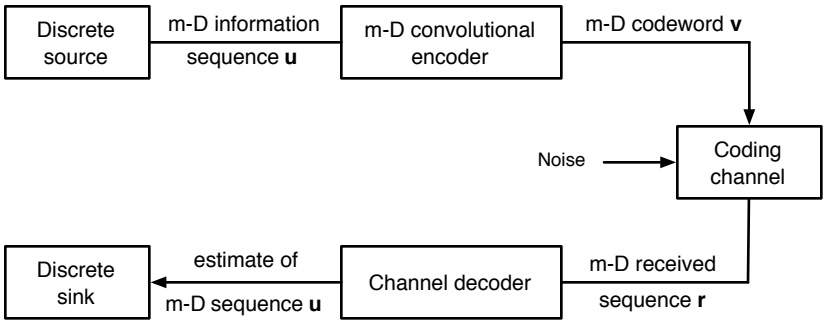


Figure 2.3: Block diagram of a m-D convolutional coding system.

shown in Figure 2.3. Here, it is assumed that the discrete source outputs an m-D information sequence. A k-input, n-output m-D convolutional encoder accepts at its input a m-D information sequence $u = (u^{(1)}, \ldots, u^{(k)})$ over $\mathbb{F}$ with coordinates in $\mathbb{N}^m$ and produces at its output a m-D codeword $v = (v^{(1)}, \ldots, v^{(n)})$ over $\mathbb{F}$ which also has its coordinates in $\mathbb{N}^m$. The m-D convolutional code resulting from this encoding process is shift-invariant along all the coordinate axes of the m-D space. In what follows, we introduce the necessary algebraic framework that will allow for such an encoding scheme.

Definition 2.0.3. *A finite m-D sequence space of $\mathbb{F}$ is the set of all mappings*

$$S = \{u : \mathbb{N}^m \to \mathbb{F}\} \tag{2.3}$$

where every m-D sequence u has finite support.

In particular, the finite m-D sequence space S has the following distinguished sequences. The *additive identity* in S is the zero sequence, where $u_{(i_1,\ldots,i_m)} = 0$ for all $(i_1, \ldots, i_m) \in \mathbb{N}^m$. The *multiplicative identity* in S is the sequence with $u_{(i_1,\ldots,i_m)} = 0$ for all but the origin $(0, \ldots, 0)$ of $\mathbb{N}^m$, and $u_{(0,\ldots,0)} = 1$. For a given m, we define an addition and a multiplication of sequences as follows.

1. The sum $w = u + v$ of two sequences u, v is given by

$$w_{(i_1,\ldots,i_m)} = u_{(i_1,\ldots,i_m)} + v_{(i_1,\ldots,i_m)}, \tag{2.4}$$

   where $u_{(i_1,\ldots,i_m)} + v_{(i_1,\ldots,i_m)}$ denotes addition of $u_{(i_1,\ldots,i_m)}$ and $v_{(i_1,\ldots,i_m)}$ in $\mathbb{F}$.

2. The product (discrete m-D convolution) $w = u * v$ of two sequences u, v is given by

$$w_{(i_1,\ldots,i_m)} = \sum_{l_m=0}^{\deg_m v} \cdots \sum_{l_1=0}^{\deg_1 v} u_{((i_1-l_1),\ldots,(i_m-l_m))} v_{(l_1,\ldots,l_m)}, \tag{2.5}$$

   where $u_{((i_1-l_1),\ldots,(i_m-l_m))}\ v_{(l_1,\ldots,l_m)}$ denotes multiplication of $u_{((i_1-l_1),\ldots,(i_m-l_m))}$ and $v_{(l_1,\ldots,l_m)}$ in $\mathbb{F}$. Since the coordinates of $\mathbb{N}^m$ are nonnegative, $u_{((i_1-l_1),\ldots,(i_m-l_m))} \triangleq 0$ for all $i_r < l_r$. Here, $\deg_i v$ is the largest support of the sequence v along the ith dimension. That is, $\deg_i v$ is the largest l_i for which $v_{(l_1,\ldots,l_i,\ldots,l_m)}$ is nonzero.

We can now easily verify that S satisfies all the axioms of a commutative ring with a multiplicative identity. In fact, a finite m-D sequence space of $\mathbb{F}$ is isomorphic to a m-variate polynomial ring over $\mathbb{F}$.

Let $R = \mathbb{F}[z_1, \ldots, z_m]$ be a polynomial ring in m variables over $\mathbb{F}$. A m-D sequence in S can be uniquely mapped to an m-variate polynomial $u(z_1, \ldots, z_m) \in R$ with the transformation

$$\begin{array}{rccc} \psi: & S & \rightarrow & R \\ & u & \mapsto & \sum\limits_{(i_1,\ldots,i_m)} u_{(i_1,\ldots,i_m)} z_1^{i_1} \cdots z_m^{i_m} \end{array}, \tag{2.6}$$

by associating the coordinates of $\mathbb{N}^m$ with monomials of R via the correspondence

$$(i_1, \ldots, i_m) \leftrightarrow z_1^{i_1} \cdots z_m^{i_m}.$$

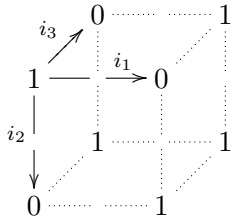
An element $u_{(i_1,\ldots,i_m)} \in \mathbb{F}$ of the sequence $u \in S$ at the coordinate $(i_1, \ldots, i_m)$ becomes the coefficient of the term $z_1^{i_1} \cdots z_m^{i_m}$ of the polynomial $u(z_1, \ldots, z_m) \in R$.

Since the product of two sequences is carried out using discrete convolution in the domain S and polynomial multiplication in the range R, the bijective map $\psi : S \rightarrow R$ is an $\mathbb{F}$-isomorphism with the law of composition

$$\psi(u * v) = \psi(u)\psi(v). \tag{2.7}$$

Example 2.0.1. Let $u \in S$ be a 3-D sequence over $\mathbb{F}_2$ with coordinates (i_1, i_2, i_3) in $\mathbb{N}^3$ as shown below.

$u =$



If we consider the top left point of the lattice as the origin, then the polynomial representation $u(z_1, z_2, z_3) \in R$ of the sequence $u \in S$ is

$$u(z_1, z_2, z_3) = \psi(u) = 1 + z_1z_2 + z_1z_3 + z_2z_3 + z_1z_2z_3.$$

Definition 2.0.4. *A finite m-D composite sequence space of $\mathbb{F}$ is the set of all mappings*

$$S^k = \{u : \mathbb{N}^m \rightarrow \mathbb{F}^k\} \tag{2.8}$$

where every m-D composite sequence $u = (u^{(1)}, \ldots, u^{(k)})$ consists of $u^{(j)} \in S$ with finite support.

For a given m and k, the following operations on composite sequences are well defined.

1. The sum $w = u + v$ of composite sequences $u = (u^{(1)}, \ldots, u^{(k)})$ and $v = (v^{(1)}, \ldots, v^{(k)})$ in S^k is given by $w^{(j)} = u^{(j)} + v^{(j)}$ in S as defined in (2.4).

2. Scalar multiplication $w = \alpha * u$ of a composite sequence $u = (u^{(1)}, \ldots, u^{(k)})$ in S^k by a sequence $\alpha \in S$ is given by $w^{(j)} = \alpha * u^{(j)}$ as defined in (2.5).

A m-D composite sequence $u \in S^k$ is transformed into a vector of k m-variate polynomials in the k-tuple R-module R^k using the $\mathbb{F}$-isomorphism

$$\begin{array}{rccc} \psi^k : & S^k & \to & R^k \\ & \left(u^{(1)}, \ldots, u^{(k)}\right) & \mapsto & \left[\psi(u^{(1)}), \ldots, \psi(u^{(k)})\right] \end{array} . \tag{2.9}$$

Example 2.0.2. Let $u \in S^3$ be a 2-D composite sequence over $\mathbb{F}_2$ with coordinates (i_1, i_2) in $\mathbb{N}^2$ as shown below.

$$u = (u^{(1)}, u^{(2)}, u^{(3)}) \quad \begin{array}{cc} 101 & 010 \\ 001 & 100 \\ 010 & 001 \\ 100 & 110 \end{array} \rightsquigarrow \begin{array}{cc} u^{(1)} \\ 1\ 0 \\ 0\ 1 \\ 0\ 0 \\ 1\ 1 \end{array}, \begin{array}{cc} u^{(2)} \\ 0\ 1 \\ 0\ 0 \\ 1\ 0 \\ 0\ 1 \end{array}, \begin{array}{cc} u^{(3)} \\ 1\ 0 \\ 1\ 0 \\ 0\ 1 \\ 0\ 0 \end{array}$$

Here, each $u^{(j)} \in S$ is a 2-D sequence over $\mathbb{F}_2$. The polynomial representation of the composite sequence $u \in S^3$ is

$$\psi^3(u) = [1 + z_1 z_2 + z_2^3 + z_1 z_2^3 \quad z_1 + z_2^2 + z_1 z_2^3 \quad 1 + z_2 + z_1 z_2^2] \in R^3.$$

The sequence space to polynomial ring transformation is useful for analyzing the structural properties of sequence spaces. For $m = 1$, the univariate polynomial ring $R = \mathbb{F}[z_1]$ is a principal ideal domain (PID), whereas for values of $m > 1$ the multivariate polynomial ring $R = \mathbb{F}[z_1, \ldots, z_m]$ is a noetherian ring but not a PID. Modules are natural generalizations of vector spaces to rings, where the scalars are taken from the ring over which they are defined instead of a field. A submodule of R^k is a nonempty subset that is closed under addition and scalar multiplication. Since S^k is isomorphic to R^k, it is natural to define subsets of S^k that are analogous to submodules of R^k. In order to do this we need to first define the following operation on S.

Definition 2.0.5. *A right-shift operation on a sequence $u \in S$, denoted by $z_1^{i_1} \cdots z_m^{i_m}(u)$, is defined as*

$$\begin{array}{rccc} z_1^{i_1} \cdots z_m^{i_m} : & S & \to & S \\ & u & \mapsto & \psi^{-1}(z_1^{i_1} \cdots z_m^{i_m} \psi(u)) \end{array} \tag{2.10}$$

The exponent i_j of the variable z_j represents the delay or the amount by which the sequence u is shifted to the right along the jth dimension of sequence space S.

Definition 2.0.6. *A m-D composite sequence subspace of S^k is a nonempty subset $\mathcal{C}$ that satisfies the following closure properties.*

i) $\mathbb{F}$-linearity*: If $u, v \in \mathcal{C}$, then the $\mathbb{F}$-linear combination $f_1 u + f_2 v \in \mathcal{C}$ for all $f_1, f_2 \in \mathbb{F}$.*

ii) Right-shift invariance*: If $u \in \mathcal{C}$, then $z_1^{i_1} \cdots z_m^{i_m}(u^{(1)}, \ldots, u^{(k)}) \in \mathcal{C}$ for all $i_1, \ldots, i_m \in \mathbb{N}^m$. That is, $\mathcal{C}$ is invariant with respect to shifts in $\mathbb{N}^m$ along the coordinate axes.*

From the $\mathbb{F}$-linearity and right-shift invariance of $\mathcal{C}$ it follows that: If $u \in \mathcal{C}$, then $\alpha * u \in \mathcal{C}$ for all $\alpha \in S$.

By virtue of the $\mathbb{F}$-linearity and right-shift invariance of $\mathcal{C} \subseteq S^k$, the $\mathbb{F}$-isomorphism defined in (2.9) transforms $\mathcal{C}$ into a R-submodule $\psi^k(\mathcal{C})$ of R^k. If $\mathcal{C}$ is any composite sequence subspace of S^k, we can speak of linear combinations, linear dependence and linear independence just as we do in modules. A set of elements $\{u_1, \ldots, u_r\}$ of $\mathcal{C}$ is said to *generate (or span)* $\mathcal{C}$ if every $u \in \mathcal{C}$ is a linear combination:

$$u = s_1 * u_1 + \cdots + s_r * u_r, \quad \text{with} \quad s_i \in S. \tag{2.11}$$

We call the set of elements $\{u_1, \ldots, u_r\}$ of $\mathcal{C}$ *independent* if no nontrivial linear combination is zero. That is,

$$\text{If} \quad s_1 * u_1 + \cdots + s_r * u_r = 0, \quad \text{with} \quad s_i \in S, \quad \text{then} \quad s_i = 0,\ i = 1, \ldots, s. \tag{2.12}$$

A generating set is a *basis* if it is independent. We must be careful not to apply to $\mathcal{C}$ any vector space results that depend on division by nonzero scalars. In particular, if $u_1, \ldots, u_r$ are linearly dependent, we cannot conclude that some u_i is a linear combination of the others. Here we note that for $m = 1$, since R is a PID any submodule of R^k is free (i.e. it has a basis). The well known 1-D convolutional codes in the sense of [30, 35, 12] can be viewed as submodules of $R = \mathbb{F}[z_1]$. For $m > 1$, since R is a noetherian ring any submodule of R^k is finitely generated but is not necessarily free. That is, for $m > 1$ every submodule of R^k has a finite number of generators, but some submodules of R^k may not have a basis. In the next chapter, we will see that this is a crucial reason which makes the generalization of convolutional codes to multidimensional spaces nontrivial.

Chapter 3

Multidimensional Convolutional Codes: Algebraic Aspects

This chapter reviews the algebraic theory of m-D convolutional codes. Works of Weiner [56, 57], and Fornasini and Valcher [18, 20, 21] are the primary resources.

3.1 Codes, Generator Matrices and Encoders

In the sequel we will consider the finite m-D composite sequence space S^k to be the information space or the input space. Our goal is to protect input sequences in S^k by adding redundancy such that the resulting encoding scheme is shift-invariant with respect to the coordinate axes of the m-D space. To achieve this we have to inject the input sequences into a larger space S^n, where $k < n$. We will refer to the finite m-D composite sequence space S^n as the output space. Let the R-modules R^k and R^n be the equivalent polynomial representations of S^k and S^n transformed using the $\mathbb{F}$-isomorphism defined in (2.9).

Note 3.1.1. To avoid cumbersome notation we will henceforth use the symbol "z" to represent the variables $z_1, \ldots, z_m$. Whenever necessary, the context (dimension) will be made clear by explicitly defining the value of m.

Let $G(z) \in R^{k \times n}$ be a rectangular ($k < n$) m-variate polynomial matrix with

elements $g(z)_x^{(y)} \in R$.

$$G(z) = \begin{bmatrix} g(z)_1^{(1)}, & \ldots, & g(z)_1^{(n)} \\ & \vdots & \\ g(z)_k^{(1)}, & \ldots, & g(z)_k^{(n)} \end{bmatrix} \tag{3.1}$$

Consider the R-module homomorphism induced by $G(z)$ between the R-modules R^k and R^n.

$$\begin{array}{ccc} R^k & \xrightarrow{G(z)} & R^n \\ u(z) & \mapsto & v(z) \end{array} \tag{3.2}$$

A input polynomial vector $u(z) \in R^k$ is encoded by left multiplication with $G(z)$ to obtain the output polynomial vector $v(z) \in R^n$. The image $\mathcal{C} = \mathrm{im}(G(z)) \subset R^n$ generated by the row space of $G(z)$ is an R-submodule of R^n.

$$\mathcal{C} = \{v(z) \in R^n : v(z) = u(z)G(z)\} \tag{3.3}$$

Since the R-matrix $G(z)$ generates the R-module $\mathcal{C}$, we refer to it as the *generator matrix* of $\mathcal{C}$. The *rank of the generator matrix* $G(z)$ is equal to largest integer $l \leq k$ for which there is nonzero $l \times l$ minor of $G(z)$. In order to be able to reconstruct the input $u(z)$ from the output $v(z)$ we require the generator map to be injective; that is, the generator matrix should be of full row rank.

Definition 3.1.1. *A generator matrix $G(z) \in R^{k \times n}$ is called an m-D convolutional encoder if it has rank k.*

A R-module is said to be *free* if it has a *basis*. The number of elements in the basis is called the *rank of the free R-module*. Clearly, the input space R^k and the output space R^n are free R-modules with rank k and n respectively because they contain the k-tuple and n-tuple standard basis vectors. Since a convolutional encoder is an generator matrix that has full row rank, its rows form a basis for the R-module $\mathcal{C}$ that it generates.

Definition 3.1.2. *A m-D convolutional code $\mathcal{C}$ is a free R-submodule of R^n. The elements of $\mathcal{C}$ are called codewords. If the rank of $\mathcal{C}$ is k, then the rate of the code is $\frac{k}{n}$.*

A *convolutional code* and its *encoder* constitute different entities. But the two are related by the fact that one cannot exist without the other. This is especially important when considering codes over modules. For $m = 1$ we have $R = \mathbb{F}[z_1]$, and so Definition 3.1.2 reduces to that of a convolutional code defined over a univariate polynomial ring in

the sense of [30, 35, 12]. Here, we refer to such a code as a 1-D convolutional code. Since $R = \mathbb{F}[z_1]$ is a PID, every R-submodule of R^n is free. Therefore, for $m = 1$ the term "free" can be dropped from the definition. This is no longer true for $m > 1$. The multivariate polynomial ring $R = \mathbb{F}[z_1, \ldots, z_m]$ is a *noetherian ring* but not a PID, and so every R-submodule of R^n is finitely generated but is not necessarily free. The spanning set of a finitely generated R-submodule can be used to construct the rows of its generator matrix. But if an R-submodule of R^n is not free, it will not admit an m-D convolutional encoder. This implies that it is not possible to inject elements from any input space R^k; $1 \leq k < n$, into the R-submodule, and for this reason we cannot use it as a convolutional code.

Given any generator matrix $G(z)$ that is not of full row rank, it is natural to wonder whether it generates an m-D convolutional code. That is, if $G(z)$ is not of full row rank, then what properties of $G(z)$ characterize $\mathcal{C} = \mathrm{im}(G(z))$ as a free R-submodule? Before we proceed, we need the following elementary lemma.

Lemma 3.1.1. *Let $G(z) \in R^{k\times n}$ and $G_1(z) \in R^{l\times n}$ be two generator matrices. Then,* $\mathrm{im}(G(z)) \subseteq \mathrm{im}(G_1(z))$ *if and only if there exists a $T(z) \in R^{k\times l}$ such that $G(z) = T(z)G_1(z)$.*

Proof. If $G(z) = T(z)G_1(z)$, then every row of $G(z)$ can be written as an R-linear combination of the rows of $G_1(z)$. Therefore, $\mathrm{im}(G(z)) \subseteq \mathrm{im}(G_1(z))$.

For the converse, assume $\mathrm{im}(G(z)) \subseteq \mathrm{im}(G_1(z))$. Let $g_x(z)$ be the xth row of $G(z)$. Since $\mathrm{im}(G(z)) \subseteq \mathrm{im}(G_1(z))$, we can write $g_x(z) = u_x(z)G_1(z)$ for some $u_x(z) \in R^l$. Form a matrix $T(z) = [u_1(z), \ldots, u_k(z)]^T \in R^{k\times l}$ such that $G(z) = T(z)G_1(z)$ as desired. □

The following proposition answers our question. A proof of this in the context of finite support behaviors over the Laurent polynomial ring can be found in [20].

Proposition 3.1.2. *Let $G(z) \in R^{k\times n}$ be a generator matrix that has rank l, where $1 \leq l < k$. Let $\mathcal{C} = \mathrm{im}(G(z))$ be the R-submodule of R^n generated by $G(z)$. $\mathcal{C}$ is free if and only if $G(z)$ factors as*

$$G(z) = T(z)G_1(z),$$

such that $T(z) \in R^{k\times l}$ has a left inverse $T(z)^{-1} \in R^{l\times k}$ and $G_1(z) \in R^{l\times n}$ has rank *l. Moreover, $\mathcal{C}$ is a m-D convolutional code of rate $\frac{l}{n}$ and $G_1(z)$ is an m-D convolutional encoder for $\mathcal{C}$.*

Proof. Assume $G(z) = T(z)G_1(z)$ such that $T(z) \in R^{k\times l}$ has a left inverse and $G_1(z) \in R^{l\times n}$ has rank l. From Lemma 3.1.1, $\mathrm{im}(G(z)) \subseteq \mathrm{im}(G_1(z))$. We also have $G_1(z) = T(z)^{-1}G(z)$,

which implies $\text{im}(G_1(z)) \subseteq \text{im}(G(z))$. So, $\mathcal{C} = \text{im}(G(z)) = \text{im}(G_1(z))$ is free and has rank l, because it is generated by a full row rank matrix $G_1(z)$. Therefore, $\mathcal{C}$ is a m-D convolutional code of rate $\frac{l}{n}$ and $G_1(z)$ is an m-D convolutional encoder for $\mathcal{C}$.

Conversely, assume $\mathcal{C} = \text{im}(G(z))$ is a free R-submodule of R^n. Let the rank of $\mathcal{C}$ be l. Since $\mathcal{C}$ has l elements in its basis, it is a m-D convolutional code of rate $\frac{l}{n}$ and admits an m-D convolutional encoder. Let this encoder be $G_1(z) \in R^{l\times n}$. So, $\mathcal{C} = \text{im}(G_1(z))$. Consequently, by virtue of Lemma 3.1.1, there exists two matrices $T(z) \in R^{k\times l}$ and $T_1(z) \in R^{l\times k}$, such that $G(z) = T(z)G_1(z)$ and $G_1(z) = T_1(z)G(z)$. Therefore $G_1(z) = T_1(z)T(z)G_1(z)$. Since $G_1(z)$ is an encoder, it has full row rank, and so $T_1(z)T(z) = I_l$ (the $l \times l$ identity matrix), which implies that $T_1(z)$ is the left inverse of $T(z)$. □

While a convolutional encoder $G(z)$ uniquely identifies the convolutional code $\mathcal{C} = \text{im}(G(z))$, the converse does not hold, as $\mathcal{C}$ can be described as the image of different encoders. So it is natural to investigate the conditions under which two encoders generate the same code.

Definition 3.1.3. *Two m-D convolutional encoders $G(z)$ and $G_1(z)$ are called equivalent if they generate the same m-D convolutional code. That is, $G(z)$ and $G_1(z)$ are equivalent encoders if* $\text{im}(G(z)) = \text{im}(G_1(z))$.

Definition 3.1.4. *A square matrix $U(z) \in R^{k\times k}$ is called* unimodular *if $U(z)$ is a unit in $R^{k\times k}$. In other words, $U(z)$ is unimodular if and only if* $\det(U(z))$ *is a unit in R.*

The proof of the following proposition from [57, 56] follows along the same lines of Proposition 3.1.2.

Proposition 3.1.3. *Two m-D convolutional encoders $G(z), G_1(z) \in R^{k\times n}$ are equivalent if and only if there exists a unimodular matrix $U(z) \in R^{k\times k}$ such that $G(z) = U(z)G_1(z)$.*

We will end this section with the following examples to clarify the above ideas.

Example 3.1.1. Consider the generator matrix $G(z) \in R^{2\times 3}$ where $R = \mathbb{F}_2[z_1, z_2]$ as shown below.

$$G(z) = \begin{bmatrix} 1 + z_1^2 & 0 & z_1 \\ 1 + z_2 & 1 + z_1 + z_2^2 & 0 \end{bmatrix}$$

We have $\text{rank}(G(z)) = 2$. So, $G(z)$ is a 2-D convolutional encoder and $\mathcal{C} = \text{im}(G(z))$ is a 2-D convolutional code of rate $\frac{2}{3}$. ▲

Example 3.1.2. Consider the generator matrix $G(z) \in R^{2\times 3}$ where $R = \mathbb{F}_2[z_1, z_2]$ as shown below.

$$G(z) = \begin{bmatrix} 1 + z_1 + z_1z_2 + z_1^2z_2 & z_1 + z_1^2z_2 & z_2 + z_1z_2^2 \\ z_1z_2 + z_1^2z_2 & z_1^2z_2 & z_1z_2^2 \end{bmatrix}$$

Since all the 2×2 minors of $G(z)$ are zero, we have $\text{rank}(G(z)) = 1$. Let $\mathcal{C} = \text{im}(G(z))$ be the R-submodule of R^3 generated by $G(z)$. Notice that $G(z)$ factors as $G(z) = T(z)G_1(z)$, with

$$T(z) = \begin{bmatrix} 1 + z_1z_2 \\ z_1z_2 \end{bmatrix} \quad \text{and} \quad G_1(z) = \begin{bmatrix} 1 + z_1 & z_1 & z_2 \end{bmatrix},$$

such that $T(z) \in R^{2\times 1}$ has a left inverse $T(z)^{-1} = [1 \quad 1] \in R^{1\times 2}$ and $G_1(z) \in R^{1\times 3}$ is of full row rank. By Proposition 3.1.2, $\mathcal{C} = \text{im}(G(z)) = \text{im}(G_1(z))$ is free and is therefore a 2-D convolutional code of rate $\frac{1}{3}$ generated by the encoder $G_1(z)$. ▲

Example 3.1.3. This example was used by Weiner in [57]. Let $\mathcal{C} = \text{im}(G(z))$ be an R-submodule of R^3 generated by

$$G(z) = \begin{bmatrix} z_1 & z_1^2 & z_1z_2 \\ z_2 & z_1z_2 & z_2^2 \end{bmatrix}.$$

It can easily be verified that $\text{rank}(G(z)) = 1$. But, $G(z)$ does not satisfy Proposition 3.1.2. To see this, factor $G(z)$ as $G(z) = T(z)G_1(z)$ such that $T(z) = [a \quad b]^T \in R^{2\times 1}$ and $G_1(z) = [f \quad g \quad h] \in R^{1\times 3}$. Now $z_1 = af$ and $z_2 = bf$, and so $f = 1$ since $f, a, b \in R$. This implies, $T(z) = [z_1 \quad z_2]^T$. But clearly, $T(z)^{-1} \notin R^{1\times 2}$. Therefore, by Proposition 3.1.2 $\mathcal{C} = \text{im}(G(z))$ is not a 2-D convolutional code. ▲

3.2 Distance Parameters

Here we define the distance of a m-D convolutional code. This is a straightforward generalization of the notion of distance in the 1-D case.

Definition 3.2.1. *Let $v \in S^n$ be a m-D composite sequence over $\mathbb{F}$.*

$$\begin{array}{rccc} v: & \mathbb{N}^m & \to & \mathbb{F}^n \\ & (i_1, \ldots, i_m) & \mapsto & v^{(1)}_{(i_1,\ldots,i_m)}, \ldots, v^{(n)}_{(i_1,\ldots,i_m)} \end{array}$$

The weight of v, denoted by $\text{wt}(v)$, is given by number of nonzero symbols $v^{(j)}_{(i_1,\ldots,i_m)}$ in the sequence v.

Equivalently, if $v(z) = \psi^n(v)$ is the polynomial representation of the sequence v, then the weight of the polynomial vector $v(z) = [v^{(1)}(z), \ldots, v^{(n)}(z)] \in R^n$ is given by

$$\mathrm{wt}(v(z)) = \sum_{j=1}^{n} \mathrm{wt}(v^{(j)}(z)),$$

where $\mathrm{wt}(v^{(j)}(z))$ is the number of nonzero terms in the polynomial $v^{(j)}(z) \in R$.

Example 3.2.1. Consider the 2-D composite sequence from Example 2.0.2 as shown below

$$\begin{array}{c} u \in S^3 \\ \begin{matrix} 101 & 010 \\ 001 & 100 \\ 010 & 001 \\ 100 & 110 \end{matrix} \end{array} \quad \overset{\psi^3}{\longmapsto} \quad \begin{array}{c} u(z) \in R^3 \\ \begin{bmatrix} 1 + z_1 z_2 + z_2^3 + z_1 z_2^3 \\ z_1 + z_2^2 + z_1 z_2^3 \\ 1 + z_2 + z_1 z_2^2 \end{bmatrix}^T \end{array}$$

Here, $\mathrm{wt}(u) = \mathrm{wt}(u(z)) = 10$. The weight can be obtained by counting either the number of nonzero symbols of the sequence u or by counting the number of terms in the vector $u(z)$.

Definition 3.2.2. *Let $\mathcal{C}$ be a m-D convolutional code. The Hamming distance between two codewords $v_1, v_2 \in \mathcal{C}$ is given by* $\mathrm{dist}(v_1, v_2) = \mathrm{wt}(v_1 - v_2)$.

The weight of a codeword $v \in \mathcal{C}$ is also the Hamming distance between v and the all-zero codeword $\mathbf{0}$. i.e. $\mathrm{wt}(v) = \mathrm{dist}(v, \mathbf{0})$. The Hamming distance gives a metric on S^n (or R^n) called the Hamming metric.

Definition 3.2.3. *The distance of a m-D convolutional code $\mathcal{C}$ is defined as*

$$\begin{aligned} \mathrm{dist}(\mathcal{C}) &= \min\{\mathrm{dist}(v_i, v_j) : \ v_i, v_j \in \mathcal{C}, \ v_i \neq v_j\} \\ &= \min\{\mathrm{wt}(v) : \ v \in \mathcal{C}, \ v \neq \mathbf{0}\}. \end{aligned}$$

The distance of the code is the minimum hamming distance between any two distinct codewords. By linearity of the code, the distance of the code is also the weight of the minimum weight nonzero codeword. If $d = \mathrm{dist}(\mathcal{C})$, then we say that the code $\mathcal{C}$ can correct up to $\lfloor \frac{d-1}{2} \rfloor$ errors.

3.3 Primeness Properties of Encoders

In this section we will explore the different primeness notions that arise when considering multivariate polynomial matrices as m-D convolutional encoders. These primeness

notions are not differentiated in 1-D convolutional codes because they are all equivalent in the 1-D case and arise only for multivariate polynomial matrices.

Definition 3.3.1. *A m-D convolutional encoder $G(z) \in R^{k\times n}$ is said to be:*

i) left factor prime (LFP) if whenever $G(z)$ is factored as $G(z) = T(z)G_1(z)$ with $T(z) \in R^{k\times k}$ and $G_1(z) \in R^{k\times n}$, then $T(z)$ is unimodular;

ii) left minor prime (LMP) if the $k \times k$ minors of $G(z)$ have no common divisors in R except for units;

iii) left zero prime (LZP) if the $k \times k$ minors of $G(z)$ generate the ring R as an ideal.

For full column rank matrices $G(z) \in R^{k\times n}$ with $k \geq n$, RFP, RMP and RZP can be similarly defined. Since our encoding operation is defined (3.3) as left multiplication with $G(z) \in R^{k\times n}$, where $k < n$, we will drop the prefix "L" and denote the primeness notions by FP, MP and ZP.

From the above definition, it is clear that these primeness notions follow the hierarchy ZP⇒MP⇒FP. For $m = 1$, it is well known [31, 47] that these notions are equivalent ZP≡MP≡FP due to the fact that in a PID the implications can be reversed as FP⇒ZP. For $m = 2$, it has been shown that FP⇒MP [44], whereas for $m > 2$ none of the implications can be reversed [60]. These results can be summarized using the following theorem by Youla and Gnavi [60].

Theorem 3.3.1. *Let $G(z) \in R^{k\times n}$ be an m-D convolutional encoder.*

i) If $m = 1$, then ZP≡MP≡FP.

ii) If $m = 2$, then ZP≢MP≡FP.

iii) If $m \geq 3$, then ZP≢MP≢FP.

iv) Always, ZP⇒MP⇒FP.

The nonequivalence of MP and ZP for $m > 1$ has important consequences with regard to generalizing well known concepts of 1-D convolutional codes to the m-D case. The remainder of this chapter will focus on this nontrivial generalization.

3.4 Encoder Inverses

Consider the use of an m-D convolutional encoder $G(z) \in R^{k\times n}$ in a communication system. Given an input vector $u(z) \in R^k$, the encoder generates a codeword $v(z) = u(z)G(z) \in \mathcal{C}$, which is transmitted over a noisy channel. At the output of the channel the decoder operates on the received vector $r(z) \in R^n$ to produce an estimate $\tilde{u}(z) \in R^k$ of the input vector $u(z) \in R^k$. The decoder can be split conceptually into two parts: a *codeword estimator* which produces an estimate $\tilde{v} \in \mathcal{C}$ of the codeword v, followed by an *encoder inverse* that recovers the input estimate $\tilde{u}$ from $\tilde{v}$. The encoder inverse is a $n \times k$ right inverse matrix $G^{-1}(z)$ of $G(z)$. That is,

$$G(z)G^{-1}(z) = I_k, \tag{3.4}$$

where I_k is the $k \times k$ identity matrix. In general, the encoder inverse may be rational (i.e. $G^{-1}(z) \in Q^{n\times k}$, where $Q = \mathbb{F}(z_1, \ldots, z_m)$ is the field of fractions of R) and not necessarily a polynomial matrix.

The proof of the following proposition uses the Quillen-Suslin theorem [34] and the Cauchy-Binet formula [25]. A proof of this proposition can be found in the paper by Youla and Pickle [61]. Weiner [56, 57] explains it in the context of m-D convolutional codes.

Theorem 3.4.1. *A m-D convolutional encoder $G(z) \in R^{k\times n}$ has a polynomial right inverse $G^{-1}(z) \in R^{n\times k}$ if and only if it is ZP.*

Proof. Assume $G(z) \in R^{k\times n}$ has a polynomial right inverse $G^{-1}(z) \in R^{n\times k}$. Then,

$$G(z)G^{-1}(z) = I_k. \tag{3.5}$$

Let Λ be a subset of $\{1, \ldots, n\}$ with k elements. We denote by $G_\Lambda(z)$ the $k \times k$ matrix whose columns are those columns of $G(z)$ that have indices from Λ. Similarly, we denote by $\bar{G}_\Lambda(z)$ the $k \times k$ matrix whose rows are those rows of $G^{-1}(z)$ that have indices from Λ. By applying the Cauchy-Binet formula to calculate the determinant in (3.5), we get

$$\det(G(z)G^{-1}(z)) = \sum_{\Lambda} \det(G_\Lambda(z)) \det(\bar{G}_\Lambda(z)) = 1 \tag{3.6}$$

where the sum extends over all possible subsets Λ of $\{1, \ldots, n\}$ with k elements. There are $\binom{n}{k}$ such subsets. Now, (3.6) implies that some R-linear combination of the full size minors

of $G(z)$ is equal to 1. Therefore, the full size minors of $G(z)$ generate R as an ideal, and so $G(z)$ is ZP.

For the converse, assume $G(z)$ is ZP. By the Quillen-Suslin theorem, $G(z)$ can be row bordered by a matrix $A(z) \in R^{(n-k)\times n}$ such that $[G(z) \;\; A(z)]^T \in R^{n\times n}$ is unimodular. Consequently, there exists a matrix $[B(z) \;\; D(z)] \in R^{n\times n}$, with $B(z) \in R^{n\times k}$ and $D(z) \in R^{n\times(n-k)}$, such that

$$\begin{bmatrix} G(z) \\ A(z) \end{bmatrix} \begin{bmatrix} B(z) & D(z) \end{bmatrix} = I_n. \tag{3.7}$$

From (3.7) it follows that $G(z)B(z) = I_k$, and so the polynomial matrix $B(z)$ is a right inverse for $G(z)$. □

Remark 3.4.1. A 1-D polynomial encoder that has a polynomial right inverse is often referred to as a *basic* encoder in the literature [22, 30, 42].

For $m = 1$, MP is equivalent to ZP (see Theorem 3.3.1). Therefore, it is sufficient to test for minor primeness to see if a 1-D convolutional encode is basic. However, this does not apply for the case $m \geq 2$. Consider for example a 2-D encoder $G(z) = [z_1 \;\; z_2] \in R^{1\times 2}$, where $R = \mathbb{F}_2[z_1, z_2]$. Clearly, $G(z)$ is MP since $\gcd(z_1, z_2) = 1$. But $G(z)$ is not ZP because z_1 and z_2 cannot generate R as an ideal. To see this, observe that the constant $1 \in R$ cannot be generated by a linear combination of z_1 and z_1. Therefore, from the above theorem it follows that $G(z)$ is not a basic encoder.

3.5 Parity Check Matrices and Dual Codes

Every R-submodule of the free module R^n has an orthogonal module associated with it. In this section we will explore the properties of this orthogonal module and derive from it the notion of a parity check matrix and dual code associated with a given m-D convolutional code.

Definition 3.5.1. *Let $\mathcal{M}$ be an R-submodule of R^n. A vector $p(z) \in R^n$ is said to be orthogonal to $\mathcal{M}$ if $v(z)p(z)^T = 0$ for all $v(z) \in \mathcal{M}$.*

Let $\mathcal{M}^\perp$ be the set of all vectors in R^n that are orthogonal to $\mathcal{M}$. Clearly, $\mathcal{M}^\perp$ is nonempty because the n-tuple zero vector is in $\mathcal{M}^\perp$. If $v(z) \in \mathcal{M}$, $\alpha \in R$ and $p_1(z), p_2(z) \in \mathcal{M}^\perp$, then $v(z)(\alpha\, p_1(z) + p_2(z))^T = \alpha\, v(z)p_1(z)^T + v(z)p_2(z)^T = 0$ implies that $\alpha\, p_1(z) +$

$p_2(z) \in \mathcal{M}^\perp$. Therefore, $\mathcal{M}^\perp$ is an R-submodule of R^n. We will refer to $\mathcal{M}^\perp$ as the orthogonal module of $\mathcal{M}$.

Definition 3.5.2. *The orthogonal module of $\mathcal{M}$, denoted by $\mathcal{M}^\perp$, is defined as*

$$\mathcal{M}^\perp = \left\{p(z) \in R^n : v(z)p(z)^T = 0 \text{ for all } v(z) \in \mathcal{M}\right\}. \tag{3.8}$$

We note here that since R is a noetherian ring, $\mathcal{M}^\perp$ is finitely generated. Therefore, there exists a generator matrix $D(z) \in R^{j\times n}$ such that $\mathcal{M}^\perp = \text{im}(D(z))$.

Definition 3.5.3. *Let $\mathcal{M}$ be an R-submodule of R^n. A matrix $H(z) \in R^{l\times n}$ is a parity check matrix for $\mathcal{M}$ if $\mathcal{M} = \left\{v(z) \in R^n : v(z)H(z)^T = 0\right\}$. This gives $\mathcal{M}$ as the left kernel of the transpose of the parity check matrix. That is, $\mathcal{M} = \ker(H(z)^T)$.*

The parity check matrix is useful in determining if a given vector belongs to the R-submodule $\mathcal{M}$. If $H(z)$ is a parity check matrix for $\mathcal{M}$, then a vector $v(z) \in R^n$ is also in $\mathcal{M}$ if and only if $v(z)H(z)^T = 0$.

Lemma 3.5.1. *Any generator matrix of $\mathcal{M}$ is a parity check matrix for $\mathcal{M}^\perp$. That is, if $\mathcal{M} = \text{im}(G(z))$ for some $G(z) \in R^{k\times n}$, then $\mathcal{M}^\perp = \ker(G(z)^T)$.*

Proof. Let $p(z) \in \mathcal{M}^\perp$. By definition $p(z)v(z)^T = 0$ for all $v(z) \in \mathcal{M}$. Since $v(z) \in \text{im}(G(z))$, it follows that $p(z)G(z)^T = 0$. So $p(z) \in \ker(G(z)^T)$. Hence, $\mathcal{M}^\perp \subseteq \ker(G(z)^T)$.

On the other hand, if $p(z) \in \ker(G(z)^T)$, then $p(z)G(z)^T = 0$. For any $v(z) \in \mathcal{M}$, we have $v(z) = u(z)G(z)$ for some $u(z) \in R^k$. Then $p(z)v(z)^T = p(z)G(z)^T u(z)^T = 0$. So $p(z) \in \mathcal{M}^\perp$. Hence, $\ker(G(z)^T) \subseteq \mathcal{M}^\perp$. □

From Lemma 3.5.1 we obtain the following properties of $\mathcal{M}^\perp$.

Proposition 3.5.2. $\mathcal{M} \subseteq \mathcal{M}^{\perp\perp}$.

Proof. Let $\mathcal{M} = \text{im}(G(z))$ and $\mathcal{M}^\perp = \text{im}(D(z))$. By Lemma 3.5.1, $\mathcal{M}^\perp = \ker(G(z)^T)$ and $\mathcal{M}^{\perp\perp} = \ker(D(z)^T)$. Since $\mathcal{M}^\perp = \text{im}(D(z)) = \ker(G(z)^T)$, we have $D(z)G(z)^T = 0$. Now $G(z)D(z)^T = (D(z)G(z)^T)^T = 0$, implies $\text{im}(G) \subseteq \ker(D(z)^T)$. Hence, $\mathcal{M} \subseteq \mathcal{M}^{\perp\perp}$. □

Proposition 3.5.3. $\mathcal{M}^\perp = \mathcal{M}^{\perp\perp\perp}$.

Proof. Let $\mathcal{M} = \text{im}(G(z))$, $\mathcal{M}^\perp = \text{im}(D(z))$, and $\mathcal{M}^{\perp\perp} = \text{im}(F(z))$. By Lemma 3.5.1, we have $\mathcal{M}^\perp = \ker(G(z)^T)$, $\mathcal{M}^{\perp\perp} = \ker(D(z)^T)$, and $\mathcal{M}^{\perp\perp\perp} = \ker(F(z)^T)$.

Since $\mathcal{M}^{\perp\perp} = \text{im}(F(z)) = \ker(D(z)^T)$, we have $F(z)D(z)^T = 0$. Now $D(z)F(z)^T = (F(z)D(z)^T)^T = 0$, implies $\text{im}(D(z)) \subseteq \ker(F(z)^T)$. Therefore, $\mathcal{M}^{\perp} \subseteq \mathcal{M}^{\perp\perp\perp}$.

From Proposition 3.5.2, $\mathcal{M} \subseteq \mathcal{M}^{\perp\perp}$. Therefore, by Lemma 3.1.1 there exists an R-matrix $T(z)$ of appropriate dimensions such that $G(z) = T(z)F(z)$. Now, if $p(z) \in \mathcal{M}^{\perp\perp\perp}$, then $p(z)F(z)^T = 0$. For any $v(z) \in \mathcal{M}$, we have $v(z) = u(z)G(z)$ for some polynomial vector $u(z)$ in the input space of $G(z)$. Now, $p(z)v(z)^T = p(z)(u(z)G(z))^T = p(z)F(z)^T T(z)^T u(z)^T = 0$, implies that $p(z) \in \mathcal{M}^{\perp}$. Therefore, $\mathcal{M}^{\perp\perp\perp} \subseteq \mathcal{M}^{\perp}$. □

From Lemma 3.5.1 we know that any generator matrix of $\mathcal{M}$ is a parity check matrix for its orthogonal module $\mathcal{M}^{\perp}$. In general, by virtue of Proposition 3.5.2, $\mathcal{M} \subseteq \mathcal{M}^{\perp\perp}$ always holds. But, if $\mathcal{M} = \mathcal{M}^{\perp\perp}$, then the following proposition shows that any generator matrix of $\mathcal{M}^{\perp}$ is a parity check matrix for $\mathcal{M}$.

Proposition 3.5.4. *Let $\mathcal{M} = \mathcal{M}^{\perp\perp}$. Then, any generator matrix of $\mathcal{M}^{\perp}$ is a parity check matrix for $\mathcal{M}$.*

Proof. Let $\mathcal{M} = \mathcal{M}^{\perp\perp}$. Let $\mathcal{M}^{\perp} = \text{im}(D(z))$. We have $\mathcal{M}^{\perp\perp} = \ker(D(z)^T) = \mathcal{M}$. Therefore, $D(z)$ is a parity check matrix for $\mathcal{M}$. □

We formalize this notion of duality as follows:

Definition 3.5.4. *Let $\mathcal{M}$ be an R-submodule of R^n. Let $\mathcal{M}^{\perp}$ be the orthogonal module of $\mathcal{M}$. $\mathcal{M}^{\perp}$ is called the dual module of $\mathcal{M}$ if $\mathcal{M} = \mathcal{M}^{\perp\perp}$.*

Proposition 3.5.5. *$\mathcal{M}^{\perp}$ is the dual module of $\mathcal{M}$ if and only if $\mathcal{M}$ has a parity check matrix.*

Proof. Assume $\mathcal{M}$ has a parity check matrix $H(z)$. So, $\mathcal{M} = \ker(H(z)^T)$. Let $\mathcal{D} = \text{im}(H(z))$ be the R-module of generated by $H(z)$. Then $\mathcal{D}^{\perp} = \ker(H(z)^T)$, and so $\mathcal{M} = \mathcal{D}^{\perp}$. From Proposition 3.5.3, we have $\mathcal{M}^{\perp\perp} = \mathcal{D}^{\perp\perp\perp} = \mathcal{D}^{\perp} = \mathcal{M}$. Therefore, $\mathcal{M}^{\perp}$ is the dual module of $\mathcal{M}$.

The converse follows from Proposition 3.5.4. □

Application to m-D Convolutional Codes

The above properties apply to any R-submodule of R^n. Recall here, from the discussion in Section 3.1, that every R-submodule of R^n is finitely generated, but we only

consider the free R-submodules of R^n as m-D convolutional codes. Next, we discuss these results in context of m-D convolutional codes.

For the case $m = 1$, since $R = \mathbb{F}[z_1]$ is a PID, all submodules of R^n are free. Therefore, by Definition 3.1.2 any R-submodule $\mathcal{C}$ of R^n is a 1-D convolutional code and its orthogonal module $\mathcal{C}^\perp$ is also a 1-D convolutional code.

Proposition 3.5.6. *Let $R = \mathbb{F}[z_1, z_2]$. Let $\mathcal{C}$ be an R-submodule of R^n. Then, $\mathcal{C}^\perp$ is free.*

A proof of the above proposition is given by Weiner in [56]. It shows that for the case $m = 2$, the orthogonal module $\mathcal{C}^\perp$ is a 2-D convolutional code even if $\mathcal{C}$ is not a free module.

The following result was first proved by Wood, Rogers, and Owens [58] for rings of characteristic zero. Weiner [56] has generalized this result to any noetherian unique factorization domain without the restriction to characteristic zero. This is of consequence to us because the ground field over which codes are defined is always required to be finite. An alternate proof for this result in the context of finite support behaviors over the Laurent polynomial ring is given by Fornasini and Valcher in [20].

Proposition 3.5.7. *Let $\mathcal{C}$ be a m-D convolutional code. $\mathcal{C}$ has a parity check matrix if and only if $\mathcal{C}$ has a MP encoder.*

The existence of a MP encoder for a code $\mathcal{C}$ is desirable because it allows us to check if a vector $v(z) \in R^n$ is a codeword using the parity check matrix $H(z)$. That is, $v(z) \in \mathcal{C}$ if and only if $v(z)H(z)^T = 0$. The matrix $H(z)^T$ is referred to as the *syndrome former* because it generates a nonzero polynomial vector $s(z) = r(z)H(z)^T$ know as the *error syndrome* if $r(z) = (v(z)+e(z)) \notin \mathcal{C}$. The information contained in the error syndrome $s(z)$ can be used to detect the error $e(z)$.

From Proposition 3.5.5 we know that an R-submodule of R^n has a dual module if and only if it has a parity check matrix. When this result is applied to m-D convolutional codes using Proposition 3.5.7 we have the following.

Proposition 3.5.8. *Let $\mathcal{C}^\perp$ be the orthogonal module of a m-D convolutional code $\mathcal{C}$. $\mathcal{C}^\perp$ is the dual module of $\mathcal{C}$ if and only if $\mathcal{C}$ has a MP encoder.*

Definition 3.5.5. *Let $\mathcal{C}^\perp$ be the dual module of an m-D convolutional code $\mathcal{C}$. If $\mathcal{C}^\perp$ is free, it is called the dual code of $\mathcal{C}$.*

For 1-D and 2-D convolutional codes that have MP encoders, the existence of a dual code is not an issue. This is because all orthogonal modules are free for $m \leq 2$ (see Proposition 3.5.6 above). This does not hold for MP m-D convolutional encoders in general. The following example taken from [56, 57, 62] shows a 3-D convolutional code with a MP encoder whose dual module is not free.

Example 3.5.1. Let $R = \mathbb{F}_2[z_1, z_2, z_3]$. Consider an encoder $G(z) = [z_2 \;\; z_3 \;\; z_1] \in R^{1\times 3}$. Let $\mathcal{C} = \text{im}(G(z))$ be the 3-D convolutional code of rate $\frac{1}{3}$ generated by $G(z)$. Let $\mathcal{C}^\perp$ be the orthogonal module of $\mathcal{C}$. By Lemma 3.5.1, $\mathcal{C}^\perp = \ker(G(z)^T)$ and so $G(z)$ is a parity check matrix for $\mathcal{C}^\perp$.

Clearly, $G(z)$ is MP. By Proposition 3.5.7, $\mathcal{C}$ has a parity check matrix, and by Proposition 3.5.8, $\mathcal{C}^\perp$ is the dual module of $\mathcal{C}$. It is not hard to see that solving for the null space of $[z_2 \;\; z_3 \;\; z_1]^T$ gives

$$H(z) = \begin{bmatrix} z_1 & 0 & z_2 \\ 0 & z_1 & z_3 \\ z_3 & z_2 & 0 \end{bmatrix}$$

as the generator matrix for the dual module. That is, $\mathcal{C}^\perp = \text{im}(H(z))$. Also, since $\mathcal{C}^\perp$ is the dual module of $\mathcal{C}$, $H(z)$ is a parity check matrix for $\mathcal{C}$.

Observe that $\det(H(z)) = 0$. Therefore, the rows of $H(z)$ are linearly dependent. Although we have $\text{rank}(H(z)) = 2$, $\mathcal{C}^\perp$ clearly cannot be expressed as the row space of less than 3 rows of $H(z)$. Therefore, the dual module $\mathcal{C}^\perp$ is not free. ▲

Example 3.5.2. Consider the 3-D convolutional code $\mathcal{C} = \text{im}(G(z))$ generated by the encoder $G(z) = [z_2 \;\; z_3 \;\; z_1]$ from Example 3.5.1. We know that $\mathcal{C}$ does not have a dual code because its dual module $\mathcal{C}^\perp = \text{im}(H(z))$ generated by

$$H(z) = \begin{bmatrix} z_1 & 0 & z_2 \\ 0 & z_1 & z_3 \\ z_3 & z_2 & 0 \end{bmatrix}$$

is not free. Now, the encoder $G(z)$ is MP. So $\mathcal{C}$ has a parity check matrix. Consider the matrices

$$H_1(z) = \begin{bmatrix} z_1 & 0 & z_2 \\ 0 & z_1 & z_3 \end{bmatrix}, \quad H_2(z) = \begin{bmatrix} z_1 & 0 & z_2 \\ z_3 & z_2 & 0 \end{bmatrix}, \quad H_3(z) = \begin{bmatrix} 0 & z_1 & z_3 \\ z_3 & z_2 & 0 \end{bmatrix}$$

formed from any two rows of $H(z)$. It can be easily verified that the code $\mathcal{C}$ can be written as

$$\mathcal{C} = \ker(H_1(z)^T) = \ker(H_2(z)^T) = \ker(H_3(z)^T) = \ker(H(z)^T),$$

and so $H_1(z), H_2(z), H_3(z)$, and $H(z)$ are parity check matrices for $\mathcal{C}$.

On the other hand, the matrices $H_1(z), H_2(z)$ and $H_3(z)$ are of full row rank. So they are 3-D encoders, and their (distinct) row spaces $\mathcal{C}_1 = \text{im}(H_1(z))$, $\mathcal{C}_2 = \text{im}(H_2(z))$ and $\mathcal{C}_3 = \text{im}(H_3(z))$ are admissible as 3-D convolutional codes. Note here that since $H_1(z), H_2(z)$ and $H_3(z)$ are formed from the rows of $H(z)$, the codes $\mathcal{C}_1, \mathcal{C}_2$ and $\mathcal{C}_3$ are contained in the non-free module $\mathcal{C}^\perp$.

However, the encoders $H_1(z), H_2(z)$ and $H_3(z)$ are not MP because their respective minors have nonunit common factors of z_1, z_2 and z_3. Therefore, the codes $\mathcal{C}_1, \mathcal{C}_2$ and $\mathcal{C}_3$ do not have parity check matrices. In particular, the matrix $G(z)$ is not a parity check matrix for any of them. For example, $[z_1 \;\; 0 \;\; z_2]G(z)^T = 0$ since $[z_1 \;\; 0 \;\; z_2] \in \mathcal{C}^\perp$, but $[z_1 \;\; 0 \;\; z_2] \notin \mathcal{C}_3$ and so $\mathcal{C}_3 \neq \ker(G(z)^T)$. ▲

The key point to observe here is that if a m-D convolutional code $\mathcal{C}$ generated by a MP encoder does not have a dual code (i.e. its dual module $\mathcal{C}^\perp$ is not free), then a parity check matrix for $\mathcal{C}$ need not necessarily generate the dual module $\mathcal{C}^\perp$. In the above example $H(z)$ is a generator matrix for $\mathcal{C}^\perp$, but $H_1(z), H_2(z)$ and $H_3(z)$ are not. There are however, classes of encoders that generate codes for which the dual code exists. As we will see next, these are the class of ZP encoders.

Weiner [56, 57] has proved the following as a corollary to Theorem 3.4.1.

Proposition 3.5.9. *Let $G(z) \in R^{k\times n}$ be a ZP encoder. Then, the rate $\frac{k}{n}$ m-D convolutional code $\mathcal{C}$ generated by $G(z)$ has a rate $\frac{n-k}{n}$ dual code.*

Proof. Since ZP⇒MP (see Theorem 3.3.1), the encoder $G(z)$ is MP. By Proposition 3.5.8, it follows that $\mathcal{C}^\perp$ is the dual module of $\mathcal{C}$. In (3.7), $D(z) \in R^{n\times(n-k)}$ consists of the last $n-k$ columns of the unimodular matrix $[B(z) \;\; D(z)]$. Therefore, $\text{rank}(D(z)) = n-k$. We will show that $\mathcal{C}^\perp = \text{im}(D(z)^T)$, thus showing that the dual module $\mathcal{C}^\perp$ is free and has rank $n-k$ because it is generated by an encoder $D(z)^T \in R^{(n-k)\times n}$ that has rank $n-k$.

By Lemma 3.5.1, $\mathcal{C}^\perp = \ker(G(z)^T)$. From (3.7), $D(z)^T G(z)^T = (G(z)D(z))^T = 0$. This implies $\text{im}(D(z)^T) \subseteq \ker(G(z)^T)$.

Let $p(z) \in \ker(G(z)^T)$. So $p(z)G(z)^T = 0$. From (3.7) we have,

$$I_n = \begin{bmatrix} B(z) & D(z) \end{bmatrix} \begin{bmatrix} G(z) \\ A(z) \end{bmatrix} = B(z)G(z) + D(z)A(z).$$

Now, $p(z) = p(z)I_n = p(z)(B(z)G(z) + D(z)A(z))^T = p(z)(G(z)^T B(z)^T + A(z)^T D(z)^T)$. But $p(z)G(z)^T = 0$, so $p(z) = p(z)A(z)^T D(z)^T$. This implies $p(z) \in \text{im}(D(z)^T)$, and so $\ker(G(z)^T) \subseteq \text{im}(D(z)^T)$. So indeed $\mathcal{C}^\perp = \text{im}(D(z)^T)$. □

3.6 Discussion

As we have seen so far, by virtue of the primeness hierarchy ZP encoders have nice structural properties. Namely, they generate codes which have parity check matrices, they generate codes for which the dual code exist, and they have a polynomial right inverse. It is therefore desirable to have ZP encoders when generating m-D convolutional codes.

Lin [36, 37] introduced the following notion of *reduced minors* in connection with multivariate polynomial matrix factorization problems.

Definition 3.6.1. *Let $G(z) \in R^{k\times n}$ be a m-D convolutional encoder. Let $a(z)_1, \ldots, a(z)_\beta$ be the $k \times k$ minors of $G(z)$, where $\beta = \binom{n}{k}$. Extracting the greatest common divisor d of $a(z)_1, \ldots, a(z)_\beta$ gives*

$$a(z)_i = d(z)b(z)_i, \quad i = 1, \ldots, \beta. \tag{3.9}$$

Then, $b(z)_1, \ldots, b(z)_\beta$ are called the reduced minors of $G(z)$.

The following result has been proved independently by Pommaret [46], Srinivas [52], and Wang and Feng [55] as a solution to what is know as Lin's conjecture [36] or Lin-Bose conjecture [37].

Theorem 3.6.1. *Let $G(z)$, $a(z)_1, \ldots, a(z)_\beta$, d and $b(z)_1, \ldots, b(z)_\beta$ be as in Definition 3.6.1. Then, $G(z)$ admits a zero prime factorization $G(z) = F(z)G_1(z)$, where $F(z) \in R^{k\times k}$ and $G_1(z) \in R^{k\times n}$, such that $\det(F(z)) = d$ and $G_1(z)$ is ZP if and only if the reduced minors $b(z)_1, \ldots, b(z)_\beta$ of $G(z)$ generate R as an ideal.*

Algorithms for such prime factorizations have been proposed in [36, 8, 43]. These factorization techniques will no doubt be useful in selecting m-D convolutional encoders with good structural properties. We note that in Theorem 3.6.1 if d is not a unit, then

$F(z)$ is not unimodular. In such a situation, if $\mathcal{C} = \text{im}(G(z))$ is the m-D convolutional code generated by the encoder $G(z)$, and $\mathcal{C}_1 = \text{im}(G_1(z))$ is the m-D convolutional code generated by the ZP encoder $G_1(z)$, from Lemma 3.1.1 it follows that $\mathcal{C} \subset \mathcal{C}_1$.

On the other hand, if $\mathcal{C} = \text{im}(G(z))$ is a m-D convolutional code generated by a ZP encoder $G(z) \in R^{k\times n}$, then any other encoder for $\mathcal{C}$ will also be ZP. This simply follows from the fact that if $\bar{G}(z) \in R^{k\times n}$ and $G(z)$ are equivalent encoders, then by Proposition 3.1.3 there exists a unimodular $U(z) \in R^{k\times k}$ such that $\bar{G}(z) = U(z)G(z)$. Now, $\bar{G}(z)G^{-1}(z)U^{-1}(z) = I_k$ implies that the polynomial matrix $G^{-1}(z)U^{-1}(z)$ is an right inverse for $\bar{G}(z)$, and so $\bar{G}(z)$ is ZP. It is therefore important to look for encoders with the lowest complexity within the class of equivalent ZP encoders. In the 1-D case such ZP (i.e. basic) encoders were introduced by Forney [22, 26] and are referred to in the literature as *minimal-basic* encoders [30] or *canonical* encoders [42].

Definition 3.6.2. *Let $R = \mathbb{F}[z_1]$. Let $G(z_1) \in R^{k\times n}$ be a 1-D convolutional encoder with elements $g(z_1)_x^{(y)} \in R$. The constraint length of the xth row of the encoder is given by*

$$\nu_x = \max_{1\le y\le n} \{\deg g(z_1)_x^{(y)}\}. \tag{3.10}$$

The maximum of all the constraint lengths is called the memory order M of the encoder. That is,

$$M = \max_{1\le x\le k} \{\nu_x\}. \tag{3.11}$$

The overall constraint length ν of the encoder is given by the sum of all the constraint lengths.

$$\nu = \sum_{x=1}^{k} \nu_x \tag{3.12}$$

Definition 3.6.3. *A 1-D basic encoder is called minimal-basic if its overall constraint length ν is minimal over all equivalent basic encoding matrices.*

The following result due to Forney [22] (see also [30, 29, 42]) captures one of the most important structural properties of 1-D convolutional encoders.

Theorem 3.6.2. *Let $G(z_1) \in R^{k\times n}$ be a 1-D basic encoder with overall constraint length ν. Then, the following statements are equivalent.*

i) $G(z_1)$ is a minimal-basic encoder.

ii) *The maximum degree μ among the $k \times k$ minors of $G(z_1)$ is equal to the overall constraint length ν.*

iii) *If we define the "indicator matrix for the coefficients of highest-degree terms in each row" of $G(z_1)$ by*

$$G_{\nu_x} = \begin{bmatrix} \bar{\nu}_1^{(1)}, & \dots, & \bar{\nu}_1^{(n)} \\ & \vdots & \\ \bar{\nu}_k^{(1)}, & \dots, & \bar{\nu}_k^{(n)} \end{bmatrix}, \tag{3.13}$$

where

$$\bar{\nu}_x^{(y)} = \underset{z_1^{\nu_x}}{\text{coeff}}\, g(z_1)_x^{(y)}, \tag{3.14}$$

and ν_x is the constraint length of the xth row of $G(z_1)$, then G_{ν_x} has rank k.

These encoders are called minimal-basic because they can be realized in hardware by a minimal number of memory (delay) elements over all equivalent basic encoders. The need for minimal realization arises not only out of hardware requirements, but more importantly because its leads to minimal complexity in trellis decoding algorithms. However, for values of $m \geq 2$ the characterization of minimal-basic m-D convolutional encoders is still an open problem [9, 10]. This issue is closely related to the question of minimizing the state-space for m-D systems, a problem which is much harder than finding minimal realizations when $m = 1$. For more information the interested reader is referred to [24, 49, 2, 59] and the references therein.

Chapter 4

A Sequence-Domain Analysis of the Encoding Process

So far we have viewed m-D convolutional codes from an algebraic perspective. In this chapter we switch to the sequence domain by viewing the code which was defined over a multivariate polynomial ring, geometrically, as a set of sequences defined over a multidimensional sequence space.

4.1 Sequence Generators

Let $G(z) \in R^{k\times n}$ be a polynomial generator matrix as defined in (3.1). The m-variate polynomials $g(z)_x^{(y)} \in R$ that make up the R-matrix $G(z)$ can be transformed into m-D sequences $g_x^{(y)} \in S$ using the isomorphism

$$g_x^{(y)} = \psi^{-1}(g(z)_x^{(y)}), \tag{4.1}$$

to construct an S-matrix $G \in S^{k\times n}$. We will refer to this S-matrix G as the *sequence generator*.

$$G = \begin{pmatrix} g_1^{(1)}, & \dots, & g_1^{(n)} \\ & \vdots & \\ g_k^{(1)}, & \dots, & g_k^{(n)} \end{pmatrix} \tag{4.2}$$

The sequence generator $G \in S^{k\times n}$ is an $\mathbb{F}$-linear, right-shift invariant transfer function matrix between the input sequence space S^k and the output sequence space S^n. A input

sequence $u \in S^k$ is encoded into an output sequence $v \in S^n$ by convolution with the sequence generator as follows:

$$\begin{aligned} v &= u * G \\ v^{(1)} &= u^{(1)} * g_1^{(1)} + \cdots + u^{(k)} * g_k^{(1)} \\ &\vdots \\ v^{(n)} &= u^{(1)} * g_1^{(n)} + \cdots + u^{(k)} * g_k^{(n)} \end{aligned} \tag{4.3}$$

The sequences $v^{(1)}, \ldots, v^{(n)} \in S$ are multiplexed to form the composite output sequence $v \in S^n$. The discrete m-D convolution operation $u^{(x)} * g_x^{(y)}$ as defined in (2.5) is given by

$$v^{(y)}_{x(i_1,\ldots,i_m)} = \sum_{l_m=0}^{\deg_m g_x^{(y)}} \cdots \sum_{l_1=0}^{\deg_1 g_x^{(y)}} u^{(x)}_{((i_1-l_1),\ldots,(i_m-l_m))} g^{(y)}_{x(l_1,\ldots,l_m)}, \tag{4.4}$$

where $v^{(y)}_{(i_1,\ldots,i_m)} = \sum_{x=1}^{k} v^{(y)}_{x(i_1,\ldots,i_m)}$ is an element of the sequence $v^{(y)} \in S$ attached to the coordinate $(i_1, \ldots, i_m)$. Since the coordinates of $\mathbb{N}^m$ are nonnegative, we assume

$$u^{(x)}_{((i_1-l_1),\ldots,(i_m-l_m))} \triangleq 0 \quad \text{for all} \quad i_r < l_r.$$

Since output sequences are obtained by the convolution of input sequences with the sequence generator is the reason why these codes are called multidimensional convolutional codes.

The sequence generator $G \in S^{k\times n}$ induces a sequence space homomorphism between the input sequence space S^k and the output sequence space S^n.

$$\begin{array}{ccc} S^k & \xrightarrow{*G} & S^n \\ u & \mapsto & v \end{array} \tag{4.5}$$

The image $\mathcal{C} = im(G) \subset S^n$ of the sequence generator G is a composite sequence subspace (see Definition 2.0.6) of the output sequence space S^n.

$$\mathcal{C} = \{v \in S^n : v = u * G\} \tag{4.6}$$

The transformation between the sequence space domain and the polynomial domain can be viewed with the help of the commutative diagram shown in Figure 4.1. By virtue of the $\mathbb{F}$-isomorphism ψ^n, $\mathcal{C}$ is naturally endowed with a module structure in R^n. Using the composition $\psi^n(v_1 * v_2) = \psi^n(v_1)\psi^n(v_2)$, all operations in the composite sequence subspace $\mathcal{C} \subset S^n$ find immediate counterparts in the R-submodule $\psi^n(\mathcal{C}) \subset R^n$. From Definitions 3.1.1 and 3.1.2 we have the following.

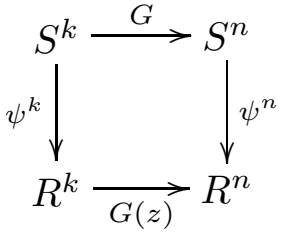


Figure 4.1: Transformation from the sequence space domain to the polynomial domain.

1. A sequence generator $G \in S^{k\times n}$ is an m-D convolutional encoder if the polynomial matrix $G(z) = \psi(G) \in R^{k\times n}$ has rank k.

2. A composite sequence subspace $\mathcal{C} \subset S^n$ is an m-D convolutional code if $\psi^n(\mathcal{C})$ is a free R-submodule of R^n.

Definition 4.1.1. *A m-D convolutional code $\mathcal{C}$ is a composite sequence subspace of S^n that has a basis. If $\mathcal{C}$ has k elements in its basis, then the rate of the code is $\frac{k}{n}$. An encoder for $\mathcal{C}$ is a sequence generator $G \in S^{k\times n}$ that induces an injective map between the input sequence space S^k and the output sequence space S^n.*

We will clarify the encoding operations in the sequence domain and polynomial domain with the following example.

Example 4.1.1. Let $m = 2$ and $R = \mathbb{F}_2[z_1, z_2]$. Consider the polynomial generator matrix

$$G(z) = \begin{bmatrix} z_1^2 z_2 & 0 & z_2 & z_1^2 & 1 & 0 \\ 0 & 1 + z_1^2 z_2 & 0 & z_1 z_2 & z_1 & 1 + z_1^2 + z_2 \end{bmatrix}. \tag{4.7}$$

Since $G(z) \in R^{2\times 6}$ is of full row rank, it is a 2-D convolutional encoder that generates a 2-D convolutional code $\mathcal{C} = \mathrm{im}(G(z)) \subset R^6$ of rate $\frac{2}{6}$. A input polynomial vector $u(z) = \begin{bmatrix} 1 + z_1 z_2 & z_2^2 \end{bmatrix} \in R^2$ is encoded as $v(z) = u(z)G(z)$ to produce the corresponding codeword $v(z) \in \mathcal{C}$.

$$v(z) = \begin{bmatrix} z_1^2 z_2 + z_1^3 z_2^2 & z_2^2 + z_1^2 z_2^3 & z_2 + z_1 z_2^2 & z_1^2 + z_1^3 z_2 + z_1 z_2^3 \\ & & & 1 + z_1 z_2 + z_1 z_2^2 \quad z_2^2 + z_1^2 z_2^2 + z_2^3 \end{bmatrix} \tag{4.8}$$

Let i_1 be the horizontal axis and i_2 be the vertical axis of a 2-D lattice $\mathbb{N}^2$. The sequence encoder $G \in S^{2\times 6}$ is obtained from polynomial encoder $G(z) \in R^{2\times 6}$ using the

transformation $G = \psi^{-1}(G(z))$.

$$G = \begin{pmatrix} 0\,0\,0 & 0\,0\,0 & 0\,0\,0 & 0\,0\,1 & 1\,0\,0 & 0\,0\,0 \\ 0\,0\,1, & 0\,0\,0, & 1\,0\,0, & 0\,0\,0, & 0\,0\,0, & 0\,0\,0 \\ & & & & & \\ 0\,0\,0 & 1\,0\,0 & 0\,0\,0 & 0\,0\,0 & 0\,1\,0 & 1\,0\,1 \\ 0\,0\,0, & 0\,0\,1, & 0\,0\,0, & 0\,1\,0, & 0\,0\,0, & 1\,0\,0 \end{pmatrix} \tag{4.9}$$

The sequence space representation of $u(z) \in R^2$ is a composite sequence $u \in S^2$.

$$\begin{array}{ccccccc} u(z) & & & u & & u^{(1)} & u^{(2)} \\ \left[1 + z_1 z_2 \quad z_2^2\right] & \xrightarrow{(\psi^2)^{-1}} & \begin{matrix} 10 & 00 \\ 00 & 10 \\ 01 & 00 \end{matrix} & \rightsquigarrow & \begin{matrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{matrix}, & \begin{matrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{matrix} \end{array}$$

The input sequence $u \in S^2$ is encoded as $v = u * G$ to obtain the corresponding codeword $v \in \mathcal{C} \subset S^6$ as shown below. Since $m = 2$, the encoding equation (4.4) reduces to

$$v^{(y)}_{x(i_1,i_2)} = \sum_{l_2=0}^{\deg_2 g_x^{(y)}} \sum_{l_1=0}^{\deg_1 g_x^{(y)}} u^{(x)}_{((i_1-l_1),(i_2-l_2))} g^{(y)}_{x(l_1,l_2)},$$

where $v^{(y)}_{(i_1,i_2)} = v^{(y)}_{1(i_1,i_2)} + v^{(y)}_{2(i_1,i_2)}$ is an element of the sequence $v^{(y)} \in S$ attached to the coordinate (i_1, i_2).

$$\begin{array}{ccc} u \in S^2 & & v \in \mathcal{C} \\ \begin{matrix} 10 & 00 \\ 00 & 10 \\ 01 & 00 \end{matrix} & \xrightarrow{*G} & \begin{matrix} 000010 & 000000 & 000100 & 000000 \\ 001000 & 000010 & 100000 & 000100 \\ 010001 & 001010 & 000001 & 100000 \\ 000001 & 000100 & 010000 & 000000 \end{matrix} \end{array} \tag{4.10}$$

The polynomial representation of the (sequence) codeword v in (4.10) using the isomorphism ψ^6 is the same as the (polynomial) codeword $v(z)$ shown in (4.8).

4.2 Encoder Memory and Constraint Lengths

In the introduction we mentioned that a convolutional encoder has memory. This notion of memory is well defined for 1-D convolutional encoders [30, 35, 12]. Here we define constraint lengths and memory orders for m-D convolutional encoders.

In Definition 3.6.2, we defined the constraint length of the xth row of a 1-D encoder as the maximum degree of its components and the memory of the encoder as the maximum

of all the constraint lengths. The notions of *degree* and *length* are synonymous in a 1-D space. A univariate polynomial $u(z_1) = 1 + z_1^3 + z_1^5$ of degree 5 has a corresponding 1-D sequence $u = 1\ 0\ 0\ 1\ 0\ 1$ of length $\deg(u(z_1)) + 1 = 6$. When generalizing the notions of constraint lengths and memory order of an encoder to the m-D case, we have at least two notions of degree.

1. The *total degree* of a multivariate polynomial $u(z) \in R$, denoted by $\operatorname{tdeg} u(z)$, is obtained by adding the exponents of the variables in every nonzero term of $u(z)$, and taking the maximum.

2. The *degree of the variable* z_i in a multivariate polynomial $u(z) \in R$, denoted by $\deg_{z_i} u(z)$, is the largest exponent of z_i occurring among the nonzero terms of $u(z)$.

The use of total degree to describe a parameter like constraint "length" in an m-D space is clearly unsuitable because it collapses into a 1-D quantity and does not provide any information about the lengths in each dimension. For the purpose of illustration, consider a 2-D sequence u as shown below.

$$u = \begin{matrix} 1\,0\,0\,0\,1 \\ 0\,0\,0\,0\,0 \\ 0\,0\,0\,1\,0 \\ 0\,1\,0\,0\,0 \end{matrix} \quad \mapsto \quad u(z) = 1 + z_1^4 + z_1^3 z_2^2 + z_1 z_2^3$$

From the sequence space representation it is clear that the lengths of the sequence u along the i_1 and i_2 dimensions are 5 and 4, respectively. The bivariate polynomial $u(z)$ corresponding to the sequence u has total degree $\operatorname{tdeg} u(z) = 5$, which comes from the term $z_1^3 z_2^2$. But it is the coefficients of the terms z_1^4 and $z_1 z_2^3$ that actually contribute to the lengths 5 and 4 of the sequence u along the i_1 and i_2 dimensions. These lengths can be obtained from the degree of the variable as $5 = \deg_{z_1} u(z) + 1$ and $4 = \deg_{z_2} u(z) + 1$. Notice that we have already used $\deg_{z_i}$ when defining m-D convolution (2.5) and the encoding operation (4.4) in the sequence space domain. That is, $\deg_{z_i} g(z)_x^{(y)} = \deg_i g_x^{(y)}$. We use it here, to generalize the notions of constraint length and memory order to the m-D case as follows.

Definition 4.2.1. *The constraint length of the xth row of a m-D convolutional encoder $G(z) \in R^{k \times n}$ along the ith dimension is defined as*

$$\nu_{i,x} = \max_{1 \le y \le n} \{\deg_{z_i} g(z)_x^{(y)}\}. \tag{4.11}$$

Definition 4.2.2. *The memory order of a m-D convolutional encoder $G(z) \in R^{k\times n}$ along the ith dimension is defined as*

$$M_i = \max_{1\le x\le k} \{\nu_{i,x}\}\,. \tag{4.12}$$

M_i is the maximum of all the constraint lengths in the ith dimension.

For univariate polynomials, deg = tdeg = $\deg_{z_1}$. Therefore for $m = 1$, the above definitions are the same as (3.10) and (3.11), with $\nu_{1,x} = \nu_x$ and $M_1 = M$.

4.3 Sequence Space Ordering

A m-D sequence over $\mathbb{F}$ is defined as a map that associates the coordinates of $\mathbb{N}^m$ with elements of $\mathbb{F}$. In the case of composite sequences in S^k (or S^n) there is no restriction on the way in which the k (or n) elements of $\mathbb{F}$ are *ordered* at each coordinate of the lattice. The symbols of a 1-D composite sequence are ordered along the i_1 dimension of $\mathbb{N}^1$ because it is the only axis in a 1-D space. For values of $m > 1$, one has the option of ordering the symbols of a composite sequence along any of the m axes $i_1, \ldots, i_m$ of the m-D lattice $\mathbb{N}^m$, thus giving rise to different possible sequence orderings. For example, in (4.10) we arbitrarily ordered the symbols of the 2-D output sequence $v \in S^6$ along the i_1 axis as shown in Figure 4.2(a). But, we could just as well have ordered the symbols of the sequence along the i_2 axis as shown in Figure 4.2(b). Similarly, we could have ordered the sequence by attaching 2 symbol along the i_1 axis and 3 symbols along the i_2 axis as shown in Figure 4.2(c), and so on.

Definition 4.3.1. *For a given sequence generator $G \in S^{k\times n}$, if the parameters k and n are factored as $k = \prod_{i=1}^{m} k_i$ and $n = \prod_{i=1}^{m} n_i$ (where k_i, n_i are positive integers), then $\prod_{i=1}^{m} k_i$ defines the ordering of the input sequence space S^k and $\prod_{i=1}^{m} n_i$ defines the ordering of the output sequence space S^n. The input sequences are ordered by attaching k_i symbols, and the output sequences are ordered by attaching n_i symbols, to each coordinate of the lattice $\mathbb{N}^m$ along the ith dimension of their respective sequence spaces.*

The singular dimension of a 1-D space is usually associated with the time axis. For a 1-D encoder the parameter M_1 is called "memory" because at any time instant, say i_1, an n-symbol encoded block attached to the coordinate (i_1) of the 1-D encoded sequence is a function not only of the corresponding k-symbol input block attached to the coordinate (i_1)

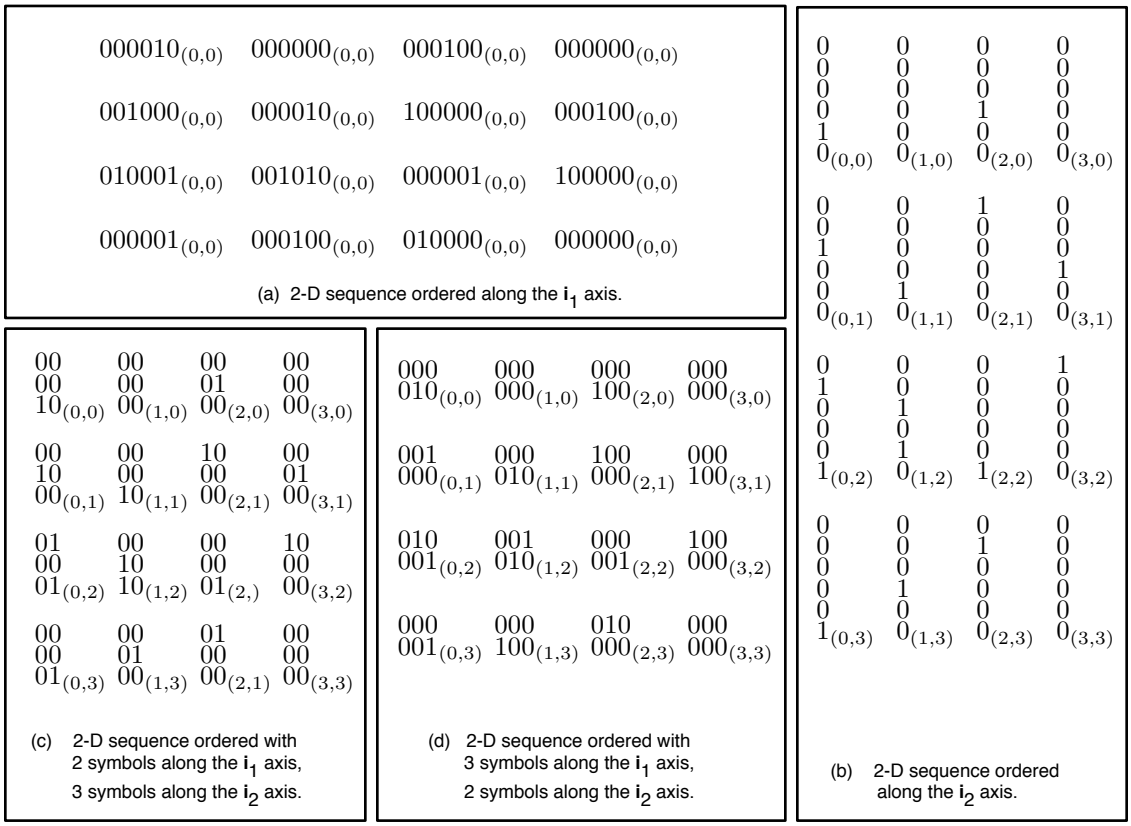


Figure 4.2: Sequence Ordering.

but also M_1 "previous" k-symbol input blocks attached to the coordinates $(i_1 - 1)$ through $(i_1 - M_1)$ of the 1-D input sequence. The notions of *past* and *future* that are natural in a 1-D space do not generalize in a straightforward manner to higher dimensions. Therefore, the term "memory" which is associated with the past in a 1-D space cannot be interpreted using time in an m-D space for values of $m > 1$.

The sequence space ordering defined above can be used to interpret the notion of memory of a m-D convolutional encoder as follows: Let $M_1, \ldots, M_m$ be the memory orders of a rate $\frac{k}{n}$ m-D convolutional encoder. For a given choice of sequence ordering as described in Definition 4.3.1, an n-symbol encoded block attached to the coordinate $(l_1, \ldots, l_m)$ of the encoded sequence $v \in S^n$ is a function of $k_i(M_i + 1)$ input symbols that start at $l_i - M_i$ and end at l_i along the ith dimension of the input sequence $u \in S^k$. This accounts for a total of $\prod_{i=1}^{m} k_i(M_i + 1)$ input symbols of the input sequence $u \in S^k$ with order $\prod_{i=1}^{m} k_i$ that affect an n-symbol block of the encoded sequence $v \in S^n$ with order $\prod_{i=1}^{m} n_i$.

Note 4.3.1. The condition $u^{(x)}_{((i_1-l_1),(i_2-l_2))} \triangleq 0$ for all $i_r < l_r$ in (4.4) implies that the boundaries of the sequence $u^{(x)} \in S$ have to be padded with M_i zeros along the ith dimension. For a composite input sequence $u \in S^k$ with order $\prod_{i=1}^{m} k_i$, the boundaries have to be padded with $k_i M_i$ zeros in the ith dimension during the encoding operation $v = u * G$. This is equivalent to zeroing out the contents of shift registers of the encoder circuit at the start and end of encoding process as is commonly done in the case of 1-D convolutional codes.

Example 4.3.1. Consider the 2-D encoder $G(z) \in R^{2\times 6}$ (4.7) from Example 4.1.1. The memory orders of this encoder along the i_1 and i_2 dimensions are $M_1 = 2$ and $M_2 = 1$. In

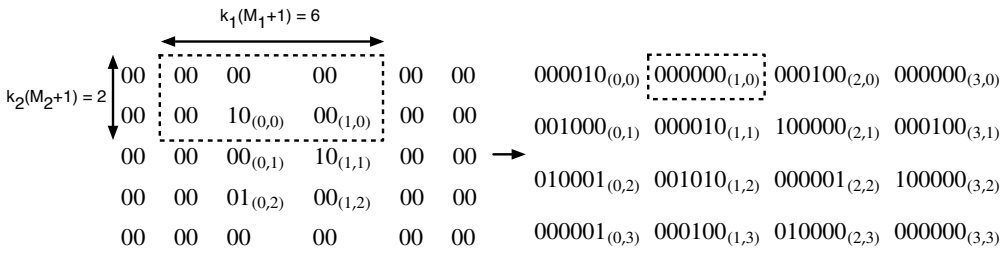


Figure 4.3: Interpretation of memory for a 2-D encoder with input sequence ordering 2×1.

Figure 4.3, the parameters $k = 2$ and $n = 6$ are factored as $k_1 \times k_2 = 2 \times 1$ and $n_1 \times n_2 = 6 \times 1$ to predefine an input sequence ordering of 2×1 in S^2 and output sequence ordering of 6×1 in S^6 as we had done in (4.10). As noted above, the boundaries of the input sequence have to be padded with $k_1 M_1 = 4$ zeros along the i_1 dimension, and $k_2 M_2 = 1$ zeros along the i_2 dimension. These padded zeros are represented in Figure 4.3 without coordinates.

Here, a n-symbol encoded block of size 6×1 at the coordinate (i_1, i_2) of the encoded sequence $v \in S^6$ is a function of $k_1(M_1 + 1) = 2(2 + 1) = 6$ input symbols that start at $l_1 - 2$ and end at l_1 along the i_1 dimension, and $k_2(M_2 + 1) = 1(1 + 1) = 2$ input symbols that start at $l_2 - 1$ and end at l_2 along the i_2 dimension of the input sequence $u \in S^2$. The boxed symbols in Figure 4.3 show a 6×1 symbol encoded block at the coordinate $(1, 0)$ of S^6 as a function of 6 input symbols along i_1 and 2 input symbols along i_2.

Figure 4.4 shows an alternate sequence ordering, where the parameters k and n are factored as $k_1 \times k_2 = 1 \times 2$ and $n_1 \times n_2 = 2 \times 3$ to predefine an input sequence ordering of 1×2 in S^2 and output sequence ordering of 2×3 in S^6. In this case, the boundaries of the input sequence are padded with $k_1 M_1 = 2$ zeros along the i_1 dimension, and $k_2 M_2 = 2$ zeros along the i_2 dimension. These padded zeros are represented in Figure 4.4 without coordinates. The boxed symbols in Figure 4.4 show a 2×3 symbol encoded block at the

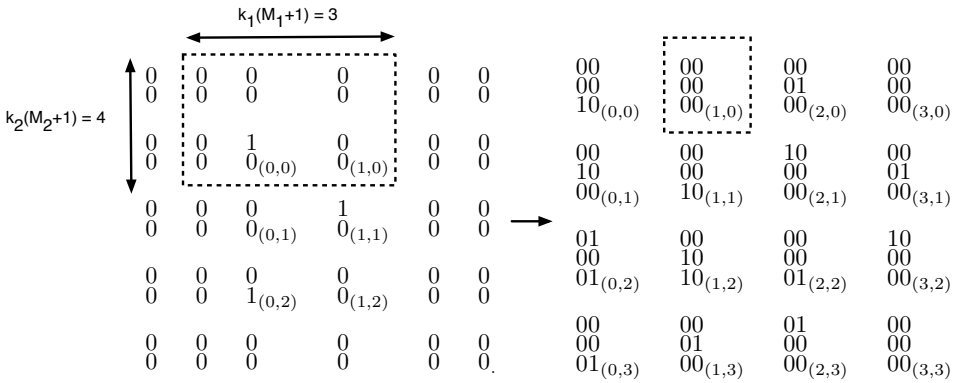


Figure 4.4: Interpretation of memory for a 2-D encoder with input sequence ordering 1×2.

coordinate $(1, 0)$ of S^6 as a function of $1(2+1) = 3$ input symbols along i_1 and $2(1+1) = 4$ input symbols along i_2. ▲

Remark 4.3.1. For $m > 1$, the factoring of the parameters k and n as $\prod_{i=1}^{m} k_i$ and $\prod_{i=1}^{m} n_i$ is not unique. As illustrated in the above example, it is important to note that the same set input symbols affect a given n-symbol encoded block regardless of the choice of sequence ordering.

The need for predefined sequence space ordering can be further justified by considering the notion of *output constraint length* [35, pg. 291]. For a 1-D encoder with memory order M_1, the output constraint length $n(M_1 + 1)$ represents the number of encoded symbols affected by a single input symbol. When extending this concept to m-dimensions it would be desirable to know the output constraint "lengths" along each dimension of the m-D space. That is, the number of encoded symbols in each dimension that are affected by a single input symbol.

The sequence space ordering described in Definition 4.3.1 can be used to define the output constraint lengths as follows: For a rate $\frac{k}{n}$ m-D convolutional encoder with a sequence space ordering of $\prod_{i=1}^{m} k_i$ in S^k and $\prod_{i=1}^{m} n_i$ in S^n, the output constraint length along the ith dimension is given by $n_{A_i} = n_i(M_i + 1)$. A single input symbol in the input sequence affects n_{A_i} output symbols along the ith dimension of the output sequence. The total number of output symbols affected by a single input symbol is $n_A = \prod_{i=1}^{m} n_{A_i}$.

Example 4.3.2. Consider the 2-D encoder $G \in S^{2\times 6}$ (4.7) with memory orders $M_1 = 2$ and $M_2 = 1$. If the parameters $k = 2, n = 6$ are factored as $k_1 \times k_2 = 1 \times 2$ and $n_1 \times n_2 = 2 \times 3$

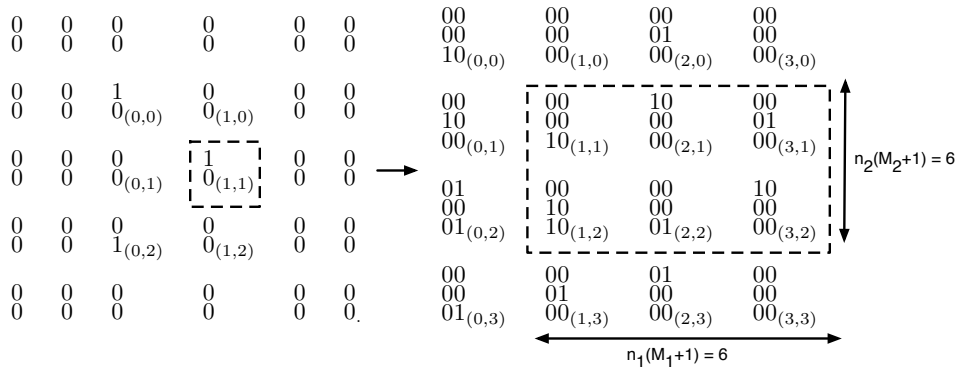


Figure 4.5: Output constraint of a 2-D encoder with output sequence ordering 2×3.

to predefine input and output sequence ordering in S^2 and S^6, then the output constraint length along i_1 is $n_{A_1} = 2(2+1) = 6$ symbols, and the output constraint length along i_2 is $n_{A_2} = 3(1+1) = 6$ symbols, with an overall output constraint of $n_A = 6 \times 6 = 36$ symbols as shown in Figure 4.5.

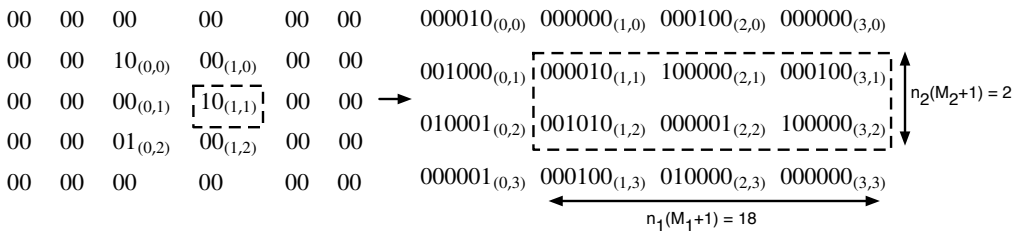


Figure 4.6: Output constraint of a 2-D encoder with output sequence ordering 6×1.

Similarly, if the parameters k, n are factored as $k_1 \times k_2 = 2 \times 1$ and $n_1 \times n_2 = 6 \times 1$ to predefine input and output sequence ordering in S^2 and S^6, then the output constraint length along i_1 is $n_{A_1} = 6(2+1) = 18$ symbols, and the output constraint length along i_2 is $n_{A_2} = 1(1+1) = 2$ symbols, with an overall output constraint of $n_A = 18 \times 2 = 36$ symbols as shown in Figure 4.6. ▲

Chapter 5

Local Invertibility

From Theorem 3.4.1, we know that a m-D convolutional encoder $G(z) \in R^{k\times n}$ has a polynomial inverse $G^{-1}(z) \in R^{n\times k}$ if and only if it is ZP. In this chapter, we define a ZP encoder that has an *invertible subsequence map* between its input and output (codeword) sequences. A encoder that has such an invertible subsequence map is called a *locally invertible encoder*.

5.1 Subsequence Mapping

Let $G(z) \in R^{k\times n}$ be a m-D generator matrix with memory orders $M_1, \ldots, M_m$. Let $G \in S^{k\times n}$ be the equivalent sequence generator of $G(z)$. Factor the parameters k and n to predefine a sequence ordering of $\prod_{i=1}^{m} k_i$ in S^k and $\prod_{i=1}^{m} n_i$ in S^n as described in Definition 4.3.1. As mentioned in Note 4.3.1, we assume that the input data sequence $u \in S^k$ is padded with $k_i M_i$ zeros in each dimension. Now, consider the production of encoded output symbols in each dimension during the convolution operation as shown in Table 5.1. The first $k_i(M_i+1)$ input symbols (*which include $k_i M_i$ padded zeros and k_i valid*

Table 5.1: Subsequence Mapping

Iteration	**Input**	**Output**
1	$k_i(M_i+1)$	n_i
2	$k_i(M_i+2)$	$2n_i$
3	$k_i(M_i+3)$	$3n_i$
	$\cdots$	
j	$k_i(M_i+j)$	jn_i

input symbols) produce n_i output symbols in the ith dimension. In the next iteration k_i additional input symbols produce n_i additional output symbols and so on. If we require the number of input and output symbols to be equal, we set

$$k_i(M_i + j) = jn_i, \tag{5.1}$$

for the jth iteration. Solving for j and substituting back in $k_i(M_i + j)$ or jn_i gives us

$$w_i = \frac{k_i n_i M_i}{n_i - k_i}. \tag{5.2}$$

A positive integer valued w_i is called the *subsequence mapping length*. It is interpreted as follows: Even though output sequences are always larger than their corresponding input sequences, given a reference coordinate of $(l_1, \ldots, l_m)$, a subsequence of w_i symbols starting at l_i of an input sequence in S^k will map to a subsequence of w_i symbols starting at $l_i + M_i$ of its corresponding output sequence in S^n along the ith dimension. The total number of symbols in this equal sized subsequence map between the input and output sequence is $\prod_{i=1}^{m} w_i$. For example, we have row vector subsequences of size w_1 in 1-D, rectangular subsequences of size $w_1 \times w_2$ in 2-D, cuboid subsequences of size $w_1 \times w_2 \times w_3$ in 3-D, orthotope (hypercube) subsequences of size $\prod_{i=1}^{m} w_i$ in m-D and so on. Since w_i is a unit that represents the number of symbols involved in the map, the parameters n_i, k_i and M_i should be such that w_i is a positive integer.

Definition 5.1.1. *A m-D generator matrix $G(z) \in R^{k \times n}$ that has a positive integer valued w_i along each of the m dimensions for any given choice of sequence ordering is said to have a subsequence map of size $\prod_{i=1}^{m} w_i$ between its input and output sequence spaces.*

Let $w = \prod_{i=1}^{m} w_i$. Consider a set of w m-D standard basis input subsequences $b_r \in S^k$, each of size $\prod_{i=1}^{m} w_i$ with order $\prod_{i=1}^{m} k_i$. Let $\hat{g}_r \in S^n$ be the corresponding m-D output subsequences of size $\prod_{i=1}^{m} w_i$ with order $\prod_{i=1}^{m} n_i$ that map to b_r during the convolution $b_r * G$.

$$\begin{array}{ccc} S^k & \longrightarrow & S^n \\ b_r & \mapsto & \hat{g}_r \end{array} \tag{5.3}$$

An inverse subsequence map will exist only if the above map is injective. Recall from Chapter 2 that the sequence spaces S^k and S^n are $\mathbb{F}$-linear, right-shift invariant systems. Linear independence (2.12) of sequences in a sequence space implies that they are $\mathbb{F}$-linearly independent and linearly independent with respect to shifts. Here, we are interested in

finding out if the subsequence map is $\mathbb{F}$-linearly independent. That is, we are only looking for symbol-wise linear independence "within" the subsequence map. Therefore, the inverse subsequence map can be found in the ground field $\mathbb{F}$. This can easily be done by serializing the m-D output subsequence map into 1-D sequences and performing elementary row or column operations on them.

A m-D sequence can be serialized (or unfolded) into a 1-D sequence in many ways. For example it can be unfolded in row-major form, column-major form or by following the path traced by a space-filling curve as it passes through the points of the m-D lattice and so on. The choice of the unfolding technique is not important as long as it is bijective and one is consistent when folding (5.8) the 1-D sequence back into a m-D sequence. For now, without getting into specific details, we denote unfolding with the following bijective operator.

$$\mathcal{U}_f : \mathbb{N}^m \longrightarrow \mathbb{N}^1 \tag{5.4}$$

The inverse subsequence map is found as follows: Each $\hat{g}_r$ (5.3) of size $\prod_{i=1}^{m} w_i$ and order $\prod_{i=1}^{m} n_i$ is unfolded into a 1-D sequence $\mathbf{g}_r$ of length w and order n.

$$\mathcal{U}_f(\hat{g}_r) = \mathbf{g}_r \tag{5.5}$$

Each $\mathbf{g}_r$ is used as a row to construct a $w \times w$ square matrix $\hat{G}$. We will refer to $\hat{G}$ as the *reduced encoding matrix*.

$$\hat{G} = [\mathbf{g}_1, \ldots, \mathbf{g}_w]^T \tag{5.6}$$

The inverse of the reduced encoding matrix (if one exists) can be computed using elementary row or column operations in $\mathbb{F}$ on the matrix $\hat{G}$.

$$\hat{G}^{-1} = [\mathbf{p}_1, \ldots, \mathbf{p}_w]^T \tag{5.7}$$

The rows of the *inverse reduced encoding matrix* $\hat{G}^{-1}$ represent the inverse subsequence map. Each 1-D sequence $\mathbf{p}_r$ of length w and order k is folded back into a m-D subsequence $\hat{p}_r$ of size $\prod_{i=1}^{m} w_i$ and order $\prod_{i=1}^{m} n_i$ using

$$\mathcal{U}_f^{-1}(\mathbf{p}_r) = \hat{p}_r, \tag{5.8}$$

to obtain the inverse subsequence map

$$\begin{array}{ccc} S^n & \longrightarrow & S^k \\ c_r & \mapsto & \hat{p}_r \end{array} . \tag{5.9}$$

Here, the $c_r \in S^n$ are m-D standard basis output subsequences, each of size $\prod_{i=1}^{m} w_i$ with order $\prod_{i=1}^{m} n_i$.

Since the subsequence map defined in (5.3) was constructed using the standard basis as input, it can be used to obtain a output subsequence corresponding to any input subsequence. Let $u_i \in \mathbb{F}$ be the w symbols that make up an input subsequence $\hat{u} \in S^k$ of size $\prod_{i=1}^{m} w_i$. The standard basis input subsequence (5.3) can be used to represent $\hat{u}$ as

$$\hat{u} = u_1 b_1 + u_2 b_2 + \cdots + u_w b_w. \tag{5.10}$$

The corresponding output subsequence $\hat{v} \in S^n$ of size $\prod_{i=1}^{m} w_i$ is given by

$$\hat{v} = u_1 \hat{g}_1 + u_2 \hat{g}_2 + \cdots + u_w \hat{g}_w. \tag{5.11}$$

Let $v_i \in \mathbb{F}$ be the w symbols that make up the output subsequence $\hat{v}$. The standard basis output subsequence from (5.9) can be used to represent this output subsequence $\hat{v}$ as

$$\hat{v} = v_1 c_1 + v_2 c_2 + \cdots + v_w c_w. \tag{5.12}$$

Now, the input subsequence $\hat{u}$ can be recovered from the output subsequence $\hat{v}$ using the inverse subsequence map

$$\hat{u} = v_1 \hat{p}_1 + v_2 \hat{p}_2 + \cdots + v_w \hat{p}_w. \tag{5.13}$$

Definition 5.1.2. *A generator matrix $G(z) \in R^{k \times n}$ is called locally invertible if has an invertible subsequence map of size $\prod_{i=1}^{m} w_i$ between its input and output sequence spaces. Equivalently, $G(z) \in R^{k \times n}$ is locally invertible if it has a nonsingular reduced encoding matrix.*

We started the construction of subsequence mapping at the beginning of this section by letting $G(z)$ be any polynomial matrix, not necessarily of full row rank. Next, we show that if $G(z)$ is locally invertible then it is an m-D convolutional encoder. That is, a locally invertible generator matrix has full row rank.

Proposition 5.1.1. *A locally invertible generator matrix is an m-D convolutional encoder.*

Proof. Let $G(z) \in R^{k \times n}$ be locally invertible. We will show that the sequence generator $G = \psi^{-1}(G(z)) \in S^{k \times n}$ induces an injective map between the input sequence space S^k and output sequence space S^n.

Assume to the contrary that the map $G : S^k \to S^n$ is not injective. Then, there exist input sequences $u_1 \neq u_2 \in S^k$ such that $u_1 * G = u_2 * G = v \in S^n$. Consequently, there exist input subsequences $\hat{u}_1 \neq \hat{u}_2$ of size $\prod_{i=1}^{m} w_i$ starting at some coordinate $(l_1, \ldots, l_m)$ in S^k of u_1 and u_2, that map to the same subsequence $\hat{v}$ of size $\prod_{i=1}^{m} w_i$ starting at the coordinate $(l_1 + M_1, \ldots, l_m + M_m)$ in S^n of the output sequence v. This is a contradiction to the fact that the subsequence map is invertible, and we conclude that the sequence generator map is injective. This implies that $G(z) = \psi(G)$ induces an injective R-module homomorphism between the free R-modules R^k and R^n, and so $\text{rank}(G(z)) = k$. □

Note 5.1.1. From here on, by virtue of the above proposition we will refer to locally invertible generator matrices as locally invertible encoders.

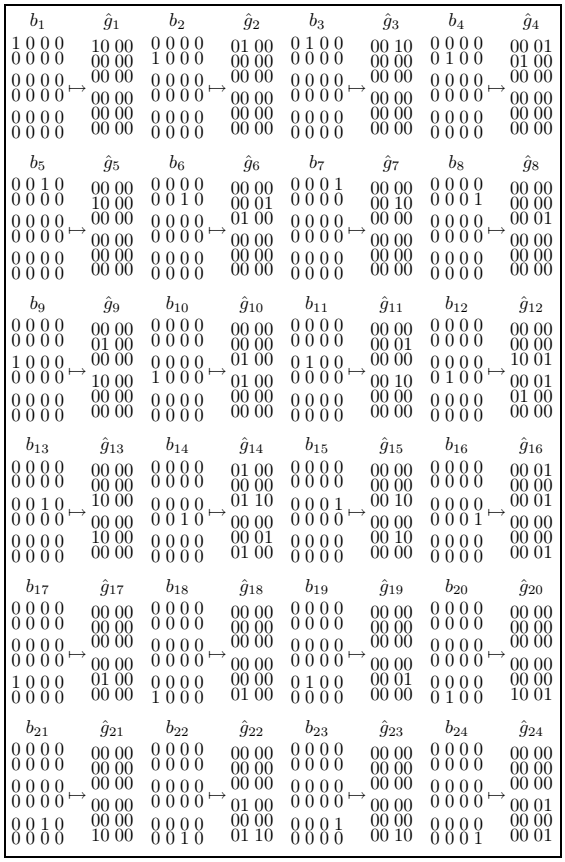
We will end this section by clarifying the above ideas with the following example.

Example 5.1.1. Consider the 2-D encoder $G(z) \in R^{2\times 6}$ (4.7) from Example 4.1.1 with memory orders $M_1 = 2$ and $M_2 = 1$. If $k = 2, n = 6$ are factored as $k_1 \times k_2 = 1 \times 2$ and $n_1 \times n_2 = 2 \times 3$ to predefine a sequence space ordering as shown in Figure 4.4, then the subsequence mapping lengths along the i_1 and i_2 dimensions are $w_1 = \frac{1\times 2\times 2}{2-1} = 4$ and $w_2 = \frac{2\times 3\times 1}{3-2} = 6$. Since w_1 and w_2 are positive integers, $G(z)$ has a subsequence map of size 4×6 between its input and output sequence spaces.

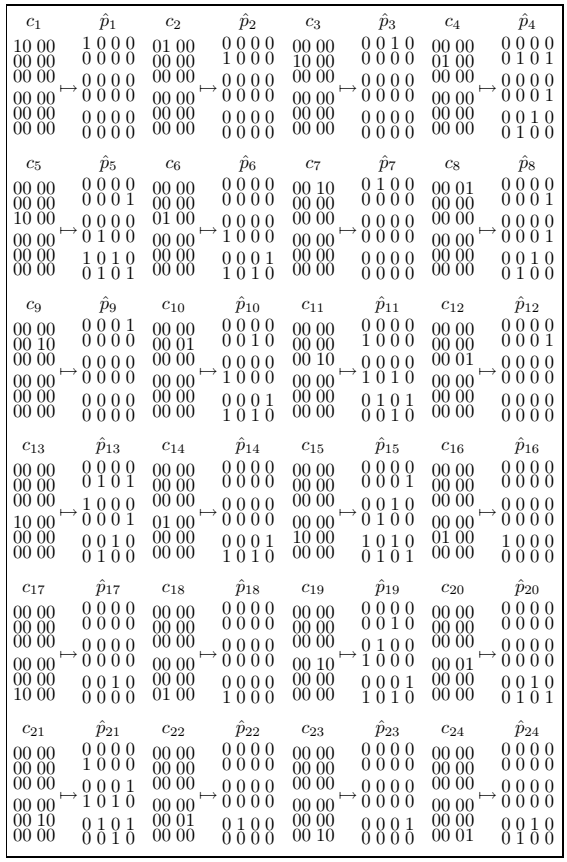
The standard basis subsequence map defined in (5.3) results in a set of $w = 24$ standard basis input subsequences $b_r \in S^2$, each of size 4×6 with order 1×2, mapping to output subsequences $\hat{g}_r \in S^6$ of size 4×6 with order 2×3 as shown in Figure 5.1(a). Each 2-D output subsequence $\hat{g}_r$ in Figure 5.1(a) is unfolded into a 1-D sequence $\mathcal{U}_{fr}(\hat{g}_r) = \mathbf{g}_r$ of length 24 to form a row of the 24×24 reduced encoding matrix $\hat{G} = [\mathbf{g}_1, \ldots, \mathbf{g}_{24}]^T$ shown in Figure 5.1(c). Here, $\mathcal{U}_{fr}$ denotes row-major unfolding and is defined as

$$\begin{array}{rccc} \mathcal{U}_{fr}: & \mathbb{N}^2 & \longrightarrow & \mathbb{N}^1 \\ & (i_1, i_2) & \mapsto & i_1 + i_2(\frac{w_1}{n_1}) \end{array}.$$

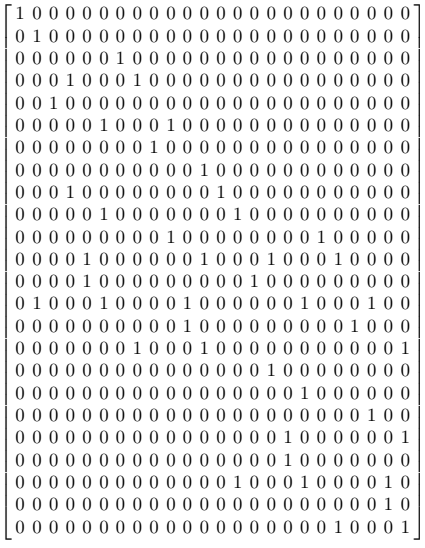
Since the reduced encoding matrix $\hat{G}$ is nonsingular, the encoder $G(z)$ is locally invertible. The rows of the inverse reduced encoding matrix $\hat{G}^{-1} = [\mathbf{p}_1, \ldots, \mathbf{p}_{24}]^T$ in Figure 5.1(d) represent the inverse subsequence map. Each 1-D sequence $\mathbf{p}_r$ of length $w = 24$ is folded $\mathcal{U}_{fr}^{-1}(\mathbf{p}_r) = \hat{p}_r$ into a 2-D subsequence $\hat{p}_r$ of size 4×2 and order 2×3 to obtain the inverse subsequence map shown in Figure 5.1(b).



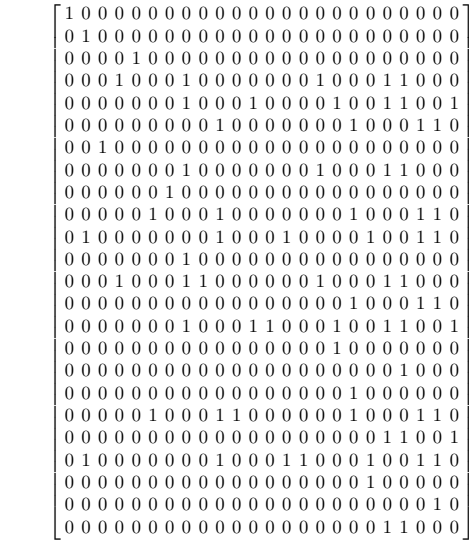
(a) Subsequence map



(b) Inverse subsequence map



(c) Reduced encoding matrix



(d) Inverse reduced encoding matrix

Figure 5.1: Subsequence map and inverse subsequence map of the 2-D locally invertible encoder from Example 4.1.1.

Consider an input subsequence $\hat{u} \in S^2$, represented as the sum of standard basis subsequences b_1, b_9 and b_{24}.

$$\begin{array}{cccccccc}
\hat{u} & & b_1 & & b_9 & & b_{24} \\
1\ 0\ 0\ 0 & & 1\ 0\ 0\ 0 & & 0\ 0\ 0\ 0 & & 0\ 0\ 0\ 0 \\
0\ 0\ 0\ 0 & & 0\ 0\ 0\ 0 & & 0\ 0\ 0\ 0 & & 0\ 0\ 0\ 0 \\
1\ 0\ 0\ 0 & & 0\ 0\ 0\ 0 & & 1\ 0\ 0\ 0 & & 0\ 0\ 0\ 0 \\
0\ 0\ 0\ 0 & = & 0\ 0\ 0\ 0 & + & 0\ 0\ 0\ 0 & + & 0\ 0\ 0\ 0 \\
0\ 0\ 0\ 0 & & 0\ 0\ 0\ 0 & & 0\ 0\ 0\ 0 & & 0\ 0\ 0\ 0 \\
0\ 0\ 0\ 1 & & 0\ 0\ 0\ 0 & & 0\ 0\ 0\ 0 & & 0\ 0\ 0\ 1
\end{array}$$

The subsequence map shown in Figure 5.1(a) can be used to encode $\hat{u}$ into its corresponding output subsequence $\hat{v} \in S^6$ as shown below.

$$\begin{array}{cccccccc}
\hat{v} & & \hat{g}_1 & & \hat{g}_9 & & \hat{g}_{24} \\
10\ \ 00 & & 10\ \ 00 & & 00\ \ 00 & & 00\ \ 00 \\
01\ \ 00 & & 00\ \ 00 & & 01\ \ 00 & & 00\ \ 00 \\
00\ \ 00 & & 00\ \ 00 & & 00\ \ 00 & & 00\ \ 00 \\
10\ \ 01 & = & 00\ \ 00 & + & 10\ \ 00 & + & 00\ \ 01 \\
00\ \ 00 & & 00\ \ 00 & & 00\ \ 00 & & 00\ \ 00 \\
00\ \ 01 & & 00\ \ 00 & & 00\ \ 00 & & 00\ \ 01
\end{array}$$

Similarly, the standard basis output subsequences (5.9) can be used to represent $\hat{v}$ as

$$\begin{array}{cccccccccccc}
\hat{v} & & \hat{c}_1 & & \hat{c}_4 & & \hat{c}_{13} & & \hat{c}_{20} & & \hat{c}_{24} \\
10\ \ 00 & & 10\ \ 00 & & 00\ \ 00 & & 00\ \ 00 & & 00\ \ 00 & & 00\ \ 00 \\
01\ \ 00 & & 00\ \ 00 & & 01\ \ 00 & & 00\ \ 00 & & 00\ \ 00 & & 00\ \ 00 \\
00\ \ 00 & & 00\ \ 00 & & 00\ \ 00 & & 00\ \ 00 & & 00\ \ 00 & & 00\ \ 00 \\
10\ \ 01 & = & 00\ \ 00 & + & 00\ \ 00 & + & 10\ \ 00 & + & 00\ \ 01 & + & 00\ \ 00 \\
00\ \ 00 & & 00\ \ 00 & & 00\ \ 00 & & 00\ \ 00 & & 00\ \ 00 & & 00\ \ 00 \\
00\ \ 01 & & 00\ \ 00 & & 00\ \ 00 & & 00\ \ 00 & & 00\ \ 00 & & 00\ \ 01
\end{array}$$

and the inverse subsequence map in Figure 5.1(b) can be used to recover the input subsequence $\hat{u}$ as shown below.

$$\begin{array}{cccccccccccc}
\hat{u} & & \hat{p}_1 & & \hat{p}_4 & & \hat{p}_{13} & & \hat{p}_{20} & & \hat{p}_{24} \\
1\ 0\ 0\ 0 & & 1\ 0\ 0\ 0 & & 0\ 0\ 0\ 0 & & 0\ 0\ 0\ 0 & & 0\ 0\ 0\ 0 & & 0\ 0\ 0\ 0 \\
0\ 0\ 0\ 0 & & 0\ 0\ 0\ 0 & & 0\ 1\ 0\ 1 & & 0\ 1\ 0\ 1 & & 0\ 0\ 0\ 0 & & 0\ 0\ 0\ 0 \\
1\ 0\ 0\ 0 & & 0\ 0\ 0\ 0 & & 0\ 0\ 0\ 0 & & 1\ 0\ 0\ 0 & & 0\ 0\ 0\ 0 & & 0\ 0\ 0\ 0 \\
0\ 0\ 0\ 0 & = & 0\ 0\ 0\ 0 & + & 0\ 0\ 0\ 1 & + & 0\ 0\ 0\ 1 & + & 0\ 0\ 0\ 0 & + & 0\ 0\ 0\ 0 \\
0\ 0\ 0\ 0 & & 0\ 0\ 0\ 0 & & 0\ 0\ 1\ 0 & & 0\ 0\ 1\ 0 & & 0\ 0\ 1\ 0 & & 0\ 0\ 1\ 0 \\
0\ 0\ 0\ 1 & & 0\ 0\ 0\ 0 & & 0\ 1\ 0\ 0 & & 0\ 1\ 0\ 0 & & 0\ 1\ 0\ 1 & & 0\ 1\ 0\ 0
\end{array}$$

5.2 Sequence Encoding and Inversion

In this section we describe a method to encode and invert multidimensional sequences by using the invertible subsequence map of locally invertible encoders.

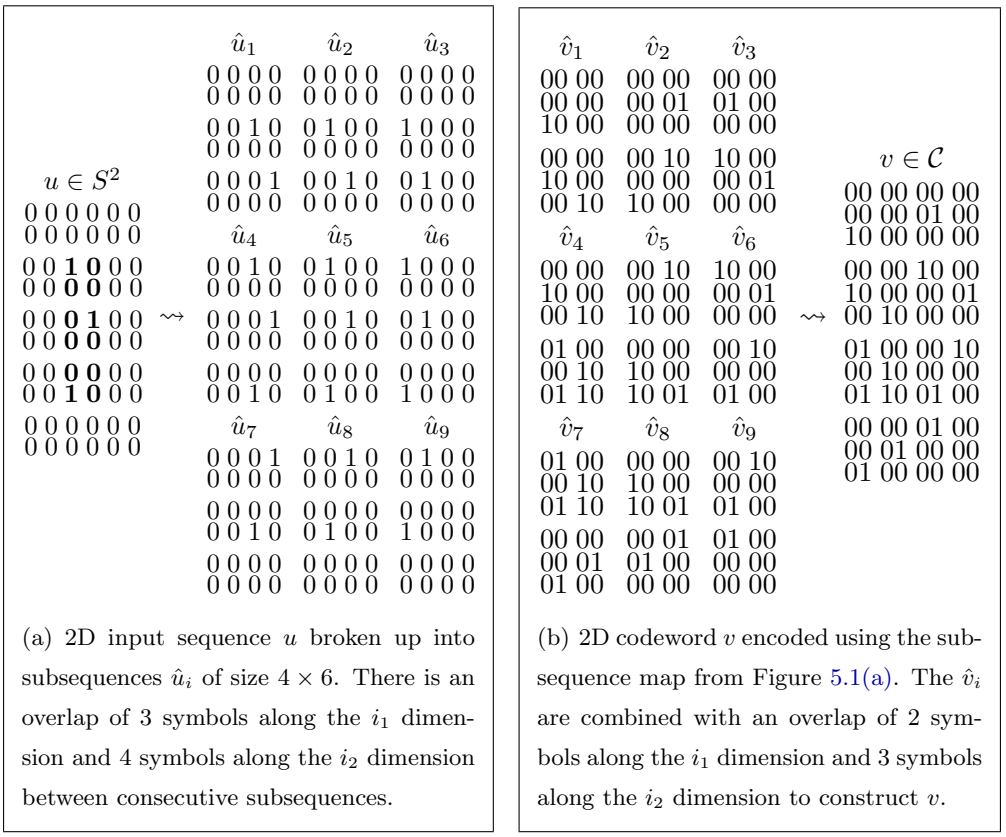
Encoding with Output Subsequence Overlap

Let $G(z) \in R^{k\times n}$ be a locally invertible encoder with an invertible subsequence map of size $\prod_{i=1}^{m} w_i$. Consider an input sequence u and the codeword $v = u * G$. The sequence u can be broken up into subsequences of size $\prod_{i=1}^{m} w_i$ that are shifted by k_i symbols to produce an overlap of $w_i - k_i$ symbols in the ith dimension between consecutive subsequences. Each input subsequence is then encoded into an output subsequence using the subsequence map (5.3) as shown in (5.11). These output subsequences corresponding to consecutive overlapping input subsequences can be put together piecemeal with a shift of n_i symbols and an overlap of $w_i - n_i$ symbols in the ith dimension to construct the entire codeword v.

Encoding without Output Subsequence Overlap

The encoding operation using input subsequences shifted by k_i symbols as mentioned above leads to redundant processing at the encoder. Notice that a shift of k_i in the input sequence space S^k corresponds to a shift of n_i in the output sequence space S^n. Therefore, if the input sequence $u \in S^k$ is broken up into subsequences of size $\prod_{i=1}^{m} w_i$ that are shifted by an amount $\frac{w_i k_i}{n_i}$ with an overlap of $\frac{w_i(n_i-k_i)}{n_i} = k_i M_i$ symbols between consecutive subsequences, then, the corresponding output subsequences would be shifted by an amount w_i to form contiguous subsequences without any overlap. This is a more efficient encoding method without any redundant processing.

Example 5.2.1. Consider the 2-D locally invertible encoder $G(z) \in R^{2\times 6}$ (4.7) in our ongoing example (Examples 4.1.1, 4.3.1, 5.1.1). The input sequence $u \in S^2$ from Figure 4.4 and its corresponding codeword $v = u*G$ are shown in Figure 5.2. Since G has an invertible subsequence map of size 4×6, the input sequence u can be considered to be made up of subsequences $\hat{u}_i$ of size 4×6 that are shifted by $k_1 = 1$ and $k_2 = 2$ symbols along the i_1 and i_2 dimensions as shown in Figure 5.2(a). This leads to an overlap of $w_1 - k_1 = 3$ and $w_2 - k_2 = 4$ symbols along i_1 and i_2 between consecutive input subsequences. Each $\hat{u}_i$ in



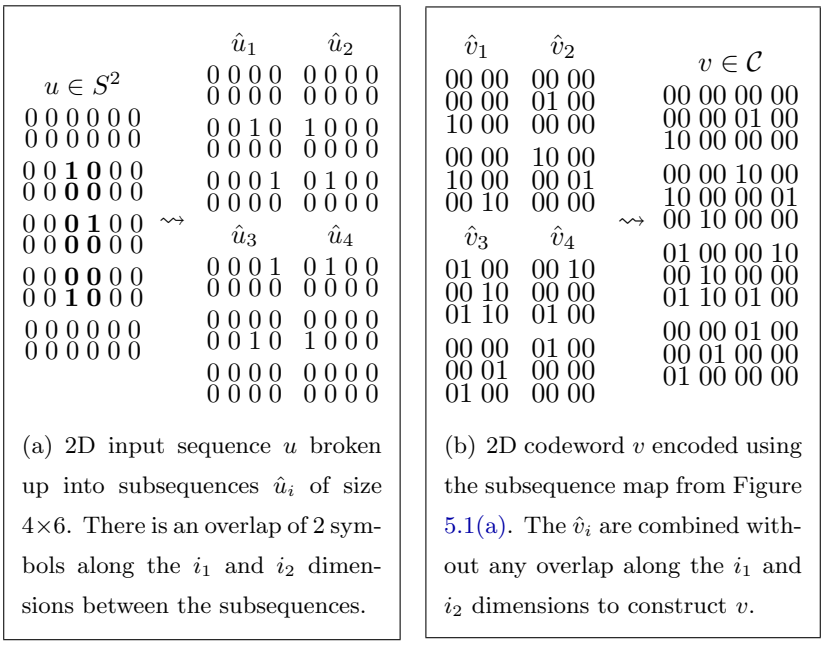
(a) 2D input sequence u broken up into subsequences $\hat{u}_i$ of size 4×6. There is an overlap of 3 symbols along the i_1 dimension and 4 symbols along the i_2 dimension between consecutive subsequences.

(b) 2D codeword v encoded using the subsequence map from Figure 5.1(a). The $\hat{v}_i$ are combined with an overlap of 2 symbols along the i_1 dimension and 3 symbols along the i_2 dimension to construct v.

Figure 5.2: 2D sequence encoding with output subsequence overlap.

Figure 5.2(a) is encoded into an output subsequence $\hat{v}_i$ using the subsequence map shown in Figure 5.1(a) as explained in Example 5.1.1. The corresponding $\hat{v}_i$ can be put together piecemeal with a shift of $n_1 = 2$ and $n_2 = 3$ symbols along the i_1 and i_2 dimensions to construct the entire codeword v as shown in Figure 5.2(b). Such an encoding process leads to an overlap of $w_1 - n_1 = 2$ and $w_2 - n_2 = 3$ symbols along i_1 and i_2 between consecutive output subsequences.

Example 5.2.2. Consider the input sequence u and its corresponding codeword $v = u * G$ shown in Figure 5.2 of the previous example. Figure 5.3(a) shows the input sequence u as being made up of subsequences $\hat{u}_i$ that are shifted by an amount $\frac{w_1 k_1}{n_1} = 2$ and $\frac{w_2 k_2}{n_2} = 4$ with an overlap of $k_1 M_1 = 2$ and $k_2 M_2 = 2$ symbols along the i_1 and i_2 dimensions. The corresponding output subsequences $\hat{v}_i$ obtained from the subsequence map (Figure 5.1(a)) are shifted by an amount $w_1 = 4$ and $w_2 = 6$ symbols along i_1 and i_2 to form contiguous



(a) 2D input sequence u broken up into subsequences $\hat{u}_i$ of size 4×6. There is an overlap of 2 symbols along the i_1 and i_2 dimensions between the subsequences.

(b) 2D codeword v encoded using the subsequence map from Figure 5.1(a). The $\hat{v}_i$ are combined without any overlap along the i_1 and i_2 dimensions to construct v.

Figure 5.3: 2D sequence encoding without output subsequence overlap.

subsequences without any overlap. Each $\hat{v}_i$ is concatenated along the appropriate dimension to construct the codeword v as shown in Figure 5.3(b).

Sequence Inversion with Input Subsequence Overlap

The input sequence can be recovered from the codeword using the inverse subsequence map. This process is similar to the encoding operation described above. Here, the codeword v (assumed to be free of errors) is considered to be made up of overlapping subsequences of size $\prod_{i=1}^{m} w_i$ that are shifted by n_i symbols along the ith dimension. Each output subsequence is then inverted into an input subsequence using the inverse subsequence map (5.9) as shown in (5.13). Consecutive input subsequences are put together piecemeal with a shift of k_i symbols and an overlap of $w_i - k_i$ symbols in the ith dimension to reconstruct the input sequence u. In Section 6.1 we will show that subsequence overlap that occurs between consecutive input subsequences during this sequence inversion technique can be used to determine if the received sequence is the same as the transmitted codeword.

Sequence Inversion without Input Subsequence Overlap

By taking into account that a shift of n_i in S^n along the ith dimension corresponds to a shift of k_i in S^k along the same dimension, an overlap between input subsequences can be avoided. In order to map output subsequences to contiguous input subsequence that are shifted by w_i, we have to consider output subsequences that are shifted by $\frac{w_i n_i}{k_i}$. Since the subsequences under consideration are of size w_i in the ith dimension, two output subsequences that are shifted by $\frac{w_i n_i}{k_i}$ symbols in S^n will have an overlap of $w_i - \frac{w_i n_i}{k_i} = -n_i M_i$ symbols, or rather a separation of $n_i M_i$ symbols between them.

The whole purpose of the encoding operation is to protect information against errors during transmission or storage in a noisy channel. Given an input sequence $u \in S^k$, the encoder generates a codeword $v \in \mathcal{C}$, which is transmitted over the channel. At the output of the channel, the received sequence $r = e + v \in S^n$ is assumed to be corrupted by noise $e \in S^n$. A decoder operates on the received sequence r to produce an estimate $\tilde{u} \in S^k$ of the input sequence u. As explained in Section 3.4, decoding of sequences can be done as a two step process. In the first step, a syndrome former can be used to generate error syndromes from the received sequence r. These error syndromes can be used to index a pre-computed error correction table to perform error correction on the received sequence so as to obtain an estimate $\tilde{v}$ of the codeword v (see for example the 1-D table-driven decoder discussed in [6, 16]). Inversion of the estimated codeword $\tilde{v} \in \mathcal{C}$ can then performed as a separate step using the sequence inversion described above to obtain an estimate $\tilde{u}$ of the input sequence u. This dissertation does not provide a decoding algorithm for error correction. However, a method to obtain the error syndromes required to build the correction tables is discussed Section 6.3.

5.3 The Reduced Encoding Matrix

The $w \times w$ reduced encoding matrix $\hat{G}$ defined in (5.6) has a specific structure. Since a standard basis is used as an input to the subsequence map in (5.3), the output subsequences $\hat{g}_r$ will just contain symbols of the sequences $g_x^{(y)}$ that make up the sequence

generator $G \in S^{k\times n}$.

$$G = \begin{pmatrix} g_1^{(1)}, & \ldots, & g_1^{(n)} \\ & \vdots & \\ g_k^{(1)}, & \ldots, & g_k^{(n)} \end{pmatrix} \tag{5.14}$$

As a result, $\hat{G}$ can be formed by inspection. In this section we will explore the structure of the reduced encoding matrix. This will allow for the explicit construction of locally invertible encoders. The structure of the reduced encoding matrix depends on the ordering of the sequence space as described in Definition 4.3.1 and the unfolding technique (5.4) used to form its rows. To see this we first need to establish the following representation.

Notice that for $k > 1$, when G is viewed as a transfer function matrix from S^k to S^n, the sequences $g_x^{(y)} \in S$ of G are not in composite form but the input sequences in S^k are in composite form with order $\prod_{i=1}^m k_i$. For a fixed y, the sequences $g_x^{(y)} \in S$, $x = 1, \ldots, k$ can be converted to composite form by multiplexing them k_i symbols at a time along the ith dimension to form a single composite sequence $g^{(y)} \in S^k$.

$$G_c = \left\{ \; g^{(1)}, \quad \ldots, \quad g^{(n)} \; \right\} \tag{5.15}$$

The ordering $\prod_{i=1}^m k_i$ of the composite sequences $g^{(y)} \in S^k$ is now consistent with the ordering of the input sequence space S^k. Since the sequences $g_x^{(y)}$ have a maximum length of $M_i + 1$ symbols in the ith dimension, the length of the composite sequence $g^{(y)}$ along the ith dimension is

$$L_i = k_i(M_i + 1). \tag{5.16}$$

Definition 5.3.1. *A sequence $s \in S^l$ with order $\prod_{i=1}^m l_i$ is said to be in reverse composite form if it is reversed l_i symbols at a time along each of its dimensions. We denote this reverse composite form by $\mathcal{R}(s)$.*

Let $\zeta + \mathcal{R}(g^{(y)})$ be the symbol wise addition of an all-zero sequence $\zeta \in S^k$ of size $\prod_{i=1}^m w_i$ and the composite generator sequence $g^{(y)} \in S^k$ of size $\prod_{i=1}^m L_i$ in reverse composite form. We define row-major unfolding of a sequence in S^k of finite size $\prod_{i=1}^m t_i$ with order $\prod_{i=1}^m k_i$, as

$$\begin{array}{rccc} \mathcal{U}_{fr}: & \mathbb{N}^m & \longrightarrow & \mathbb{N}^1 \\ & (i_1, \ldots, i_m) & \mapsto & i_1 + \sum_{j=1}^{m-1} i_{j+1} \prod_{l=1}^{j} \frac{t_l}{k_l} \end{array} . \tag{5.17}$$

If the map in (5.3) is constructed by numbering the standard basis input subsequences $b_r \in S^k$ in row-major form and if the corresponding $\hat{g}_r$ in (5.5) are unfolded into 1-D sequences using row-major unfolding $\mathcal{U}_{fr}(\hat{g}_r) = \mathbf{g}_r$ to form "rows" of the reduced encoding matrix $\hat{G}$ as described in (5.6), then the resulting "columns" of $\hat{G}$ will be shifts of the 1-D sequences

$$\mathcal{U}_{fr}\big(\zeta + \mathcal{R}(g^{(y)})\big). \tag{5.18}$$

The values m, k_i and w_i determine the amount by which the the columns are shifted. The reduced encoding matrix will now have an elegant nested structure as described below.

The $w \times w$ reduced encoding matrix $\hat{G}$ has $1 + \frac{w_m - L_m}{k_m}$ columns made up of the matrix $\hat{G}_{m-1}$, with the jth column shifted down by $(j-1)k_m \prod_{i=1}^{m-1} w_i$ rows.

$$\hat{G} = \begin{bmatrix} \hat{G}_{m-1} & & & & \\ \vdots & \hat{G}_{m-1} & & & \\ & \vdots & & & \\ & & \ddots & & \\ & & & \hat{G}_{m-1} & \\ & & & \vdots & \end{bmatrix} \tag{5.19}$$

Similarly for values of x ranging from $(m-1)$ to 2, the matrix $\hat{G}_x$ consists of $1 + \frac{w_x - L_x}{k_x}$ columns made up of the matrix $\hat{G}_{x-1}$, with the jth column shifted down by $(j-1)k_x \prod_{i=1}^{x-1} w_i$ rows.

$$\hat{G}_x = \begin{bmatrix} \hat{G}_{x-1} & & & & \\ \vdots & \hat{G}_{x-1} & & & \\ & \vdots & & & \\ & & \ddots & & \\ & & & \hat{G}_{x-1} & \\ & & & \vdots & \end{bmatrix} \tag{5.20}$$

The matrix G_1 has $1 + \frac{w_1 - L_1}{k_1}$ columns made up of the matrix $\hat{G}_0$, with the jth column shifted down by $(j-1)k$ rows.

$$\hat{G}_1 = \begin{bmatrix} \hat{G}_0 & & & \\ \vdots & \hat{G}_0 & & \\ & \vdots & & \\ & & \ddots & \\ & & & \hat{G}_0 \\ & & & \vdots \end{bmatrix} \tag{5.21}$$

The fundamental matrix $\hat{G}_0$ has n columns consisting of the 1-D sequences defined in (5.18). That is,

$$\hat{G}_0 = \Big[\mathcal{U}_{fr}\big(\zeta + \mathcal{R}(g^{(1)})\big),\ \ldots,\ \mathcal{U}_{fr}\big(\zeta + \mathcal{R}(g^{(n)})\big)\Big] \tag{5.22}$$

where each $\mathcal{U}_{fr}\big(\zeta+\mathcal{R}(g^{(y)})\big)$ is treated as a column vector. To summarize, the $w\times w$ reduced encoding matrix $\hat{G}$ is made up of m nested column-matrices. The columns of these nested matrices are in turn just shifts of the 1-D sequences $\mathcal{U}_{fr}\big(\zeta + \mathcal{R}(g^{(y)})\big)$.

Example 5.3.1. Consider the 2-D locally invertible encoder from Example 4.1.1. The reduced encoding matrix $\hat{G}$ (shown in Figure 5.1(c)) corresponding to a sequence ordering of 1×2 in S^2 and 2×3 in S^6 can be constructed directly from the sequence encoder G without using the subsequence map.

$$G = \begin{pmatrix} 0\,0\,0 & 0\,0\,0 & 0\,0\,0 & 0\,0\,1 & 1\,0\,0 & 0\,0\,0 \\ 0\,0\,1, & 0\,0\,0, & 1\,0\,0, & 0\,0\,0, & 0\,0\,0, & 0\,0\,0 \\ \\ 0\,0\,0 & 1\,0\,0 & 0\,0\,0 & 0\,0\,0 & 0\,1\,0 & 1\,0\,1 \\ 0\,0\,0, & 0\,0\,1, & 0\,0\,0, & 0\,1\,0, & 0\,0\,0, & 1\,0\,0 \end{pmatrix} \tag{5.23}$$

Since the input sequence space is ordered as 1×2, the composite generator sequences $g^{(y)} \in S^2$ of G_c are formed by interleaving $k_1 = 1$ and $k_2 = 2$ symbols of $g_x^{(y)} \in S$ of G along the i_1 and i_2 dimensions respectively.

$$G_c = \left\{ \begin{matrix} 0\,0\,0 & 0\,0\,0 & 0\,0\,0 & 0\,0\,1 & 1\,0\,0 & 0\,0\,0 \\ 0\,0\,0 & 1\,0\,0 & 0\,0\,0 & 0\,0\,0 & 0\,1\,0 & 1\,0\,1 \\ \\ 0\,0\,1 & 0\,0\,0 & 1\,0\,0 & 0\,0\,0 & 0\,0\,0 & 0\,0\,0 \\ 0\,0\,0, & 0\,0\,1, & 0\,0\,0, & 0\,1\,0, & 0\,0\,0, & 1\,0\,0 \end{matrix} \right\}$$

The lengths of the composite sequences $g^{(y)}$ along i_1 and i_2 are $L_1 = k_1(M_1 + 1) = 3$ and $L_2 = k_2(M_2 + 1) = 4$. Each $g^{(y)}$ is reversed $k_1 = 1$ and $k_2 = 2$ symbols at a time along i_1 and i_2 to obtain the reverse composite form $\mathcal{R}(g^{(y)})$.

$$\mathcal{R}(G_c) = \left\{ \begin{matrix} 1\,0\,0 & 0\,0\,0 & 0\,0\,1 & 0\,0\,0 & 0\,0\,0 & 0\,0\,0 \\ 0\,0\,0 & 1\,0\,0 & 0\,0\,0 & 0\,1\,0 & 0\,0\,0 & 0\,0\,1 \\ \\ 0\,0\,0 & 0\,0\,0 & 0\,0\,0 & 1\,0\,0 & 0\,0\,1 & 0\,0\,0 \\ 0\,0\,0, & 0\,0\,1, & 0\,0\,0, & 0\,0\,0, & 0\,1\,0, & 1\,0\,1 \end{matrix} \right\}$$

The reduced encoding matrix from Figure 5.1(c) is shown once again in Figure 5.4. The blanks in the matrix denote zeros and serve to highlight the nested structure described in equations (5.19)-(5.22). Notice that the first $n = 6$ columns of $\hat{G}$ in Figure 5.4 are 1-D sequences of the form $\mathcal{U}_{fr}(\zeta + \mathcal{R}(g^{(y)}))$, where $\zeta \in S^2$ is an all-zero sequence of size $w_1 \times w_2 = 4 \times 6$. The remaining columns of $\hat{G}$ are shifted versions of the first 6 columns. ▲

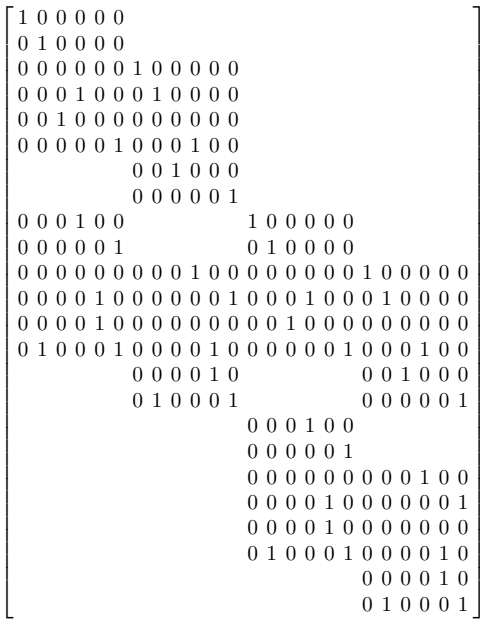


Figure 5.4: Reduced encoding matrix of the 2-D encoder from Example 4.1.1 with input sequence ordering 1×2.

For values of $m > 1$, we know that the sequence space ordering proposed in Definition 4.3.1 is not unique. In the 2-D example discussed above, it is natural to question the choice of sequence ordering as 1×2 in S^2 and 2×3 in S^6. The following example serves to illustrate that the subsequence map generated by a different choice of sequence ordering may not be invertible.

Example 5.3.2. Consider the 2-D encoder from the above example. If we factor k, n as $k_1 \times k_2 = 2 \times 1$ and $n_1 \times n_2 = 3 \times 2$ to predefine an ordering of 2×1 in S^2 and 3×2 in S^6, then the subsequence mapping lengths along i_1 and i_2 are $w_1 = \frac{2 \times 3 \times 2}{3-2} = 12$ and $w_2 = \frac{1 \times 2 \times 1}{2-1} = 2$. In this case we have input subsequences of size 12×2 in S^2 mapping to output subsequences of size 12×2 in S^6. As in the above example, the reduced encoding matrix can be constructed from the sequence encoder G (5.23). Since the input sequence space is ordered as 2×1, the composite generator sequences $g^{(y)} \in S^2$ of G_c are formed by interleaving $k_1 = 2$ and $k_2 = 1$ symbols of $g_x^{(y)} \in S$ of G along the i_1 and i_2 dimensions respectively.

$$G_c = \left\{ \begin{array}{cccccc} 00\ 00\ 00 & 01\ 00\ 00 & 00\ 00\ 00 & 00\ 00\ 10 & 10\ 01\ 00 & 01\ 00\ 01 \\ 00\ 00\ 10, & 00\ 00\ 01, & 10\ 00\ 00, & 00\ 01\ 00, & 00\ 00\ 00, & 01\ 00\ 00 \end{array} \right\}$$

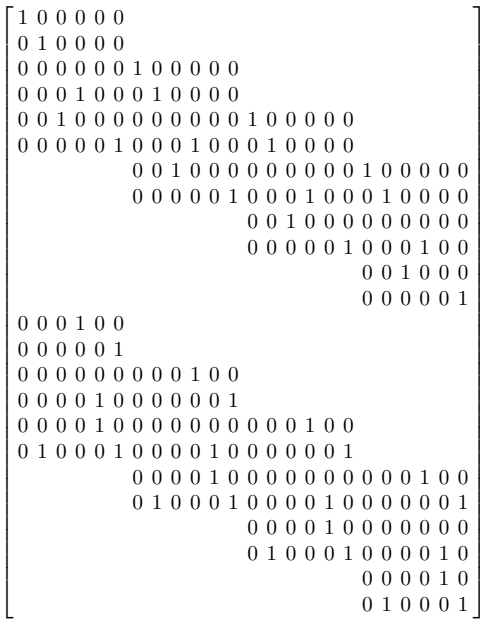


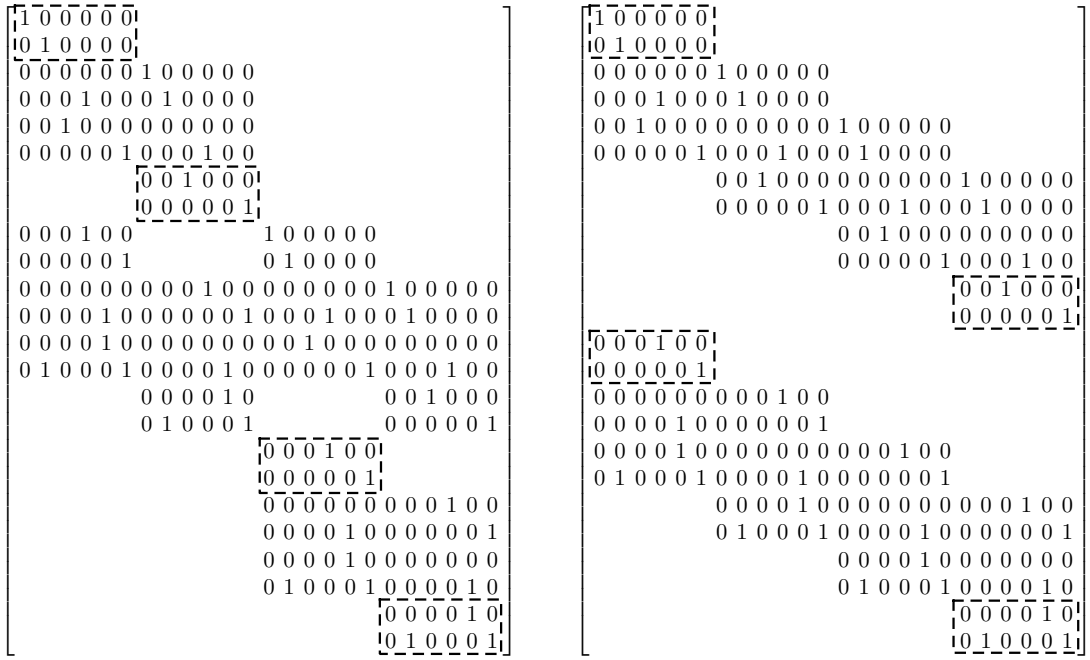
Figure 5.5: Reduced encoding matrix of the 2-D encoder from Example 4.1.1 with input sequence ordering 2×1.

The lengths of the composite sequences $g^{(y)}$ along i_1 and i_2 are $L_1 = k_1(M_1 + 1) = 6$ and $L_2 = k_2(M_2 + 1) = 2$. Each $g^{(y)}$ is reversed $k_1 = 2$ and $k_2 = 1$ symbols at a time along i_1 and i_2 to obtain the reverse composite form $\mathcal{R}(g^{(y)})$.

$$\mathcal{R}(G_c) = \left\{ \begin{array}{cccccc} 10\ 00\ 00 & 01\ 00\ 00 & 00\ 00\ 10 & 00\ 01\ 00 & 00\ 00\ 00 & 00\ 00\ 01 \\ 00\ 00\ 00, & 00\ 00\ 01, & 00\ 00\ 00, & 10\ 00\ 00, & 00\ 01\ 10, & 01\ 00\ 01 \end{array} \right\}$$

The 2-D sequences $\zeta + \mathcal{R}(g^{(y)})$ are then unfolded into 1-D sequences $\mathcal{U}_{fr}(\zeta + \mathcal{R}(g^{(y)}))$ to form the 6 columns of the fundamental matrix $\hat{G}_0$. Here, $\zeta \in S^6$ is an all-zero sequence of size 12×2. The 24×24 reduced encoding matrix $\hat{G}$ shown in Figure 5.5 is then constructed by shifting $\hat{G}_0$ by appropriate amounts as explained in (5.19)-(5.22). In Figure 5.5 notice that column 3 is identical to column 13, and column 9 is identical to column 19. Therefore $\hat{G}$ is singular, and consequently the subsequence map (5.3) of size 12×2 does not have an inverse. Note however that the same encoder (as shown in the pervious example) has an invertible subsequence map of size 4×6 when the sequence space ordering is 1×2 in S^2 and 2×3 in S^6, and is therefore still considered to be locally invertible. ▲

Consider again the reduced encoding matrices from the above examples (Figure 5.4 and Figure 5.5) as shown in Figure 5.6. We will make some key observations about their



(a) $\hat{G}_{24\times 24}$: Reduced encoding matrix of the 2-D encoder from Example 4.1.1 with input sequence ordering 1×2 and output sequence ordering 2×3.

(b) $\hat{G}_{24\times 24}$: Reduced encoding matrix of the 2-D encoder from Example 4.1.1 with input sequence ordering 2×1 and output sequence ordering 3×2.

Figure 5.6: Indicator matrices of the 2-D encoder from Example 4.1.1.

structures which will provide us with necessary conditions for them to be invertible. As shown in the above examples, the columns of $\hat{G}$ are shifted versions of the 1-D sequences $\mathcal{U}_{fr}\big(\zeta + \mathcal{R}(g^{(y)})\big)$. In Figure 5.6(a) the composite sequences $g^{(y)}$ have order 1×2, and in Figure 5.6(b) the composite sequences $g^{(y)}$ have order 2×1. In both cases the composite sequences are constructed from the same encoder

$$G(z) = \begin{bmatrix} z_1^2 z_2 & 0 & z_2 & z_1^2 & 1 & 0 \\ 0 & 1 + z_1^2 z_2 & 0 & z_1 z_2 & z_1 & 1 + z_1^2 + z_2 \end{bmatrix},$$

which has constraint lengths $\nu_{1,1} = 2, \nu_{1,2} = 2$, $\nu_{1,2} = 1, \nu_{1,2} = 1$ and memory orders $M_1 = 2, M_2 = 1$.

The boxed terms in Figure 5.6 (in top to bottom order) correspond to the coefficients of the $z_1^2 z_2^1$, z_2^1, z_1^2, and $z_1^0 z_2^0$ terms of the polynomial entries $g(z)_x^{(y)}$ in $G(z)$. The top

to bottom ordering arises due to the reversal of the sequences, and each box has $k = 2$ rows because the sequences are in composite form. We shall refer to each box as the *indicator matrix* for the coefficient of the corresponding monomial and denote them by

$$G_{z_1^2 z_2^1} = \begin{bmatrix} 1\,0\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0\,0 \end{bmatrix} \quad G_{z_2^1} = \begin{bmatrix} 0\,0\,1\,0\,0\,0 \\ 0\,0\,0\,0\,0\,1 \end{bmatrix}$$

$$G_{z_1^2} = \begin{bmatrix} 0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,0\,1 \end{bmatrix} \quad G_{z_1^0 z_2^0} = \begin{bmatrix} 0\,0\,0\,0\,1\,0 \\ 0\,1\,0\,0\,0\,1 \end{bmatrix}.$$

Taking into account that the blanks in Figure 5.6 denote zeros, observe that the nonzero elements of the indicator matrices are the only nonzero terms in the corresponding rows of $\hat{G}$. This phenomenon is independent of the sequence ordering and the unfolding technique. In this particular example, it arises due to the fact that each set of $n = 6$ columns of $\hat{G}$ is identical to the previous set of 6 columns but shifted down by a fixed number of rows as explained in (5.19)-(5.22). Now, if $G(z) \in R^{2\times 6}$ had a row without a $z_1^2 z_2^1$ term in any of its components (i.e. if $G_{z_1^2 z_2^1}$ had a zero row), then one of the first $k = 2$ rows of $\hat{G}$ would also be a zero row, and consequently $\hat{G}$ would be singular. Similarly, if $G_{z_1^0 z_2^0}$ had a zero row, then one of the last 2 rows of $\hat{G}$ would have been a zero row. The same conditions apply to the monomials z_2^1 and z_1^2 as well, and correspond to the indicator matrices (boxes in the middle portion) $G_{z_2^1}$ and $G_{z_1^2}$ of $\hat{G}$. Finally, needless to say, observe that $\hat{G}$ will be singular if the rows of the indicator matrices are linearly dependent.

Next, we generalize these observations to the m-D case. Let $G(z) \in R^{k\times n}$ be an m-D generator matrix ($m \geq 1$) as shown below

$$G(z) = \begin{bmatrix} g(z)_1^{(1)}, & \ldots, & g(z)_1^{(n)} \\ & \vdots & \\ g(z)_k^{(1)}, & \ldots, & g(z)_k^{(n)} \end{bmatrix}.$$

As defined in Section 4.2, we denote the constraint length of the xth row of $G(z)$ in the ith dimension by $\nu_{i,x}$, and the memory order of $G(z)$ in the ith dimension by M_i. Here, we define the *total degree* τ_x *of the* x*th row of* $G(z)$ as the maximum total degree of its components. That is,

$$\tau_x = \max_{1\leq y\leq n} \{\mathrm{tdeg}\, g(z)_x^{(y)}\}. \tag{5.24}$$

It is important to note that $\nu_{i,x}$ and τ_x are not necessarily obtained from the exponents of the same terms of the components of the xth row. In general

$$\sum_{i=1}^{m} \nu_{i,x} \geq \tau_x, \tag{5.25}$$

with equality if and only if the xth row of $G(z)$ has a component $g(z)_x^{(y)}$ with a nonzero $z_1^{\nu_{1,x}} z_2^{\nu_{2,x}} \cdots z_m^{\nu_{m,x}}$ term.

Condition 5.3.1. The following are necessary conditions for an m-D generator matrix $G(z) \in R^{k \times n}$ to be locally invertible.

i) The constraint lengths of every row of $G(z)$ along the ith dimension must be equal. That is,

$$M_i = \nu_{i,x}; \quad x = 1, \ldots, k.$$

ii) Every row of $G(z)$ must have at least one component with a nonzero $z_1^0 z_2^0 \cdots z_m^0$ term (i.e. a nonzero constant term).

iii) Every row of $G(z)$ must have at least one component with a nonzero $z_1^{M_1} z_2^{M_2} \cdots z_m^{M_m}$ term. In this case (due to (i) above), we have equality in (5.25). That is,

$$\tau_x = \sum_{i=1}^{m} M_i = \sum_{i=1}^{m} \nu_{i,x}; \quad x = 1, \ldots, k.$$

iv) If we consider the monomials $z_1^0 z_2^0 \cdots z_m^0$ and $z_1^{M_1} z_2^{M_2} \cdots z_m^{M_m}$ as the diagonally opposite vertices of an m-D hypercube, then the monomials

$$z_1^{M_1}, z_2^{M_2}, \ldots, z_m^{M_m}, \quad z_1^{M_1} z_2^{M_2}, z_1^{M_1} z_3^{M_3}, \ldots, z_1^{M_1} z_m^{M_m}, \quad z_2^{M_2} z_3^{M_3}, z_2^{M_2} z_4^{M_4}, \ldots z_2^{M_2} z_m^{M_m},$$
$$\ldots, z_1^{M_1} z_2^{M_2} z_3^{M_3}, z_1^{M_1} z_2^{M_2} z_4^{M_4}, \ldots, z_1^{M_1} z_2^{M_2} z_m^{M_m}, \quad \ldots\ldots\ldots\ldots$$

corresponding to remaining 2^{m-1} vertices of the hypercube must appear as nonzero terms in at least one component of every row of $G(z)$.

v) If we denote the 2^m monomial terms defined above in condition (iv) as $\alpha_1, \alpha_2, \ldots, \alpha_{2^m}$, and define a $k \times n$ "indicator matrix for the coefficients of α_jth term in the components of $G(z)$" by

$$G_{\alpha_j} = \begin{bmatrix} \bar{g}_1^{(1)}, & \ldots, & \bar{g}_1^{(n)} \\ & \vdots & \\ \bar{g}_k^{(1)}, & \ldots, & \bar{g}_k^{(n)} \end{bmatrix}, \tag{5.26}$$

where

$$\bar{g}_x^{(y)} = \underset{\alpha_j}{\text{coeff}}\ g(z)_x^{(y)},$$

then each of the 2^m indicator matrices $G_{\alpha_1}, \ldots, G_{\alpha_{2^m}}$ must have rank k.

Next, we show that when $m = 1$, the reduced encoding matrix is a submatrix of the semi-infinite generator matrix $\mathbf{G}$ defined below. Let $G(z_1) \in R^{k\times n}$ be a 1-D generator matrix with a subsequence map of size $w_1 = \frac{nkM_1}{n-k}$ as defined in Definition 5.1.1. We denote the generator sequences $g_x^{(y)} \in S$ of G as

$$g_x^{(y)} = \left(g_{x,(0)}^{(y)}, g_{x,(1)}^{(y)}, \ldots, g_{x,(M_1)}^{(y)}\right). \tag{5.27}$$

For a 1-D input sequence $u \in S^k$ of arbitrary length, the encoding equation $v = u * G$ can be written in matrix form as $v = u\mathbf{G}$, where

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_0 & \mathbf{G}_1 & \mathbf{G}_2 & \ldots & \mathbf{G}_{M_1} & & \\ & \mathbf{G}_0 & \mathbf{G}_1 & \ldots & \mathbf{G}_{M_1-1} & \mathbf{G}_{M_1} & \\ & & \mathbf{G}_0 & \ldots & \mathbf{G}_{M_1-2} & \mathbf{G}_{M_1-1} & \mathbf{G}_{M_1} \\ & & \ddots & & & & \ddots \end{bmatrix}. \tag{5.28}$$

The blanks in $\mathbf{G}$ denote zeros and each $\mathbf{G}_j$ is a $k \times n$ submatrix of the form

$$\mathbf{G}_j = \begin{bmatrix} g_{1(j)}^{(1)} & g_{1(j)}^{(2)} & \cdots & g_{1(j)}^{(n)} \\ g_{2(j)}^{(1)} & g_{2(j)}^{(2)} & \cdots & g_{2(j)}^{(n)} \\ \vdots & & & \\ g_{k(j)}^{(1)} & g_{k(j)}^{(2)} & \cdots & g_{k(j)}^{(n)} \end{bmatrix}. \tag{5.29}$$

The matrix $\mathbf{G}$ is well known (see for example [35, pg. 292]) and is referred to as the semi-infinite generator matrix corresponding to the fact that the input sequence u can be of arbitrary length. If $u \in S^k$ has finite length of kt symbols, then $\mathbf{G}$ has kt rows and $n(M_1 + t)$ columns.

Consider the composite sequence representation G_c of the 1-D sequence generator G. For a fixed y, the sequences $g_x^{(y)} \in S$, $x = 1, \ldots, k$ are multiplexed k symbols at a time to form the composite sequences $g^{(y)} \in S^k$ of length $L_1 = k(M_1 + 1)$.

$$g^{(y)} = \left(g_{1,(0)}^{(y)} g_{2,(0)}^{(y)} \cdots g_{k,(0)}^{(y)},\ \ g_{1,(1)}^{(y)} g_{2,(1)}^{(y)} \cdots g_{k,(1)}^{(y)},\ \ \ldots,\ \ g_{1,(M_1)}^{(y)} g_{2,(M_1)}^{(y)} \cdots g_{k,(M_1)}^{(y)}\right)_{1\times L_1} \tag{5.30}$$

The reverse composite form is given as

$$\mathcal{R}(g^{(y)}) = \Big(g^{(y)}_{1,(M_1)} g^{(y)}_{2,(M_1)} \cdots g^{(y)}_{k,(M_1)},\ \ g^{(y)}_{1,(M_1-1)} g^{(y)}_{2,(M_1-1)} \cdots g^{(y)}_{k,(M_1-1)},\ \ \ldots, \\ \ldots,\ \ g^{(y)}_{1,(0)} g^{(y)}_{2,(0)} \cdots g^{(y)}_{k,(0)}\Big)_{1\times L_1}. \quad (5.31)$$

The addition of an all-zero sequence $\zeta \in S^k$ of length $w = w_1$ with the reverse composite sequence $\mathcal{R}(g^{(y)})$ of length L_1 is given by

$$\zeta + \mathcal{R}(g^{(y)}) = \Big(\underbrace{g^{(y)}_{1,(M_1)} g^{(y)}_{2,(M_1)} \cdots g^{(y)}_{k,(M_1)},\ \ \ldots,\ \ g^{(y)}_{1,(0)} g^{(y)}_{2,(0)} \cdots g^{(y)}_{k,(0)}}_{L_1 = k(M_1+1)},\ \ \underbrace{00\ldots0,\ \ \ldots 0}_{w-L_1}\Big)_{1\times w} \quad (5.32)$$

For $m = 1$, since the subsequences are already in 1-D form, no unfolding is necessary in the construction of the reduced encoding matrix. So here, $\mathcal{U}_{fr}\big(\zeta + \mathcal{R}(g^{(y)})\big) = \big(\zeta + \mathcal{R}(g^{(y)})\big)$. The fundamental matrix (5.22) is given as

$$\hat{G}_0 = \Big[\big(\zeta + \mathcal{R}(g^{(1)})\big),\ \ldots,\ \big(\zeta + \mathcal{R}(g^{(n)})\big)\Big], \quad (5.33)$$

where each $\big(\zeta + \mathcal{R}(g^{(y)})\big)$ is treated as a column vector. The resulting $w \times w$ reduced encoding matrix (5.19) is given by

$$\hat{G} = \begin{bmatrix} \hat{G}_0 & & & \\ \vdots & \hat{G}_0 & & \\ & \vdots & & \\ & & \ddots & \\ & & & \hat{G}_0 \\ & & & \vdots \end{bmatrix} = \begin{bmatrix} \mathbf{G}_{M_1} & & & \\ \mathbf{G}_{M_1-1} & \mathbf{G}_{M_1} & & \\ \vdots & \mathbf{G}_{M_1-1} & & \\ \vdots & \vdots & \ddots & \\ \vdots & \vdots & & \mathbf{G}_{M_1} \\ \mathbf{G}_0 & \vdots & & \mathbf{G}_{M_1-1} \\ & \mathbf{G}_0 & & \vdots \\ & & & \vdots \\ & & & \mathbf{G}_0 \end{bmatrix} \quad (5.34)$$

where blanks denote zeros and each $\mathbf{G}_j$ is a $k \times n$ submatrix of the form defined in (5.29). As mentioned in Section 5.1, any input subsequence of length $1 \times w$ starting at a coordinate (l_1) in S^k will map to an output subsequence of length $1 \times w$ starting at the coordinate $(l_1 + M_1)$ in S^n. Setting the value of $l_1 = 0$ gives us the subsequence mapping coordinates (0) in S^k and (M_1) in S^n. This in turn corresponds to the fact that $\hat{G}$ (5.34) is a $w \times w$ submatrix of the semi-infinite generator matrix $\mathbf{G}$ (5.28) starting at column position M_1.

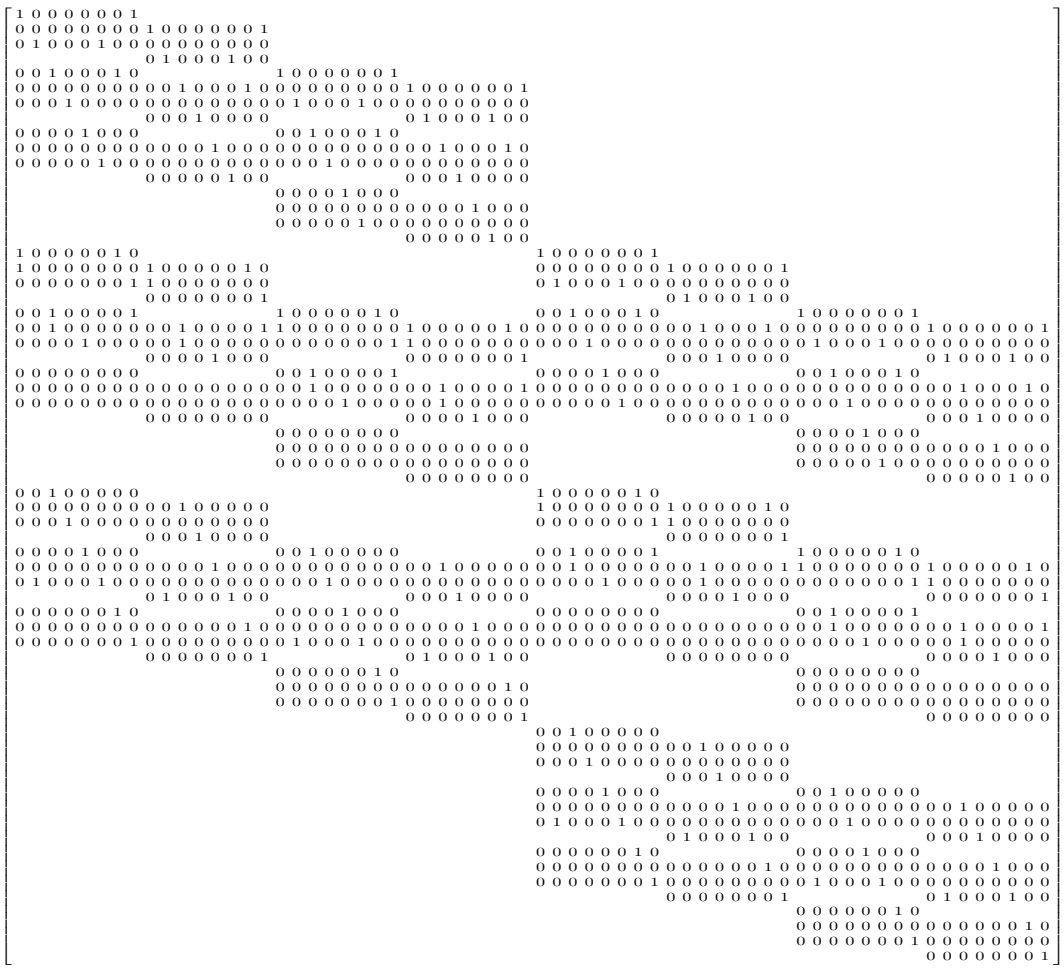


Figure 5.7: $\hat{G}_{64\times 64}$: Reduced encoding matrix of a 3-D locally invertible encoder

The following example demonstrates the construction of a 3-D locally invertible encoder by exploiting the structure of the reduced encoding matrix.

Example 5.3.3. Let $\mathbb{F} = \mathbb{F}_2, m = 3, k = 1, n = 8$ and $M_1 = M_2 = M_3 = 2$. We factor k, n as $\prod_{i=1}^{3} k_i = 1 \times 1 \times 1$ and $\prod_{i=1}^{3} n_i = 2 \times 2 \times 2$ to predefine a sequence space ordering of $1 \times 1 \times 1 \in S^1$ and $2 \times 2 \times 2 \in S^8$. The subsequence mapping lengths along i_1, i_2 and i_3 are $w_1 = w_2 = w_3 = 4$, and $w = \prod_{i=1}^{3} w_i = 64$. Consider the reduced encoding matrix $\hat{G}_{64\times 64}$ shown in Figure 5.7 constructed using $n = 8$, randomly generated 1-D sequences of the form $a^{(y)} = \mathcal{U}_{fr}(\zeta + \mathcal{R}(g^{(y)}))$. Here ζ is a all-zero 3-D subsequence of size $\prod_{i=1}^{3} w_i = 4 \times 4 \times 4$ in S^1, and $g^{(y)}$ is a element of G_c of size $\prod_{i=1}^{3} L_i = 3 \times 3 \times 3$, where $L_i = k_i(M_i + 1) = 3$. If $\hat{G}$ is nonsingular (it is in this case), the polynomial matrix $G(z) = [g(z)_1^{(1)}, \ldots, g(z)_1^{(8)}] \in R^{1\times 8}$

obtained from the 1-D sequences $a^{(y)}$ will locally invertible. For example, $g(z)_1^{(1)}$ is obtained from $a^{(1)}$ as follows.

$$a^{(1)} = [100(0)000(0)000(00000)110(0)000(0)000(00000)000(0)000(0)000]^T$$

The zeros in braces represent blanks in the reduced encoding matrix shown in Figure 5.7.

$$\begin{array}{lll} \zeta + \mathcal{R}(g^{(1)}) = \mathcal{U}_{fr}^{-1}(a^{(1)}) & \mathcal{R}(g^{(1)}) & g^{(1)} \\ \begin{matrix} 1\,0\,0\,0 & 1\,1\,0\,0 & 0\,0\,0\,0 \\ 0\,0\,0\,0 & 0\,0\,0\,0 & 0\,0\,0\,0 \\ 0\,0\,0\,0, & 0\,0\,0\,0, & 0\,0\,0\,0, \\ 0\,0\,0\,0 & 0\,0\,0\,0 & 0\,0\,0\,0 \end{matrix} \rightarrow \begin{matrix} 1\,0\,0 & 1\,1\,0 & 0\,0\,0 \\ 0\,0\,0, & 0\,0\,0, & 0\,0\,0, \\ 0\,0\,0 & 0\,0\,0 & 0\,0\,0 \end{matrix} \rightarrow \begin{matrix} 0\,0\,0 & 0\,0\,0 & 0\,0\,0 \\ 0\,0\,0, & 0\,0\,0, & 0\,0\,0, \\ 0\,0\,0 & 0\,1\,1 & 0\,0\,1 \end{matrix} \\ i_3{=}0 \quad i_3{=}1 \quad i_3{=}2 & i_3{=}0 \quad i_3{=}1 \quad i_3{=}2 & i_3{=}0 \quad i_3{=}1 \quad i_3{=}2 \end{array}$$

$$\downarrow^{\psi}$$

$$g(z)_1^{(1)} = z_1 z_2^2 z_3 + z_1^2 z_2^2 z_3 + z_1^2 z_2^2 z_3^2$$

Since $k = 1$, there is no multiplexing involved and $g^{(y)} = g_x^{(y)}$. The remaining sequences $a^{(2)}, \ldots, a^{(8)}$ can be transformed in a similar manner to construct the locally invertible encoder

$$G(z) = \begin{bmatrix} z_1 z_2^2 z_3 + z_1^2 z_2^2 z_3 + z_1^2 z_2^2 z_3^2 \\ z_2 + z_2^2 z_3^2 \\ z_1^2 z_2^2 + z_1 z_2 z_3 + z_1^2 z_2 z_3 + z_1^2 z_2 z_3^2 \\ z_2^2 + z_2 z_3^2 \\ z_1^2 z_2 + z_2 z_3 + z_1^2 z_3^2 \\ z_2 + z_3^2 + z_2^2 z_3^2 \\ z_1^2 + z_1^2 z_2^2 z_3 + z_1^2 z_2 z_3^2 \\ 1 + z_1^2 z_2 z_3 + z_2^2 z_3 + z_1^2 z_2^2 z_3^2 \end{bmatrix}^T .$$

5.4 Structural Properties of Locally Invertible Encoders

Locally invertible encoders possess many interesting and useful properties desirable in any m-D convolutional encoder. In this section we discuss these properties.

Definition 5.4.1. *A m-D convolutional encoder is said to be catastrophic if it encodes an infinite weight input sequence into a finite weight codeword.*

When such a finite weight codeword corresponding to an infinite weight input sequence is transmitted over a noisy channel, it is possible that a finite number of channel

errors can change all the nonzero symbols of this codeword into zeros, thereby causing the channel decoder to generate the all-zero codeword as an estimate. Thus, a finite number of channel errors can cause an infinite number of decoding errors. This situation of catastrophic error propagation is clearly undesirable and must be avoided.

The subject of catastrophicity was first treated in a paper by Massey and Sain [41] where they gave necessary and sufficient conditions for a 1-D polynomial encoder to be noncatastrophic. Fornasini and Valcher [18] have extended this result to the 2-D case over the Laurent polynomial ring $\mathbb{F}[z_1^{\pm}, z_2^{\pm}]$. Weiner [56] has generalized this test for the m-D case over the polynomial ring $R = \mathbb{F}[z_1, \ldots, z_m]$. The test for catastrophic encoders depends crucially on the ring or field over which the encoder is defined and generally involves computing the gcd of the full size minors of the encoder. The following proposition shows that a locally invertible encoder is noncatastrophic without having to compute the gcd of its full size minors.

In order to allow for input sequences of infinite weight we have to drop the finite support restriction on the input space. Let $R_\infty = \mathbb{F}[[z_1, \ldots, z_m]]$ be the power series ring in m variables over $\mathbb{F}$. Let $S_\infty = \psi^{-1}(R_\infty)$ be the equivalent sequence space representation of R_∞. So here, the input sequence space S_∞^k has infinite support, and contains some sequences $u \in S_\infty^k$ with $\mathrm{wt}(u) = \infty$. As usual we will consider the m-D convolutional encoder $G(z) \in R^{k\times n}$ to be polynomial (i.e. $R = \mathbb{F}[z_1, \ldots, z_m]$). Let $G \in S^{k\times n}$ be the equivalent sequence encoder with finite support. The m-D convolutional code

$$\mathcal{C}_\infty = \{v \in S_\infty^n : v = u * G\} \tag{5.35}$$

now has some codewords $v \in \mathcal{C}_\infty$ with $\mathrm{wt}(v) = \infty$.

Proposition 5.4.1. *A locally invertible encoder is noncatastrophic.*

Proof. Let $G(z) \in R^{k\times n}$ be a locally invertible encoder with memory orders $M_1, \ldots, M_m$. By definition $G = \psi^{-1}(G(z)) \in S^{k\times n}$ has an invertible subsequence map of size $\prod_{i=1}^m w_i$ between the input sequences in S_∞^k and output (codeword) sequences in $\mathcal{C}_\infty \subset S_\infty^n$.

To the contrary, assume that $G(z)$ is catastrophic. Then, there exists an infinite weight input sequence $u \in S_\infty^k$ that produces a finite weight codeword $v \in \mathcal{C}_\infty$. Let the coordinate $(l_1, \ldots, l_m)$ be the largest support of this finite weight codeword v. That is, $v_{(s_1,\ldots,s_m)} = 0$ for all $s_i > l_i$. Now, since $u \in S_\infty^k$ has infinite weight, there exists a nonzero subsequence $\hat{u}$ of u, of size $\prod_{i=1}^m w_i$, starting at some coordinate $(s_1, \ldots, s_m)$; $s_i > l_i$ in S_∞^k

that maps to an all-zero subsequence $\hat{v}$ of v, also of size $\prod_{i=1}^{m} w_i$, starting at the coordinate $(s_1 + M_1, \ldots, s_m + M_m)$ in S_∞^n. This is a contradiction to the fact that subsequence map is invertible, and we conclude that $G(z)$ is noncatastrophic. □

In Section 5.2 we saw that the subsequence map of a locally invertible encoder can be used to encode an input sequence into its corresponding codeword. We also saw that the inverse subsequence map can be used to invert the codeword back into the original input sequence. The following proposition shows that such an inversion is possible because a locally invertible encoder is ZP. The proof is constructive and provides a technique to extract an encoder inverse from the inverse subsequence map.

Proposition 5.4.2. *Let $\hat{G}^{-1}$ be the $w \times w$ inverse reduced encoding matrix of a locally invertible encoder $G(z) \in R^{k \times n}$. An encoder inverse $G(z)^{-1} \in R^{n \times k}$, such that $G(z)G(z)^{-1} = I_{k \times k}$, can be obtained from k columns of $\hat{G}^{-1}$.*

Proof. Let $v = u * G$ be the codeword corresponding to an input sequence $u \in S^k$. Since G has a subsequence map of size $\prod_{i=1}^{m} w_i$ between u and v, the codeword v can be considered to be made up of overlapping subsequences $\hat{v} \in S^n$ of size $\prod_{i=1}^{m} w_i$ that are shifted by n_i symbols in the ith dimension. Since the subsequence map is invertible, each output subsequence $\hat{v}$ can inverted into an input subsequence $\hat{u} \in S^k$ using the inverse subsequence map (5.9). Consecutive input subsequences can be put together piecemeal with a shift of k_i symbols and an overlap of $w_i - k_i$ symbols in the ith dimension to reconstruct the input sequence u. See for example Section 5.2.

The inverse subsequence map (5.9) is obtained from the rows of the inverse reduced encoding matrix $\hat{G}^{-1}$. Therefore, the subsequence inversion can be performed using matrix multiplication with the inverse reduced encoding matrix as shown below. Without loss of generality we can assume that row-major unfolding (5.17) is used to construct the reduced encoding matrix (5.5).

$$\hat{u} = \mathcal{U}_{fr}^{-1}\big(\mathcal{U}_{fr}(\hat{v})\hat{G}^{-1}\big) \tag{5.36}$$

Here, $\mathcal{U}_{fr}(\hat{v})$ is the 1-D sequence representation of the m-D output subsequence $\hat{v}$. When the subsequence inversion is viewed as shown in (5.36), each symbol attached to a coordinate of the m-D input subsequence $\hat{u}$ is obtained by multiplying the 1-D output subsequence $\mathcal{U}_f(\hat{v})$ with a column $\mathbf{g}_r^{-1}$ of the inverse reduced encoding matrix.

$$\hat{G}^{-1} = [\mathbf{p}_1, \ldots, \mathbf{p}_w]^T = [\mathbf{g}_1^{-1}, \ldots, \mathbf{g}_w^{-1}] \tag{5.37}$$

The overlapping symbols between consecutive input subsequences described above are now obtained from first $w-k$ columns of $\hat{G}^{-1}$. The columns $\mathbf{g}_1^{-1},\ldots,\mathbf{g}_{w-k}^{-1}$ that are responsible for the overlap right-shift the overlapping symbols of consecutive input subsequences during each step of the sequence inversion. It is the remaining (last) k columns of $\hat{G}^{-1}$ that produce new (non-overlapping) symbols of the sequence u during each step of the sequence inversion.

$$u^{(1)}_{(i_1,\ldots,i_m)},\ldots,u^{(k)}_{(i_1,\ldots,i_m)} = \mathcal{U}_{fr}^{-1}\big(\mathcal{U}_{fr}(\hat{v})[\mathbf{g}_{w-k+1}^{-1},\ldots,\mathbf{g}_{w}^{-1}]\big) \tag{5.38}$$

Here, $u^{(1)}_{(l_1,\ldots,l_m)},\ldots,u^{(k)}_{(l_1,\ldots,l_m)}$ is a k symbol block with order $\prod_{i=1}^{m}k_i$ that is attached to the coordinate $(l_1,\ldots,l_m)$ of the composite input sequence $u\in S^k$. The specific value of $(l_1,\ldots,l_m)$ depends on the starting coordinate of the output subsequence $\hat{v}$ that is under consideration. The k symbol blocks obtained during each successive step of the sequence inversion can be put together to construct the sequence u. Therefore, the entire input sequence u can be recovered from the codeword v using the last k columns. This in turn implies that the columns $\mathbf{g}_{w-k+1}^{-1},\ldots,\mathbf{g}_{w}^{-1}$ are equivalent to the inverse sequence generator $G^{-1}\in S^{n\times k}$, because they perform the operation $u=v*G^{-1}$.

Since the sequence v is in composite form with order $\prod_{i=1}^{m}n_i$, the columns of $\hat{G}^{-1}$ have to be in reverse composite form with order $\prod_{i=1}^{m}n_i$. Each of the last k columns of $\hat{G}^{-1}$ have to be folded back into m-D subsequences and then reversed n_i symbols at a time along each dimension to obtain the inverse composite generator sequences $g^{(y)^{-1}}$.

$$\mathcal{U}_{fr}^{-1}(\mathbf{g}_r^{-1})\rightarrow\mathcal{R}(g^{(y)^{-1}})\rightarrow g^{(y)^{-1}};\quad r=w-k+1,\ldots,w. \tag{5.39}$$

The polynomial representation of $g^{(y)^{-1}}$; $y=1,\ldots,k$ using the isomorphism ψ^n gives us the yth column

$$\psi^n(g^{(y)^{-1}}) = [g^{-1}(z)^{(1)}_y,\ldots,g^{-1}(z)^{(n)}_y]^T, \tag{5.40}$$

of the encoder inverse $G(z)^{-1}\in R^{n\times k}$ with elements $g^{-1}(z)^{(x)}_y\in R$. □

Pseudo-Inveres: Under the assumption that row-major unfolding was used in the construction of the reduced encoding matrix $\hat{G}$, the proof of Proposition 5.4.2 showed that the first $w-k$ columns $\mathbf{g}_1^{-1},\ldots,\mathbf{g}_{w-k}^{-1}$ of $\hat{G}^{-1}$ right-shift the overlapping symbols of consecutive input subsequences during each step of the sequence inversion. The encoder inverse $G(z)^{-1}\in R^{n\times k}$ was then extracted from the last k columns. Using a similar technique, it can be shown that the polynomial representation (5.39)-(5.40) of each set of k

columns in $\mathbf{g}_1^{-1}, \ldots, \mathbf{g}_{w-k}^{-1}$ yields a matrix $G(z)^{-1}_{(d_1,\ldots,d_m)} \in R^{n\times k}$, such that

$$G(z)G(z)^{-1}_{(d_1,\ldots,d_m)} = z_1^{d_1} \cdots z_m^{d_m} I_{k\times k}\,, \tag{5.41}$$

with $0 \leq d_i \leq \frac{w_i-k_i}{k_i}$ and $\sum_{i=1}^{m} d_i \neq 0$. There are a total of $\frac{w-k}{k}$ such matrices corresponding to the $w-k$ columns. We shall refer to these matrices as polynomial inverses with delay or *pseudo-inverses*. Here, the exponent d_i is considered as the *delay* in the ith dimension. The delay d_i is a multiple of k_i and corresponds to the right-shift caused by the k columns under consideration. The delay ranges form $0 \leq d_i \leq \frac{w_i-k_i}{k_i}$ due to the fact that the subsequences are of size w_i along the ith dimension, and can therefore have a maximum overlap of $w_i - k_i$. In Section 6.2 we will describe a method to detect errors using these pseudo-inverses.

The following properties are an immediate consequence of Proposition 5.4.2.

Corollary 5.4.3. *A locally invertible encoder is ZP.*

Proof. Follows from Theorem 3.4.1. □

Corollary 5.4.4. *Let $G(z) \in R^{k\times n}$ be a locally invertible encoder. Then, the rate $\frac{k}{n}$ m-D convolutional code $\mathcal{C}$ generated by $G(z)$ has a rate $\frac{n-k}{n}$ dual code $\mathcal{C}^{\perp}$.*

Proof. By Corollary 5.4.3 and Corollary 3.5.9. □

In Section 6.3 we will use the invertible subsequence map of the locally invertible encoder to derive an encoder for the dual code $\mathcal{C}^{\perp}$.

Example 5.4.1. Consider the 24×24 inverse reduced encoder matrix $\hat{G}^{-1} = [\mathbf{g}_1^{-1}, \ldots, \mathbf{g}_{24}^{-1}]$ shown in Figure 5.1(d) of the 2-D encoder $G(z) \in R^{2\times 6}$.

$$G(z) = \begin{bmatrix} z_1^2 z_2 & 0 & z_2 & z_1^2 & 1 & 0 \\ 0 & 1+z_1^2 z_2 & 0 & z_1 z_2 & z_1 & 1+z_1^2+z_2 \end{bmatrix}$$

Since row-major unfolding was used in the construction of $\hat{G}$, as per Proposition 5.4.2, the last $k=2$ columns $\mathbf{g}_{23}^{-1}$ and $\mathbf{g}_{24}^{-1}$ of $\hat{G}^{-1}$ correspond to the inverse composite sequence generator

$$G_c^{-1} = \{g^{(1)^{-1}}, \quad g^{(2)^{-1}}\} = \{\mathcal{R}(\mathcal{U}_{fr}^{-1}(\mathbf{g}_{23}^{-1})), \quad \mathcal{R}(\mathcal{U}_{fr}^{-1}(\mathbf{g}_{24}^{-1}))\}.$$

The encoder inverse $G(z)^{-1} \in R^{6\times 2}$

$$G(z)^{-1} = \begin{bmatrix} 1 & z_1 & 1 & z_2 & 1+z_2 & z_1 z_2 \\ 0 & 1 & z_1 & 0 & z_1 z_2 & 0 \end{bmatrix}^T,$$

is obtained by folding these columns and reversing them $n_1 = 2$ and $n_2 = 3$ symbols at a time as shown below.

$$\mathbf{g}_{23}^{-1} = 000001\ 000110\ 010000\ 101010$$
$$\mathbf{g}_{24}^{-1} = 000010\ 000000\ 001000\ 010000$$

$$\mathcal{U}_{fr}^{-1}(\mathbf{g}_{23}^{-1}) \quad g^{(1)^{-1}} \quad \mathcal{U}_{fr}^{-1}(\mathbf{g}_{24}^{-1}) \quad g^{(2)^{-1}}$$

$$\begin{array}{c} 00\ 00 \\ 00\ 01 \\ 01\ 10 \\ \\ 01\ 10 \\ 00\ 10 \\ 00\ 10 \end{array} \rightarrow \begin{array}{c} 00\ 00 \\ 01\ 00 \\ 10\ 01 \\ \\ 10\ 01 \\ 10\ 00 \\ 10\ 00 \end{array} \rightarrow \begin{array}{c} 10\ 01 \\ 10\ 00 \\ 10\ 00 \\ \\ 00\ 00 \\ 01\ 00 \\ 10\ 01 \end{array} \overset{\psi^6}{\rightarrow} \begin{bmatrix} 1 \\ z_1 \\ 1 \\ z_2 \\ 1+z_2 \\ z_1 z_2 \end{bmatrix}, \quad \begin{array}{c} 00\ 00 \\ 00\ 00 \\ 10\ 00 \\ \\ 00\ 01 \\ 10\ 00 \\ 00\ 00 \end{array} \rightarrow \begin{array}{c} 00\ 00 \\ 00\ 00 \\ 00\ 10 \\ \\ 01\ 00 \\ 00\ 10 \\ 00\ 00 \end{array} \rightarrow \begin{array}{c} 01\ 00 \\ 00\ 10 \\ 00\ 00 \\ \\ 00\ 00 \\ 00\ 00 \\ 00\ 10 \end{array} \overset{\psi^6}{\rightarrow} \begin{bmatrix} 0 \\ 1 \\ z_1 \\ 0 \\ z_1 z_2 \\ 0 \end{bmatrix}$$

The first $w - k = 22$ columns $\mathbf{g}_1^{-1}, \ldots \mathbf{g}_{22}^{-1}$ of $\hat{G}^{-1}$ correspond to the pseudo-inverses defined in (5.41). For example, the polynomial representation of columns $\mathbf{g}_9^{-1}$ and $\mathbf{g}_{10}^{-1}$ as shown below

$$\mathbf{g}_{9}^{-1} = 000000\ 000000\ 100000\ 000000$$
$$\mathbf{g}_{10}^{-1} = 000001\ 000110\ 000000\ 101000$$

$$\mathcal{U}_{fr}^{-1}(\mathbf{g}_{9}^{-1}) \quad g^{(1)^{-1}} \quad \mathcal{U}_{fr}^{-1}(\mathbf{g}_{10}^{-1}) \quad g^{(2)^{-1}}$$

$$\begin{array}{c} 00\ 00 \\ 00\ 00 \\ 00\ 00 \\ \\ 10\ 00 \\ 00\ 00 \\ 00\ 00 \end{array} \rightarrow \begin{array}{c} 00\ 00 \\ 00\ 00 \\ 00\ 00 \\ \\ 00\ 10 \\ 00\ 00 \\ 00\ 00 \end{array} \rightarrow \begin{array}{c} 00\ 10 \\ 00\ 00 \\ 00\ 00 \\ \\ 00\ 00 \\ 00\ 00 \\ 00\ 00 \end{array} \overset{\psi^6}{\rightarrow} \begin{bmatrix} z_1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{array}{c} 00\ 00 \\ 00\ 01 \\ 01\ 10 \\ \\ 00\ 10 \\ 00\ 10 \\ 00\ 00 \end{array} \rightarrow \begin{array}{c} 00\ 00 \\ 01\ 00 \\ 10\ 01 \\ \\ 10\ 00 \\ 10\ 00 \\ 00\ 00 \end{array} \rightarrow \begin{array}{c} 10\ 00 \\ 10\ 00 \\ 00\ 00 \\ \\ 00\ 00 \\ 01\ 00 \\ 10\ 01 \end{array} \overset{\psi^6}{\rightarrow} \begin{bmatrix} 1 \\ 0 \\ 1 \\ z_2 \\ z_2 \\ z_1 z_2 \end{bmatrix}$$

corresponds to the pseudo-inverse $G(z)_{(3,1)}^{-1} \in R^{6\times 2}$

$$G(z)_{(3,1)}^{-1} = \begin{bmatrix} z_1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & z_2 & z_2 & z_1 z_2 \end{bmatrix}^T,$$

with delays $d_1 = 3$ and $d_2 = 1$, such that

$$G(z)G(z)_{(3,1)}^{-1} = z_1^3 z_2^1 I_{2\times 2}.$$

$$G(z)^{-1} \quad \begin{bmatrix} 1 & 0 \\ z_1 & 1 \\ 1 & z_1 \\ z_2 & 0 \\ 1{+}z_2 & z_1z_2 \\ z_1z_2 & 0 \end{bmatrix} \qquad G(z)^{-1}_{(1,0)} \quad \begin{bmatrix} z_1 & 1 \\ 1{+}z_2 & z_1 \\ z_1 & 1 \\ z_1z_2 & z_2 \\ z_1{+}z_1z_2 & z_2 \\ 1 & z_1z_2 \end{bmatrix} \qquad G(z)^{-1}_{(2,0)} \quad \begin{bmatrix} 0 & z_1 \\ 0 & 1{+}z_2 \\ 1 & z_1 \\ 1 & z_1z_2 \\ z_2 & z_1z_2 \\ 0 & 1 \end{bmatrix}$$

$$G(z)^{-1}_{(3,0)} \quad \begin{bmatrix} 0 & 1 \\ 0 & z_1 \\ z_1 & 0 \\ z_1 & z_2 \\ z_1z_2 & 0 \\ 0 & z_1{+}z_1z_2 \end{bmatrix} \qquad G(z)^{-1}_{(0,1)} \quad \begin{bmatrix} 0 & z_1 \\ 0 & z_2 \\ 1 & 0 \\ 0 & z_1z_2 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad G(z)^{-1}_{(1,1)} \quad \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ z_1 & 1 \\ 0 & 0 \\ 0 & z_2 \\ 0 & 0 \end{bmatrix}$$

$$G(z)^{-1}_{(2,1)} \quad \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & z_1 \\ 0 & 0 \\ 0 & z_1z_2 \\ 0 & 0 \end{bmatrix} \qquad G(z)^{-1}_{(3,1)} \quad \begin{bmatrix} z_1 & 1 \\ 0 & 0 \\ 0 & 1 \\ 0 & z_2 \\ 0 & z_2 \\ 0 & z_1z_2 \end{bmatrix} \qquad G(z)^{-1}_{(0,2)} \quad \begin{bmatrix} 0 & z_1 \\ 0 & z_2 \\ z_2 & z_1 \\ 0 & z_1z_2 \\ 0 & z_1z_2 \\ 0 & z_2 \end{bmatrix}$$

$$G(z)^{-1}_{(1,2)} \quad \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ z_1z_2 & 0 \\ 0 & z_2 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad G(z)^{-1}_{(2,2)} \quad \begin{bmatrix} z_2 & z_1 \\ 0 & 0 \\ 0 & 0 \\ 0 & z_1z_2 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad G(z)^{-1}_{(3,2)} \quad \begin{bmatrix} z_1z_2 & 0 \\ 0 & z_1z_2 \\ 0 & 1 \\ 0 & 0 \\ 0 & z_2 \\ 0 & 0 \end{bmatrix}$$

Figure 5.8: Polynomial representation of the inverse reduced encoding matrix.

There are a total of $\frac{w-k}{k} = 11$ such pseudo-inverses with the delay d_1 ranging from 0 to $\frac{w_1-k_1}{k_1} = 3$ and the delay d_2 ranging from 0 to $\frac{w_2-k_2}{k_2} = 2$ as shown in Figure 5.8. ▲

The following proposition was first proved by Dholakia [11]. The statement of the proposition is a trivial rephrasing (see Section 5.5) of [11, Theorem 3.2].

Proposition 5.4.5. *A rate $\frac{1}{2}$ 1-D convolutional encoder is basic if and only if it is locally invertible.*

Proof. For $m = 1$, we have $R = \mathbb{F}[z_1]$. Let $G(z_1) = [g(z_1)^{(1)}, g(z_1)^{(2)}] \in R^{1\times 2}$ be a rate $\frac{1}{2}$ 1-D convolutional encoder with memory order M_1. Assume $G(z_1)$ is locally invertible. Then, from Corollary 5.4.3, $G(z_1)$ is also basic.

For the converse, assume $G(z_1)$ is basic. Since $w = \frac{1\times 2\times M_1}{2-1} = 2M_1$ is a positive integer, $G(z_1)$ has a subsequence map of length $2M_1$. The $2M_1 \times 2M_1$ reduced encoding matrix $\hat{G}$ that represents this subsequence map is shown in (5.47). The blanks in the matrix denote zeros.

Assume $\hat{G}$ is singular. This implies that there exists a nonzero column vector $\lambda \in \mathbb{F}^{2M_1\times 1}$

$$\lambda = [\underbrace{\alpha_{M_1-1}, \beta_{M_1-1}, \alpha_{M_1-2}, \beta_{M_1-2}, \ldots, \alpha_0, \beta_0}_{2M_1}]^T \tag{5.42}$$

with elements $\alpha_i, \beta_i \neq 0$, $i = 0, \ldots, M_1 - 1$, such that

$$\hat{G}\lambda = [\underbrace{0, \ldots, 0}_{2M_1}]^T. \tag{5.43}$$

This is equivalent to the statement

$$\alpha(z_1)g(z_1)^{(1)} + \beta(z_1)g(z_1)^{(2)} = 0, \tag{5.44}$$

where

$$\alpha(z_1) = \alpha_0 + \alpha_1 z_1 + \cdots + \alpha_{M_1-1} z_1^{M_1-1}, \tag{5.45}$$

$$\beta(z_1) = \beta_0 + \beta_1 z_1 + \cdots + \beta_{M_1-1} z_1^{M_1-1}. \tag{5.46}$$

We have memory order $M_1 = \max\{\deg g(z_1)^{(1)}, \deg g(z_1)^{(2)}\}$. Without loss of generality, let $M_1 = \deg g(z_1)^{(2)}$. From (5.44) it follows that either $g(z_1)^{(2)} \mid \alpha(z_1)g(z_1)^{(1)}$, or $\alpha(z_1) = \beta(z_1) = 0$. But, $\alpha_i, \beta_i \neq 0$ in (5.42) implies that $\alpha(z_1), \beta(z_1) \neq 0$. Since $G(z_1)$ is ZP, $g(z_1)^{(2)} \nmid g(z_1)^{(1)}$, and so $g(z_1)^{(2)} \mid \alpha(z_1)$. But, $\deg g(z_1)^{(2)} > \deg \alpha(z_1)$. Therefore, $\alpha(z_1) = \beta(z_1) = 0$ and λ is a all-zero vector. This contradicts our initial assumption that $\hat{G}$ is singular. Hence, $\hat{G}$ is indeed nonsingular and so $G(z_1)$ locally invertible. □

$$\hat{G} = \begin{bmatrix}
g_{(M_1)}^{(1)} & g_{(M_1)}^{(2)} & & & & & \\
g_{(M_1-1)}^{(1)} & g_{(M_1-1)}^{(2)} & g_{(M_1)}^{(1)} & g_{(M_1)}^{(2)} & & & \\
\vdots & \vdots & g_{(M_1-1)}^{(1)} & g_{(M_1-1)}^{(2)} & & & \\
\vdots & \vdots & \vdots & \vdots & \ddots & & \\
\vdots & \vdots & \vdots & \vdots & & g_{(M_1)}^{(1)} & g_{(M_1)}^{(2)} \\
g_{(0)}^{(1)} & g_{(0)}^{(2)} & \vdots & \vdots & & g_{(M_1-1)}^{(1)} & g_{(M_1-1)}^{(2)} \\
 & & g_{(0)}^{(1)} & g_{(0)}^{(2)} & & \vdots & \vdots \\
 & & & & & \vdots & \vdots \\
 & & & & & \vdots & \vdots \\
 & & & & & g_{(0)}^{(1)} & g_{(0)}^{(2)}
\end{bmatrix} \tag{5.47}$$

Here we note that the reduced encoding matrix in (5.47) bears a strong resemblance to the well known *Sylvester matrix* [1, 33, 23] that is used to compute the resultant of two univariate polynomials. In fact the two are equivalent under certain conditions. We explain.

Definition 5.4.2. *Let $g(z_1)^{(1)}, g(z_1)^{(2)} \in \mathbb{F}[z_1]$ be two univariate polynomials of degrees M_1 and N_1 as shown below.*

$$g(z_1)^{(1)} = g^{(1)}_{(M_1)} z_1^{M_1} + g^{(1)}_{(M_1-1)} z_1^{M_1-1} + \cdots + g^{(1)}_{(1)} z_1 + g^{(1)}_{(0)} \tag{5.48}$$

$$g(z_1)^{(2)} = g^{(2)}_{(N_1)} z_1^{N_1} + g^{(2)}_{(N_1-1)} z_1^{N_1-1} + \cdots + g^{(1)}_{(1)} z_1 + g^{(2)}_{(0)} \tag{5.49}$$

The Sylvester matrix $\mathcal{S}$ associated with $g(z_1)^{(1)}$ and $g(z_1)^{(2)}$ is a $(M_1 + N_1) \times (M_1 + N_1)$ matrix of the form

$$\mathcal{S} = \begin{bmatrix} g^{(1)}_{(M_1)} & g^{(1)}_{(M_1-1)} & \cdots & g^{(1)}_{(0)} & & & \\ & g^{(1)}_{(M_1)} & g^{(1)}_{(M_1-1)} & \cdots & g^{(1)}_{(0)} & & \\ & & \ddots & & & & \\ & & & g^{(1)}_{(M_1)} & g^{(1)}_{(M_1-1)} & \cdots & g^{(1)}_{(0)} \\ g^{(2)}_{(N_1)} & g^{(2)}_{(N_1-1)} & \cdots & g^{(2)}_{(0)} & & & \\ & g^{(2)}_{(N_1)} & g^{(2)}_{(N_1-1)} & \cdots & g^{(2)}_{(0)} & & \\ & & \ddots & & & & \\ & & & g^{(2)}_{(N_1)} & g^{(2)}_{(N_1-1)} & \cdots & g^{(2)}_{(0)} \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ \updownarrow N_1 \text{ rows} \\ \\ \\ \\ \updownarrow M_1 \text{ rows} \end{matrix} \tag{5.50}$$

$$\begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix}$$

$$\begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix}$$

$$\longleftarrow \quad M_1 + N_1 \text{ columns} \quad \longrightarrow$$

For example, the Sylvester matrix of $g(z_1)^{(1)} = g^{(1)}_{(3)} z_1^3 + g^{(1)}_{(2)} z_1^2 + g^{(1)}_{(1)} z_1 + g^{(1)}_{(0)}$ and $g(z_1)^{(2)} = g^{(2)}_{(2)} z_1^2 + g^{(2)}_{(1)} z_1 + g^{(2)}_{(0)}$ is an 5×5 matrix of the form

$$\mathcal{S} = \begin{bmatrix} g^{(1)}_{(3)} & g^{(1)}_{(2)} & g^{(1)}_{(1)} & g^{(1)}_{(0)} & \\ & g^{(1)}_{(3)} & g^{(1)}_{(2)} & g^{(1)}_{(1)} & g^{(1)}_{(0)} \\ g^{(2)}_{(2)} & g^{(2)}_{(1)} & g^{(2)}_{(0)} & & \\ & g^{(2)}_{(2)} & g^{(2)}_{(1)} & g^{(2)}_{(0)} & \\ & & g^{(2)}_{(2)} & g^{(2)}_{(1)} & g^{(2)}_{(0)} \end{bmatrix}.$$

The generator polynomials $g(z_1)^{(1)}, g(z_1)^{(2)}$ in Proposition 5.4.5 can be of arbitrary degrees, and the memory order is given by $M_1 = \max\{\deg g(z_1)^{(1)}, \deg g(z_1)^{(2)}\}$. However, when $\deg g(z_1)^{(1)} = \deg g(z_1)^{(2)} = M_1$, we have $\hat{G} = \mathcal{S}^T E$, where $\hat{G}$ is the reduced encoding

matrix (5.47), $\mathcal{S}$ is the Sylvester matrix (5.50) and E is an $M_1 \times M_1$ elementary matrix used for column interchange.

Akritas in [1], discusses a lesser know matrix that was formulated by Sylvester [53] in 1853. We will denote this matrix by $\hat{\mathcal{S}}$. The determinant of $\hat{\mathcal{S}}$ is called Sylvester's form of the resultant in [1, pg. 329]. In fact, $\hat{G} = \hat{\mathcal{S}}^T$, where $\hat{G}$ is the reduced encoding matrix (5.47). Since $\hat{\mathcal{S}}$ is singular if and only if $g(z_1)^{(1)}$ and $g(z_1)^{(2)}$ have a common zero, by virtue of Proposition 5.4.2 we have essentially proved the following.

Corollary 5.4.6. *Let $a(x)$ of degree r, and $b(x)$ of degree s, $s \geq r$, be two univariate polynomials in $\mathbb{F}[x]$ that are devoid of common zeros. Let $\hat{\mathcal{S}}$ be the $2s \times 2s$ nonsingular Sylvester matrix associated with $a(x)$ and $b(x)$. Then, the jth column ($j = 1, \ldots, 2s$) of $(\hat{\mathcal{S}}^{-1})^T$ can be used to obtain polynomials $f(x)_j, g(x)_j \in \mathbb{F}[x]$ such that*

$$a(x)f(x)_j + b(x)g(x)_j = x^{2s-j}. \tag{5.51}$$

Example 5.4.2. Consider the polynomials $a(x) = 1 + x$ and $b(x) = 1 + x + x^2$ in $\mathbb{F}_2[x]$ of degrees $r = 1$ and $s = 2$ respectively. Clearly, $a(x)$ and $b(x)$ are devoid of common zeros and so the $2s \times 2s = 4 \times 4$ reduced encoding matrix $\hat{G}$ associated with the rate $\frac{k}{n} = \frac{1}{2}$ encoder $[1 + x \;\; 1 + x + x^2]$ is nonsingular.

$$\hat{\mathcal{S}}^T = \hat{G} = \begin{bmatrix} 0 & 1 & & \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ & & 1 & 1 \end{bmatrix}, \quad (\hat{\mathcal{S}}^{-1})^T = \hat{G}^{-1} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

As per Proposition 5.4.2, the jth column of $\hat{G}^{-1}$ has to be reversed $n = 2$ symbols at a time to obtain the polynomials $f(x)_j, g(x)_j$ such that $a(x)f(x)_j + b(x)g(x)_j = x^{4-j}$.

$$\begin{aligned}
&\mathbf{g}_4^{-1} = 1001 \rightarrow \{01 \;\; 10\} \xrightarrow{\psi^2} [x \;\; 1], && (1+x)(x) + (1+x+x^2)(1) = 1. \\
&\mathbf{g}_3^{-1} = 1011 \rightarrow \{11 \;\; 10\} \xrightarrow{\psi^2} [1+x \;\; 1], && (1+x)(1+x) + (1+x+x^2)(1) = x. \\
&\mathbf{g}_2^{-1} = 0011 \rightarrow \{11 \;\; 00\} \xrightarrow{\psi^2} [1 \;\; 1], && (1+x)(1) + (1+x+x^2)(1) = x^2. \\
&\mathbf{g}_1^{-1} = 1100 \rightarrow \{00 \;\; 11\} \xrightarrow{\psi^2} [x \;\; x], && (1+x)(x) + (1+x+x^2)(x) = x^3.
\end{aligned}$$

In Section 3.6 we defined a 1-D ZP (basic) encoder as minimal-basic if its overall constraint length was minimal over all equivalent basic encoding matrices. Next we show that a 1-D locally invertible encoder satisfies this property.

Proposition 5.4.7. *A 1-D locally invertible encoder is minimal-basic.*

Proof. Locally invertible ⇒ basic has been proved in Corollary 5.4.3. Let $G(z_1)$ be a rate $\frac{k}{n}$ 1-D locally invertible encoder with memory order M_1. Consider the $w \times w$ reduced encoding matrix $\hat{G}$ (5.34) associated with the encoder $G(z_1)$ as shown below.

$$\hat{G} = \begin{bmatrix} \mathbf{G}_{M_1} & & & \\ \mathbf{G}_{M_1-1} & \mathbf{G}_{M_1} & & \\ \vdots & \mathbf{G}_{M_1-1} & & \\ \vdots & \vdots & \ddots & \\ \vdots & \vdots & & \mathbf{G}_{M_1} \\ \mathbf{G}_0 & \vdots & & \mathbf{G}_{M_1-1} \\ & \mathbf{G}_0 & & \vdots \\ & & & \vdots \\ & & & \mathbf{G}_0 \end{bmatrix}, \quad \mathbf{G}_j = \begin{bmatrix} g_{1(j)}^{(1)} & g_{1(j)}^{(2)} & \cdots & g_{1(j)}^{(n)} \\ g_{2(j)}^{(1)} & g_{2(j)}^{(2)} & \cdots & g_{2(j)}^{(n)} \\ \vdots & & & \\ g_{k(j)}^{(1)} & g_{k(j)}^{(2)} & \cdots & g_{k(j)}^{(n)} \end{bmatrix}. \tag{5.52}$$

The blanks in the reduced encoding matrix $\hat{G}$ denote zeros. The only nonzero elements in the first k rows of $\hat{G}$ are the elements of the $k \times n$ submatrix

$$\mathbf{G}_{M_1} = \begin{bmatrix} g_{1(M_1)}^{(1)} & g_{1(M_1)}^{(2)} & \cdots & g_{1(M_1)}^{(n)} \\ g_{2(M_1)}^{(1)} & g_{2(M_1)}^{(2)} & \cdots & g_{2(M_1)}^{(n)} \\ \vdots & & & \\ g_{k(M_1)}^{(1)} & g_{k(M_1)}^{(2)} & \cdots & g_{k(M_1)}^{(n)} \end{bmatrix}. \tag{5.53}$$

Since $\hat{G}$ is nonsingular, $\mathbf{G}_{M_1}$ cannot have an all-zero row. Therefore, the constraint lengths $\nu_{1,x}$ of all the rows of $G(z_1)$ have to be equal. That is, $\nu_{1,x} = M_1$ for $x = 1, \ldots, k$. This implies that the elements $g_{x(M_1)}^{(y)}$ of $\mathbf{G}_{M_1}$ are the coefficients of the highest-degree terms in each row of $G(z_1)$. So $\mathbf{G}_{M_1} = G_{\nu_x}$, where G_{ν_x} is the indicator matrix defined in (3.13). Since $\hat{G}$ is nonsingular, we have $\text{rank}(G_{\nu_x}) = k$. Hence, by Theorem 3.6.2, the locally invertible encoder $G(z_1)$ is minimal-basic. □

5.5 Related Work

In [6], Bitzer and Vouk introduced a Table-Driven decoder for 1-D convolutional codes based on the concept of one-to-one mapping between information and encoded blocks

of equal length. The rate $\frac{1}{2}$ 1-D convolutional encoders used in this technique later came to be know as locally invertible encoders [11, 12, 16]. The structural properties of these rate $\frac{1}{2}$ encoders have been well documented in [11, Sec. 3.1]. An extension of these results (under the name locally invertible encoders) for the rate $\frac{1}{n}$ case and for the rate $\frac{k}{n}$ case ($k \geq 1$) have have been reported in [11, 4, 5] and [13] respectively. In this section we discuss the similarities and differences between the work presented in this dissertation with the publications cited above. Before we can get to the discussion we have to establish the following results.

Let $G = [g^{(1)}, \ldots, g^{(n)}] \in S^{1\times n}$ be a rate $\frac{1}{n}$ 1-D convolutional encoder with constraint length ν_1. We denote the generator sequences of G as $g^{(y)} = \left(g^{(y)}_{(0)}, g^{(y)}_{(1)}, \ldots, g^{(y)}_{(\nu_1)}\right)$. Bahl and Jelinek [3] have defined the notion of nondegenerate encoders as follows.

Definition 5.5.1. *A rate $\frac{1}{n}$ 1-D convolutional encoder with constraint length ν_1 is nondegenerate if*

i) there exists $y_1 \in (1, 2, \ldots, n)$ such that $g^{(y_1)}_{(0)}$ is nonzero, and

ii) there exists $y_2 \in (1, 2, \ldots, n)$ such that $g^{(y_2)}_{(\nu_1)}$ is nonzero, and

iii) there exists $y_1, y_2 \in (1, 2, \ldots, n)$, $y_1 \neq y_2$, such that $g^{(y_1)} \neq g^{(y_2)}$.

Remark 5.5.1. We note here that condition *(ii)* in Definition 5.5.1 is redundant since a rate $\frac{1}{n}$ 1-D convolutional encoder with constraint length ν_1 has to have at least one $g^{(y)}$ where $g^{(y)}_{(\nu_1)}$ is nonzero.

In Section 5.4 Definition 5.4.1, we defined an m-D convolutional encoder as catastrophic if it encoded an infinite weight input sequence into a finite weight codeword. The following theorem due to Massey and Sain [41] gives necessary and sufficient conditions for a 1-D polynomial encoder to be noncatastrophic.

Theorem 5.5.1. *Let $R = \mathbb{F}[z_1]$. A 1-D convolutional encoder $G(z_1) \in R^{k\times n}$ is noncatastrophic if the gcd of the $k \times k$ minors of $G(z_1)$ is a power of z_1.*

Proposition 5.5.2. *A rate $\frac{1}{n}$ 1-D convolutional encoder that is both nondegenerate and noncatastrophic is basic.*

Proof. Here $m = 1$, so $R = \mathbb{F}[z_1]$. Let $G(z_1) = [g(z_1)^{(1)}, \ldots, g(z_1)^{(n)}] \in R^{1\times n}$ be a rate $\frac{1}{n}$ nondegenerate and noncatastrophic 1-D convolutional encoder. Since $G(z_1)$ is noncatastrophic, by Theorem 5.5.1 we have $\gcd(g(z_1)^{(1)}, \ldots, g(z_1)^{(n)}) = \alpha z_1^d$, where $d \geq 0$ and

$\alpha \in \mathbb{F}$. But since $G(z_1)$ is also nondegenerate, by Definition 5.5.1 condition *(i)*, there exists a $y_1 \in (1, 2, \ldots, n)$ such that $g(z_1)^{(y_1)}$ has a nonzero constant term. Therefore, the gcd of $(g(z_1)^{(1)}, \ldots, g(z_1)^{(n)})$ is a unit in R. This, by Definition 3.3.1 makes $G(z_1)$ MP. From Theorem 3.3.1, for $m = 1$ we have ZP≡MP. So by Theorem 3.4.1 it follows that $G(z_1)$ is basic. □

Proposition 5.5.3. *A rate $\frac{1}{2}$ 1-D convolutional encoder is basic if and only if it is both nondegenerate and noncatastrophic.*

Proof. Here $m = 1$, so $R = \mathbb{F}[z_1]$. Assume $G(z_1) = [g(z_1)^{(1)}, g(z_1)^{(2)}] \in R^{1\times 2}$ is a basic 1-D convolutional encoder. By Theorem 3.4.1, $G(z_1)$ is ZP, and so $g(z_1)^{(1)}$ and $g(z_1)^{(2)}$ are relatively prime. Therefore, $G(z_1)$ is noncatastrophic. The relative primeness of $g(z_1)^{(1)}$ and $g(z_1)^{(2)}$ also implies that either $g(z_1)^{(1)}$ or $g(z_1)^{(2)}$ has a nonzero constant term and that $g(z_1)^{(1)} \neq g(z_1)^{(2)}$. So conditions *(i)* and *(iii)* in Definition 5.5.1 are satisfied. For condition *(ii)* in Definition 5.5.1 see Remark 5.5.1. Hence $G(z_1)$ is nondegenerate. The converse follows from Proposition 5.5.2. □

Remark 5.5.2. It is important to note that for $(n > 2)$, rate $\frac{1}{n}$ 1-D convolutional encoders: *basic $\not\Rightarrow$ nondegenerate and noncatastrophic.* This follows from the fact that there exist *systematic encoders* of the form $G(z_1) = [1, g(z_1)^{(2)}, g(z_1)^{(3)}, \ldots, g(z_1)^{(n)}] \in R^{1\times n}$, with $g(z_1)^{(2)} = g(z_1)^{(3)}$, that are degenerate because Definition 5.5.1 (iii) is not satisfied, but are *basic* because they have trivial inverses of the form $G(z_1)^{-1} = [1, 0, \ldots, 0]^T \in R^{n\times 1}$.

Proposition 5.5.4. *A rate $\frac{1}{n}$ 1-D locally invertible encoder is nondegenerate.*

Proof. Let $G(z_1) \in R^{1\times n}$ be a rate $\frac{1}{n}$ 1-D locally invertible encoder with constraint length ν_1. Since the encoder has only one row, we have $\nu_1 = M_1$, where M_1 is the memory order of the encoder. Since $G(z_1)$ is locally invertible, it has a subsequence map of length $w = \frac{nM_1}{n-1} = \frac{n\nu_1}{n-1}$. The $w \times w$ reduced encoding matrix $\hat{G}$ that represents this subsequence map is shown in (5.54). The blanks in the matrix denote zeros.

Since $\hat{G}$ is nonsingular, the last row of $\hat{G}$ cannot be a zero row, so condition *(i)* in Definition 5.5.1 is satisfied. For condition *(ii)* in Definition 5.5.1 see Remark 5.5.1. Finally, since $\hat{G}$ is nonsingular, no two columns of $\hat{G}$ can be identical, so condition *(iii)* in Definition 5.5.1 is satisfied. □

$$\hat{G} = \begin{bmatrix}
g^{(1)}_{(M_1)} & \cdots & g^{(n)}_{(M_1)} & & & & & & & & \\
g^{(1)}_{(M_1-1)} & \cdots & g^{(n)}_{(M_1-1)} & g^{(1)}_{(M_1)} & \cdots & g^{(n)}_{(M_1)} & & & & & \\
\vdots & & \vdots & g^{(1)}_{(M_1-1)} & \cdots & g^{(n)}_{(M_1-1)} & & & & & \\
\vdots & & \vdots & \vdots & & \vdots & \ddots & & & & \\
\vdots & & \vdots & \vdots & & \vdots & & g^{(1)}_{(M_1)} & \cdots & g^{(n)}_{(M_1)} & \\
g^{(1)}_{(0)} & \cdots & g^{(n)}_{(0)} & \vdots & & \vdots & & g^{(1)}_{(M_1-1)} & \cdots & g^{(n)}_{(M_1-1)} & \\
 & & & g^{(1)}_{(0)} & \cdots & g^{(n)}_{(0)} & & \vdots & & \vdots & \\
 & & & & & & & \vdots & & \vdots & \\
 & & & & & & & \vdots & & \vdots & \\
 & & & & & & & g^{(1)}_{(0)} & \cdots & g^{(n)}_{(0)} &
\end{bmatrix} \tag{5.54}$$

Fact 5.5.1. By virtue of Propositions 5.4.1, 5.5.4, 5.5.2 and Propositions 5.4.5, 5.5.3 we have established:

i) Rate $\frac{1}{n}$ 1-D locally invertible encoder $\Rightarrow$ nondegenerate and noncatastrophic $\Rightarrow$ basic.

ii) Rate $\frac{1}{2}$ 1-D locally invertible encoder $\equiv$ nondegenerate and noncatastrophic $\equiv$ basic.

We are interested in these results for the following reasons. In [11, 12, 16], rate $\frac{1}{2}$ 1-D convolutional encoders with constraint length ν_1 that have an invertible reduced encoding matrix $[\mathbf{G}]_{2\nu_1 \times 2_{\nu_1}}$ are called locally invertible encoders. Since a rate $\frac{1}{2}$ encoder has only one row, we have $\nu_1 = M_1$. Substituting the values $m = 1, k = 1$ and $n = 2$ in (5.2), gives us a subsequence mapping length of $w = 2M_1 = 2\nu_1$. Therefore the reduced encoding matrix defined in (5.19) and illustrated in (5.47) is same as $[\mathbf{G}]_{2\nu_1 \times 2_{\nu_1}}$. So a rate $\frac{1}{2}$ 1-D locally invertible encoder defined in this dissertation (see Definition 5.1.2) is the same as that found in [6, 11, 12, 16]. In particular, by using the result from Proposition 5.5.3, we see that the statement of Proposition 5.4.5 is just a trivial rephrasing of [11, Theorem 3.2.].

Assuming $\mathbb{F} = \mathbb{F}_2$, in [11, 4, 5] it is said that a rate $\frac{1}{n}$ 1-D locally invertible encoder with constraint length ν_1 has a one-to-one correspondence between $\mathbf{w}$-bit blocks of information and encoded bits, for some finite length $\mathbf{w}$. The first ν_1 bits of a $\mathbf{w}$-bit information block represent the current encoder state. The remaining $\mathbf{w} - \nu_1$ bits generate $n(\mathbf{w} - \nu_1)$

encoded bits, from which the encoded bits involved in the one-to-one correspondence are chosen. A $\mathbf{w} \times \mathbf{w}$ invertible matrix $[\mathbf{G}]_{\mathbf{w}\times\mathbf{w}}$ representing the one-to-one correspondence is formed by selecting $\mathbf{w}$ linearly independent columns out of the $n(\mathbf{w}-\nu_1)$ columns in $[\mathbf{G}]_{\mathbf{w}\times n(\mathbf{w}-\nu_1)}$, where $[\mathbf{G}]_{\mathbf{w}\times n(\mathbf{w}-\nu_1)}$ is a $\mathbf{w} \times n(\mathbf{w}-\nu_1)$ section of the semi-infinite generator matrix $\mathbf{G}$. The $\mathbf{w} \times \mathbf{w}$ matrix $[\mathbf{G}]_{\mathbf{w}\times\mathbf{w}}$ described above is not the same as the $w \times w$ reduced encoding matrix $\hat{G}$ defined in (5.19) and illustrated in (5.54). We have shown in Section 5.3, that $\hat{G}$ is a $w \times w$ submatrix of the semi-infinite generator matrix. In particular, the w columns of $\hat{G}$ are obtained from *consecutive* columns of $\mathbf{G}$, and so $\hat{G}$ can be a *square* matrix only if the parameters $k = 1, n, M_1$ are such that $w = \frac{nM_1}{n-1}$ is a positive integer. As a result, a rate $\frac{1}{n}$ 1-D locally invertible encoder as defined Definition 5.1.2 is not the same as that found in [11, 12, 4, 5].

Due to the difference in definitions, some of the results reported in [11, 4, 5] do not apply to the locally invertible encoders defined in this dissertation. In [11, 4, 5] it has been shown that *a rate* $\frac{1}{n}$ *1-D convolutional encoder is both noncatastrophic and nondegenerate if and only if it is locally invertible.* While this result holds true for the rate $\frac{1}{2}$ case as proved in Fact 5.5.1 (ii), and the "if" part has been proved in Fact 5.5.1 (i), the reverse implication does not hold. That is, for $n > 2$, a rate $\frac{1}{n}$ 1-D convolutional encoder that is both nondegenerate and noncatastrophic $\not\Rightarrow$ locally invertible. This is easily demonstrated with the following example.

Consider a rate $\frac{1}{3}$ 1-D convolutional encoder $G(z_1) = [z_1^6 \quad z_1^4 \quad 1+z_1+z_1^3+z_1^5+z_1^6]$ with memory order $M_1 = 6$. The encoder $G(z_1)$ is basic because it has a polynomial inverse $G(z_1)^{-1} = [z_1^2 \quad 1 \quad 1+z_1+z_1^2]^T$. It can easily be verified that $G(z_1)$ is both nondegenerate and noncatastrophic because it satisfies Definition 5.5.1 and Theorem 5.5.1. Since $w = \frac{nkM_1}{n-k} = 9$ is a positive integer, $G(z_1)$ has a subsequence map of length 9. But this subsequence map is not invertible because the 9×9 reduced encoding matrix $\hat{G}$ shown below is singular.

$$\hat{G} = \begin{bmatrix} 1 & 0 & 1 & & & & & & \\ 0 & 0 & 1 & 1 & 0 & 1 & & & \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ & & & 0 & 0 & 1 & 0 & 0 & 1 \\ & & & & & & 0 & 0 & 1 \end{bmatrix}$$

Since the columns of $\hat{G}$ are shifts of the generator sequence (in reverse form), $g(z_1)^{(1)} = z_1^2 g(z_1)^{(2)}$ causes columns 2 and 7 of $\hat{G}$ to be identical. As a result, $\hat{G}$ in singular. Hence, $G(z_1)$ is not locally invertible.

Let ν be the overall constraint length (3.12) of a rate $\frac{k}{n}$ 1-D convolutional encoder with memory order M_1. In [13], a rate $\frac{k}{n}$ 1-D convolutional encoder is called locally invertible if all the nonzero rows in $\mathbf{G}_{[M_1, M_1+\nu]}$ are linearly independent, where $\mathbf{G}_{[M_1, M_1+\nu]}$ is a finite section of the semi-infinite generator matrix $\mathbf{G}$. Once again, the matrix $\mathbf{G}_{[M_1, M_1+\nu]}$ defined above is not the same as the $w \times w$ reduced encoding matrix $\hat{G}$ defined in (5.19) and illustrated in (5.52). In particular, we require the parameters k, n, M_1 to be such that $w = \frac{nkM_1}{n-k}$ is a positive integer in order for a subsequence map to exist. As observed in Condition 5.3.1, it is also necessary that $\nu = kM_1$. That is, the constraint lengths ν_x, $x = 1, \ldots, k$ of every row of the encoder must be equal. Otherwise, $\hat{G}$ will have a zero row. Once again, due to the difference in definitions, some of the results reported in [13] do not apply to the locally invertible encoders defined in this dissertation. In [13] it is has been stated that *a rate $\frac{k}{n}$ 1-D convolutional encoder is minimal-basic if and only if it is locally invertible*. We have already shown in Theorem 5.4.7 that a 1-D locally invertible encoder is minimal-basic. But the reverse implication does not hold. That is, 1-D minimal-basic encoder $\not\Rightarrow$ locally invertible.

Consider the following rate $\frac{2}{3}$ 1-D convolutional encoder with constraint lengths $\nu_1 = 1, \nu_2 = 2$ and memory order $M_1 = 2$.

$$G(z_1) = \begin{bmatrix} 1 + z_1 & z_1 & 1 \\ z_1^2 & 1 & 1 + z_1 + z_1^2 \end{bmatrix}$$

The full size minors of $G(z_1)$ are

$$\{1 + z_1 + z_1^3,\ \ 1 + z_1 + z_1^2 + z_1^3,\ \ 1 + z_1^2 + z_1^3\}.$$

Clearly, $G(z_1)$ is MP. Since ZP≡MP for $m = 1$, $G(z_1)$ is basic. By Theorem 3.6.2, $G(z_1)$ is also minimal-basic because the maximum degree $\mu = 3$ of its full size minors is equal to its overall constraint length $\nu = \nu_1 + \nu_2 = 3$. Since $w = \frac{nkM_1}{n-k} = 12$ is a positive integer, $G(z_1)$ has a subsequence map of length 12. But this subsequence map is not invertible because

the 12×12 reduced encoding matrix $\hat{G}$ shown below is singular.

$$\hat{G} = \begin{bmatrix}
0 & 0 & 0 & & & & & & & & & \\
1 & 0 & 1 & & & & & & & & & \\
1 & 1 & 0 & 0 & 0 & 0 & & & & & & \\
0 & 0 & 1 & 1 & 0 & 1 & & & & & & \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & & & \\
0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & & & \\
 & & & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
 & & & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\
 & & & & & & 1 & 0 & 1 & 1 & 1 & 0 \\
 & & & & & & 0 & 1 & 1 & 0 & 0 & 1 \\
 & & & & & & & & & 1 & 0 & 1 \\
 & & & & & & & & & 0 & 1 & 1
\end{bmatrix}$$

Since the columns of $\hat{G}$ are shifts of the generator sequence (in reverse composite form), $\nu_1 < \nu_2$ causes the first row of $\hat{G}$ to be an all zero row. Hence $G(z_1)$ is not locally invertible.

Chapter 6

Syndrome Formers

6.1 Error Syndromes via Subsequence Overlap

In Section 5.2 we saw that the invertible subsequence map of a locally invertible encoder could be used to recover an input sequence from its corresponding codeword. In this section we show that the overlapping input subsequences obtained during such a sequence inversion technique can be used to detect errors.

Let $G(z) \in R^{k\times n}$ be a locally invertible encoder with an invertible subsequence map of size $\prod_{i=1}^{m} w_i$. Let $v = u * G \in \mathcal{C}$ be a codeword corresponding to an input sequence $u \in S^k$. The codeword v, when transmitted over a channel is assumed to be corrupted by noise. Let $r = v + e \in S^n$ be the sequence received at the output of the channel corresponding to the codeword v and the error sequence $e \in S^n$. Let e be such that it does not convert v into another codeword $\grave{v} \in \mathcal{C}$. That is $r = e + v \neq \grave{v}$. If e converts v to $\grave{v}$, then e is an undetectable error sequence and is beyond the error detecting capability of the code. The received sequence r is viewed as being made up of overlapping subsequences $\hat{r}$ of size $\prod_{i=1}^{m} w_i$ that are shifted by n_i symbols in the ith dimension of S^n. In this case, there is an overlap of $w_i - n_i$ symbols between consecutive received subsequences in the ith dimension. The inverse subsequence map (5.9) is used to invert each $\hat{r}$ into an estimate $\tilde{u}$ of the input subsequence $\hat{u} \in S^k$. This operation can be represented using the inverse reduced encoding matrix as shown below.

$$\tilde{u} = \mathcal{U}_f^{-1}(\mathcal{U}_f(\hat{r})\hat{G}^{-1}) \tag{6.1}$$

The inverted subsequences $\tilde{u}$ are put together piecemeal with a shift of k_i symbols and an

overlap of $w_i - k_i$ symbols in S^k to produce an estimate of the input sequence $u \in S^k$. During this inversion process, let $\hat{r}_1$ and $\hat{r}_2$ be two shifted subsequences such that their overlapping symbols have an error. In this case, the overlapping symbols of the corresponding estimates $\tilde{u}_1$ and $\tilde{u}_2$ will not be the same.

$$\tilde{u}_1 = \mathcal{U}_{fr}^{-1}(\mathcal{U}_{fr}(\hat{e}_1 + \hat{v}_1)\hat{G}^{-1}) \tag{6.2}$$

$$\tilde{u}_2 = \mathcal{U}_{fr}^{-1}(\mathcal{U}_{fr}(\hat{e}_2 + \hat{v}_2)\hat{G}^{-1}) \tag{6.3}$$

Since $\hat{r}_1$ and $\hat{r}_2$ are shifted subsequences, the overlapping symbols of $\tilde{u}_1$ and $\tilde{u}_2$ are obtained from different columns of the reduced encoding matrix $\hat{G}^{-1}$. Therefore, if $\hat{e}_1$ and $\hat{e}_2$ in (6.2) and (6.2) are nonzero subsequences, the overlapping symbols of $\tilde{u}_1$ and $\tilde{u}_2$ will be different. This implies that the received sequence r is not the same as the transmitted sequence v.

For any subsequence $\tilde{u} \in S^k$, there are a total of $\frac{w_i - k_i}{k_i}$ consecutive subsequences that have overlapping symbols with $\tilde{u}$ in the ith dimension of S^k. There are a total of $\frac{w-k}{k}$ such subsequences that in the m-D space S^k. If $\tilde{u}$ is an estimate obtained by inverting a subsequence $\hat{r} \in S^n$ that has an error, then the overlapping symbols of each of the above $\frac{w-k}{k}$ estimates will be different. A difference in overlapping symbols only indicates the presence of an error and not the exact error sequence. However, the difference in symbols between consecutive overlapping subsequences can be used to generate an error syndrome. In the case of 1-D codes generated by 1-D locally invertible encoders, the Table-Driven decoder [6] performs error correction based on this technique. A similar approach can be extended for error correction of m-D codes generated by m-D locally invertible encoders.

6.2 Pseudo Inverses

In Section 6.1 we saw that the presence of an error in the received sequence results in a difference in overlapping symbols between successive input subsequences. In this section, we divert our attention to the multivariate polynomial ring R to derive an algebraic interpretation of the difference in overlapping symbols using the encoder inverse and pseudo-inverses obtained from the inverse of the reduced encoding matrix.

Recall from Proposition 5.4.2 and equation (5.41) that the columns of the inverse reduced encoding matrix $\hat{G}^{-1}$ of a locally invertible encoder $G(z) \in R^{k \times n}$ can be used to obtain an encoder inverse $G(z)^{-1} \in R^{n \times k}$ such that

$$G(z)G(z)^{-1} = I_{k \times k}, \tag{6.4}$$

and a set of $\frac{w-k}{k}$ unique pseudo-inverses $G(z)^{-1}_{(d_1,\ldots,d_m)} \in R^{n\times k}$ that satisfy the condition

$$G(z)G(z)^{-1}_{(d_1,\ldots,d_m)} = z_1^{d_1}\cdots z_m^{d_m} I_{k\times k}, \tag{6.5}$$

with a delay d_i in the i^{th} dimension that ranges from $0 \le d_i \le \frac{w_i-k_i}{k_i}$ and $\sum_{i=1}^{m} d_i \neq 0$.

The following result is needed for demonstrating the use of these pseudo-inverses for error detection.

Proposition 6.2.1. *Let $G(z)^{-1}$ and $G(z)^{-1}_{(d_1,\ldots,d_m)}$ be as defined above. Then,*

$$G(z)^{-1}_{(d_1,\ldots,d_m)} \neq z_1^{d_1}\cdots z_m^{d_m} G(z)^{-1} \quad \textit{for all} \quad d_i > \frac{w_i - n_i}{n_i}. \tag{6.6}$$

Proof. Let M_i^{-1} and $M^{-1}_{i,(d_1,\ldots,d_m)}$ denote the memory orders of $G(z)^{-1}$ and $G(z)^{-1}_{(d_1,\ldots,d_m)}$ in the ith dimension. Each column of the $w \times w$ inverse reduced encoding matrix $\hat{G}^{-1}$ is an m-D subsequence of size $\prod_{i=1}^{m} w_i$ with order $\prod_{i=1}^{m} n_i$ unfolded as 1-D sequence. Therefore, the largest exponent of the z_ith variable in $G(z)^{-1}$ and $G(z)^{-1}_{(d_1,\ldots,d_m)}$ can be at most $\frac{w_i-n_i}{n_i}$. Hence, $M_i^{-1}, M^{-1}_{i,(d_1,\ldots,d_m)} \le \frac{w_i-n_i}{n_i}$.

From (6.5), the values of the delays d_i range from $0 \le d_i \le \frac{w_i-k_i}{k_i}$. Since w_i is a positive integer, we have $n_i > k_i$. Hence, $\frac{w_i-k_i}{k_i} > \frac{w_i-n_i}{n_i}$.

Assume in the worst case that all entries in $G(z)^{-1}$ are constants. That is, the memory orders of $G(z)^{-1}$ are zero in every dimension. Now, contrary to the statement of the proposition, if

$$G(z)^{-1}_{(d_1,\ldots,d_m)} = z_1^{d_1}\cdots z_m^{d_m} G(z)^{-1},$$

then any $G(z)^{-1}_{(d_1,\ldots,d_m)}$ with delay $d_i > \frac{w_i-n_i}{n_i}$ will have $M^{-1}_{i,(d_1,\ldots,d_m)} > \frac{w_i-n_i}{n_i}$. □

The property defined in Proposition 6.2.1 can be used to detect errors as follows: Let $\mathcal{C}$ be the code generated by a locally invertible encoder $G(z) \in R^{k\times n}$. Let $v(z) = u(z)G(z) \in \mathcal{C}$ be the codeword corresponding to an input data polynomial vector $u(z) \in R^k$. From equations (6.4) and (6.5) we have

$$u(z) = v(z)G(z)^{-1} \tag{6.7}$$

$$\begin{aligned} u(z)_{(d_1,\ldots,d_m)} &= v(z)G(z)^{-1}_{(d_1,\ldots,d_m)} \\ &= z_1^{d_1}\cdots z_m^{d_m} u(z). \end{aligned} \tag{6.8}$$

Let $r(z) = (e(z) + v(z)) \in R^n$ be the polynomial vector received at the output of a noisy channel, corresponding to an error polynomial vector $e(z) \in R^n$. Let $\tilde{u}(z)$ be an estimate

of the input based on $r(z)$.

$$\begin{aligned}\tilde{u}(z) &= r(z)G(z)^{-1} \\ &= (e(z)+v(z))G(z)^{-1} \\ &= e(z)G(z)^{-1} + u(z)\end{aligned} \tag{6.9}$$

Let $\tilde{u}(z)_{(d_1,\ldots,d_m)}$ be the $\frac{w-k}{k}$ estimates of the input polynomial obtained from each of the pseudo-inverses.

$$\begin{aligned}\tilde{u}(z)_{(d_1,\ldots,d_m)} &= r(z)G(z)^{-1}_{(d_1,\ldots,d_m)} \\ &= (e(z)+v(z))G(z)^{-1}_{(d_1,\ldots,d_m)} \\ &= e(z)G(z)^{-1}_{(d_1,\ldots,d_m)} + z_1^{d_1}\cdots z_m^{d_m}u(z)\end{aligned} \tag{6.10}$$

From equations (6.9) and (6.10) and Proposition 6.2.1, for values of $d_i > \frac{w_i-n_i}{n_i}$ we have

$$\tilde{u}(z)_{(d_1,\ldots,d_m)} = z_1^{d_1}\cdots z_m^{d_m}\tilde{u}(z) \quad \text{if and only if} \quad r(z)\in\mathcal{C}. \tag{6.11}$$

Notice that this condition holds only because of Proposition 6.2.1. Otherwise, the estimates $\tilde{u}(z)_{(d_1,\ldots,d_m)}$ would be just be a delayed versions of $\tilde{u}(z)$ even if $r(z)\notin\mathcal{C}$.

Example 6.2.1. Consider the 2-D locally invertible encoder $G(z)\in R^{2\times 6}$ from Example 4.1.1. The codeword $v(z)=u(z)G(z)$ (from equation (4.8)) corresponding to an input data polynomial vector $u(z)=[1+z_1z_2 \quad z_2^2]$ is shown below.

$$\begin{aligned}v(z) = \big[&z_1^2z_2+z_1^3z_2^2 \quad z_2^2+z_1^2z_2^3 \quad z_2+z_1z_2^2 \quad z_1^2+z_1^3z_2+z_1z_2^3 \\ &1+z_1z_2+z_1z_2^2 \quad z_2^2+z_1^2z_2^2+z_2^3\big]\end{aligned}$$

Consider the encoder inverse $G(z)^{-1}$ and the $\frac{w-k}{k}=11$ pseudo-inverses $G(z)^{-1}_{(d_1,d_2)}$ obtained from the columns of the 24×24 inverse reduced encoding matrix $\hat{G}^{-1}$ shown in Figure 5.8.

When the received sequence $r(z)$ is error free (i.e. $r(z)=v(z)$), the estimate of $u(z)$ obtained from the encoder inverse $G(z)^{-1}$ in Figure 5.8 is

$$\tilde{u}(z) = r(z)G(z)^{-1} = [1+z_1z_2 \quad z_2^2].$$

The estimates of the input $u(z)$ obtained from the pseudo-inverses in Figure 5.8 are shown below.

$$\begin{aligned}
\tilde{u}(z)_{(1,0)} &= r(z)G(z)^{-1}_{(1,0)} = [z_1 + z_1^2 z_2 \quad z_1 z_2^2] = z_1\tilde{u}(z)\\
\tilde{u}(z)_{(2,0)} &= r(z)G(z)^{-1}_{(2,0)} = [z_1^2 + z_1^3 z_2 \quad z_1^2 z_2^2] = z_1^2\tilde{u}(z)\\
\tilde{u}(z)_{(3,0)} &= r(z)G(z)^{-1}_{(3,0)} = [z_1^3 + z_1^4 z_2 \quad z_1^3 z_2^2] = z_1^3\tilde{u}(z)\\
\tilde{u}(z)_{(0,1)} &= r(z)G(z)^{-1}_{(0,1)} = [z_2 + z_1 z_2^2 \quad z_2^3] = z_2\tilde{u}(z)\\
\tilde{u}(z)_{(1,1)} &= r(z)G(z)^{-1}_{(1,1)} = [z_1 z_2 + z_1^2 z_2^2 \quad z_1 z_2^3] = z_1 z_2\tilde{u}(z)\\
\tilde{u}(z)_{(2,1)} &= r(z)G(z)^{-1}_{(2,1)} = [z_1^2 z_2 + z_1^3 z_2^2 \quad z_1^2 z_2^3] = z_1^2 z_2\tilde{u}(z)\\
\tilde{u}(z)_{(3,1)} &= r(z)G(z)^{-1}_{(3,1)} = [z_1^3 z_2 + z_1^4 z_2^2 \quad z_1^3 z_2^3] = z_1^3 z_2\tilde{u}(z)\\
\tilde{u}(z)_{(0,2)} &= r(z)G(z)^{-1}_{(0,2)} = [z_2^2 + z_1 z_2^3 \quad z_2^4] = z_2^2\tilde{u}(z)\\
\tilde{u}(z)_{(1,2)} &= r(z)G(z)^{-1}_{(1,2)} = [z_1 z_2^2 + z_1^2 z_2^3 \quad z_1 z_2^4] = z_1 z_2^2\tilde{u}(z)\\
\tilde{u}(z)_{(2,2)} &= r(z)G(z)^{-1}_{(2,2} = [z_1^2 z_2^2 + z_1^3 z_2^3 \quad z_1^2 z_2^4] = z_1^2 z_2^2\tilde{u}(z)\\
\tilde{u}(z)_{(3,2)} &= r(z)G(z)^{-1}_{(3,2)} = [z_1^3 z_2^2 + z_1^4 z_2^3 \quad z_1^3 z_2^4] = z_1^3 z_2^2\tilde{u}(z)
\end{aligned}$$

Since $\tilde{u}(z)_{(d_1,d_2)} = z_1^{d_1} z_2^{d_2} u(z);\ \forall d_1, d_2$, from (6.11) we conclude that there is no error in the received vector $r(z)$.

Now, consider an error polynomial vector $e(z) = [0 \quad 0 \quad z_1 z_2^2 \quad 0 \quad 0 \quad 0]$, such that the received polynomial vector $r(z) = e(z) + v(z)$ is

$$r(z) = \left[z_1^2 z_2 + z_1^3 z_2^2 \quad z_2^2 + z_1^2 z_2^3 \quad z_2 \quad z_1^2 + z_1^3 z_2 + z_1 z_2^3 \quad 1 + z_1 z_2 + z_1 z_2^2 \quad z_2^2 + z_1^2 z_2^2 + z_2^3\right].$$

The estimate of $u(z)$ obtained from the encoder inverse $G(z)^{-1}$ is

$$\tilde{u}(z) = r(z)G(z)^{-1} = [1 + z_1 z_2 + z_1 z_2^2 \quad z_2^2 + z_1^2 z_2^2].$$

The estimates of the input $u(z)$ obtained from the pseudo-inverses are shown below.

$$\begin{aligned}
\tilde{u}(z)_{(1,0)} &= r(z)G(z)^{-1}_{(1,0)} = [z_1 + z_1^2 z_2 + z_1^2 z_2^2 \quad 0] \neq z_1\tilde{u}(z)\\
\tilde{u}(z)_{(2,0)} &= r(z)G(z)^{-1}_{(2,0)} = [z_1^2 + z_1^3 z_2 + z_1 z_2^2 \quad 0] \neq z_1^2\tilde{u}(z)\\
\tilde{u}(z)_{(3,0)} &= r(z)G(z)^{-1}_{(3,0)} = [z_1^3 + z_1^4 z_2 + z_1^2 z_2^2 \quad z_1^3 z_2^2] \neq z_1^3\tilde{u}(z)\\
\tilde{u}(z)_{(0,1)} &= r(z)G(z)^{-1}_{(0,1)} = [z_2 \quad z_2^3] \neq z_2\tilde{u}(z)\\
\tilde{u}(z)_{(1,1)} &= r(z)G(z)^{-1}_{(1,1)} = [z_1 z_2 \quad z_1 z_2^2 + z_1 z_2^3] \neq z_1 z_2\tilde{u}(z)\\
\tilde{u}(z)_{(2,1)} &= r(z)G(z)^{-1}_{(2,1)} = [z_1^2 z_2 + z_1^3 z_2^2 \quad z_1^2 z_2^2 + z_1^2 z_2^3] \neq z_1^2 z_2\tilde{u}(z)\\
\tilde{u}(z)_{(3,1)} &= r(z)G(z)^{-1}_{(3,1)} = [z_1^3 z_2 + z_1^4 z_2^2 \quad z_1 z_2^2 + z_1^3 z_2^3] \neq z_1^3 z_2\tilde{u}(z)\\
\tilde{u}(z)_{(0,2)} &= r(z)G(z)^{-1}_{(0,2)} = [z_2^2 \quad z_1^2 z_2^2 + z_2^4] \neq z_2^2\tilde{u}(z)\\
\tilde{u}(z)_{(1,2)} &= r(z)G(z)^{-1}_{(1,2)} = [z_1 z_2^2 \quad z_1 z_2^4] \neq z_1 z_2^2\tilde{u}(z)
\end{aligned}$$

$$\begin{aligned}\tilde{u}(z)_{(2,2)} &= r(z)G(z)^{-1}_{(2,2} &= [z_1^2z_2^2 + z_1^3z_2^3 \quad z_1^2z_2^4] &\neq z_1^2z_2^2\tilde{u}(z)\\ \tilde{u}(z)_{(3,2)} &= r(z)G(z)^{-1}_{(3,2)} &= [z_1^3z_2^2 + z_1^4z_2^3 \quad z_1^3z_2^4 + z_1z_2^2] &\neq z_1^3z_2^2\tilde{u}(z)\end{aligned}$$

In order to conclude that that the received vector has an error, it is sufficient that any one of the estimates violates condition (6.11). In this case $\tilde{u}(z)_{(d_1,d_2)} \neq z_1^{d_1}z_2^{d_2}u(z)\ \forall d_1, d_2$ and so $r(z) \notin \mathcal{C}$.

6.3 Parity Check Matrices

Let $\mathcal{C} = \text{im}(G(z))$ be an m-D convolutional code of rate $\frac{k}{n}$ generated by a ZP encoder $G(z) \in R^{k\times n}$. We know from Corollary 3.5.9 that $\mathcal{C}^\perp = \ker(G(z)^T)$ is the rate $\frac{n-k}{n}$ dual code of $\mathcal{C}$. Since $\mathcal{C}^\perp$ has rank $n-k$, there exists a matrix $H(z) \in R^{(n-k)\times n}$ of full-row rank

$$H(z) = \begin{bmatrix} h(z)_1^{(1)}, & \dots, & h(z)_1^{(n)} \\ & \vdots & \\ h(z)_{n-k}^{(1)}, & \dots, & h(z)_{n-k}^{(n)} \end{bmatrix}, \tag{6.12}$$

with elements $h(z)_x^{(y)} \in R$ such that $\mathcal{C}^\perp = \text{im}(H(z))$ and $\mathcal{C} = \ker(H(z)^T)$. That is, $H(z)$ is an encoder for $\mathcal{C}^\perp$ and a parity-check matrix for $\mathcal{C}$. In this section we show that if the code $\mathcal{C}$ is generated by a locally invertible encoder, then the encoder $H(z)$ of the dual code $\mathcal{C}^\perp$ can be easily computed by performing elementary operations on the ground field on a finite set of subsequences obtained from codewords in $\mathcal{C}$.

Let $\mathcal{C}$ be an m-D convolution code generated by a locally invertible encoder $G(z) \in R^{k\times n}$. Let $G = \psi^{-1}(G(z)) \in S^{k\times n}$ be the equivalent sequence encoder of $G(z)$. We know that the $\prod_{i=1}^m w_i$ sized invertible subsequence map of the locally invertible encoder is completely determined by the $w \times w$ reduced encoding matrix $\hat{G}$. This is because the rows of $\hat{G}$ are obtained from a subsequence map (5.3) that is generated using a standard basis as input. Therefore, finding the null space of $\prod_{i=1}^m w_i$ sized subsequences obtained from codewords in $\mathcal{C}$, is the same as solving for the null space of $\hat{G}$. But since $\hat{G}$ is nonsingular, the only solution to $\hat{G}h = 0$ is the trivial solution $h = 0$.

The subsequence mapping length w_i defined in (5.2) is the number of input symbols in S^k that map to an equal number of codeword symbols in $\mathcal{C}$ along the ith dimension of their respective sequence spaces. So, $w_i + k_i$ input symbols in S^k will map to $w_i + n_i$ codeword symbols in $\mathcal{C}$ along the ith dimension. Now, consider such an extended subsequence map

generated by a standard basis of size $\prod_{i=1}^{m}(w_i + k_i)$ used as input during the encoding operation $v = u * G$.

$$\begin{array}{ccc} S^k & \rightarrow & S^n \\ b_r & \mapsto & \hat{v}_r \end{array} \tag{6.13}$$

Let $t = \prod_{i=1}^{m}(w_i + k_i)$. Here, for $r = 1, \ldots, t$ each b_r of size $\prod_{i=1}^{m}(w_i + k_i)$ and order $\prod_{i=1}^{m} k_i$ maps to an output subsequence $\hat{v}_r$ of size $\prod_{i=1}^{m}(w_i + n_i)$ and order $\prod_{i=1}^{m} n_i$.

A set of parity-check subsequences that are orthogonal to the above map can be found as follows: Construct a matrix V as shown below by taking each $\hat{v}_r$ and unfolding it into a 1-D sequence to form a row.

$$\mathcal{U}_{fr}(\hat{v}_r) = \bar{v}_r \tag{6.14}$$

Without loss of generality we assume row-major unfolding (5.17) is used.

$$V = [\bar{v}_1, \ldots, \bar{v}_t]^T \tag{6.15}$$

The resulting matrix V has $\prod_{i=1}^{m}(w_i + k_i)$ rows and $\prod_{i=1}^{m}(w_i + n_i)$ columns.

Proposition 6.3.1. *The matrix V has full row rank.*

Proof. Assume the statement is false. This is equivalent to saying that the subsequence map (6.13) is not injective. Then, there exist a set of $\alpha_r \in \mathbb{F}$, not all zero, such that a nonzero input subsequence $\sum_{r=1}^{t} \alpha_r b_r \in S^k$ of size $\prod_{i=1}^{m}(w_i + k_i)$ maps to an all-zero subsequence $\sum_{r=1}^{t} \alpha_r \hat{v}_r = \mathbf{0} \in S^n$ of size $\prod_{i=1}^{m}(w_i + n_i)$. This in turn implies that $\sum_{r=1}^{t} \alpha_r b_r$ has a nonzero subsequence of size $\prod_{i=1}^{m} w_i$ that maps to an all-zero subsequence of size $\prod_{i=1}^{m} w_i$ of $\sum_{r=1}^{t} \alpha_r \hat{v}_r = \mathbf{0}$. But, this is a contradiction to the fact that a locally invertible encoder has an invertible subsequence map of size $\prod_{i=1}^{m} w_i$. Therefore, (6.13) is an injective map, and so V has full row rank. □

By Proposition 6.3.1, the nullity of V is

$$n_h = \prod_{i=1}^{m}(w_i + n_i) - \prod_{i=1}^{m}(w_i + k_i). \tag{6.16}$$

Let $\bar{M}_g$ be the set of n_h 1-D composite sequences $\bar{h}_j$, each of length $\prod_{i=1}^{m}(w_i + n_i)$, obtained by solving for the null space of V.

$$\bar{M}_g = \left\{\bar{h}_1, \ldots, \bar{h}_{n_h}\right\} \tag{6.17}$$

Let $p = \prod_{i=1}^{m}(w_i + n_i)$. Since $\bar{h}_j$ is in the null space of V, the vector dot product

$$\left\langle \bar{v}_r, \bar{h}_j \right\rangle = \sum_{i=0}^{p-1} \bar{v}_{r,(i)} \bar{h}_{j,(i)} = 0, \tag{6.18}$$

for all $\bar{v}_r$; $r = 1, \ldots, t$ in (6.15), and $\bar{h}_j$; $j = 1, \ldots, n_h$ in (6.17).

Next, we fold each 1-D sequence $\bar{h}_j$ of length $1 \times p$ into an m-D sequence $\hat{h}_j \in S^n$ of size $\prod_{i=1}^{m}(w_i + n_i)$ and order $\prod_{i=1}^{m} n_i$ using

$$\mathcal{U}_{fr}^{-1}(\bar{h}_j) = \hat{h}_j. \tag{6.19}$$

Let M_g represent this set of m-D composite sequences.

$$M_g = \left\{ \hat{h}_1, \ldots, \hat{h}_{n_h} \right\} \tag{6.20}$$

From (6.18) above, the m-D dot product

$$\left\langle \hat{v}_r, \hat{h}_j \right\rangle = \sum_{l_m=0}^{w_m+n_m-1} \cdots \sum_{l_1=0}^{w_1+n_1-1} \hat{v}_{r,(l_1,\ldots,l_m)} \hat{h}_{j,(l_1,\ldots,l_m)} = 0, \tag{6.21}$$

for all $\hat{v}_r$; $r = 1, \ldots, t$ in (6.13), and $\hat{h}_j$; $j = 1, \ldots, n_h$ in (6.20). This follows from the fact that the matrix V is a 1-D representation of the m-D output subsequences $\hat{v}_r \in S^n$, and so the null space of V is same as the null space of the output subsequences in (6.13).

Let $\grave{v} \in S^n$ be a $\prod_{i=1}^{m}(w_i + n_i)$ sized subsequence of any codeword $v \in \mathcal{C}$. Since the subsequence map (6.13) is generated using a standard basis as input, we can write

$$\grave{v} = \sum_{r=1}^{t} \alpha_r \hat{v}_r, \qquad \alpha_r \in \mathbb{F}. \tag{6.22}$$

From (6.21) and (6.22), it follows that

$$\left\langle \grave{v}, \hat{h}_j \right\rangle = \sum_{r=1}^{t} \alpha_r \left\langle \hat{v}_r, \hat{h}_j \right\rangle = 0. \tag{6.23}$$

Any codeword $v \in \mathcal{C}$ can be considered to be made up of overlapping subsequences of size $\prod_{i=1}^{m}(w_i+n_i)$ that are shifted by n_i symbols in the ith dimension. From (6.23), the m-D dot product of $\hat{h}_j$, $j = 1, \ldots, n_h$ with each of these overlapping subsequences is zero. Therefore,

$$v * \mathcal{R}(\hat{h}_j) = \mathbf{0} \qquad \text{for all} \quad v \in \mathcal{C}. \tag{6.24}$$

Here, $\mathbf{0} \in S^n$ is an all-zero m-D composite sequence, and $*$ denotes m-D convolution of the composite codeword $v := (v^{(1)}, \ldots, v^{(n)})$ with the reverse composite form (see Definition 5.3.1) of the composite sequence $\hat{h}_j := (\hat{h}_j^{(1)}, \ldots, \hat{h}_j^{(n)})$.

Clearly, (6.24) implies that $\mathcal{R}(\hat{h}_j)$ is orthogonal to all $v \in \mathcal{C}$, and so $\mathcal{R}(\hat{h}_j) \in \mathcal{C}^\perp$. Since a locally invertible encoder is ZP, from Corollary 3.5.9 it follows that the orthogonal module $\mathcal{C}^\perp$ is the dual code of $\mathcal{C}$, and $\text{rank}(\mathcal{C}^\perp) = n - k$. From (6.16), we have $n_h \geq n - k$. Therefore, the reverse composite form of the set M_g (6.20)

$$\mathcal{R}(M_g) = \left\{ \mathcal{R}(\hat{h}_1), \ldots, \mathcal{R}(\hat{h}_{n_h}) \right\}, \tag{6.25}$$

is a *spanning set* for the dual code $\mathcal{C}^\perp$ and the *basis* of $\mathcal{C}^\perp$ is contained in $\mathcal{R}(M_g)$.

Remark 6.3.1. In the special case of 1-D convolutional codes generated by locally invertible encoders, we have $w = w_1, n = n_1$ and $k = k_1$. So $n_h = (w_1 + n_1) - (w_1 + k_1) = n - k$ implies that the set $\mathcal{R}(M_g)$ is the basis of the dual code $\mathcal{C}^\perp$.

For $m > 1$, the solution to the null space of the output subsequences in (6.13) gives us n_h m-D composite sequences $\hat{h}_j \in S^n$ that are $\mathbb{F}$-linearly independent. But this does not ensure R-linear independence (i.e. linear independence with respect to shifts. See for example, Definition 2.0.6 and equation (2.12)). So, it is possible that

$$\hat{h}_j = \sum_{\substack{i=1 \\ i \neq j}}^{n_h} \omega_i * \hat{h}_i; \quad \hat{h}_i, \hat{h}_j \in M_g, \;\; \omega_i \in S. \tag{6.26}$$

Since $\text{rank}(\mathcal{C}^\perp) = n - k$, only $n - k$ sequences in the spanning set $\mathcal{R}(M_g)$ (out of the total n_h) are R-linearly independent and form a basis of $\mathcal{C}^\perp$. These basis elements represent the composite generator sequences of the dual code $\mathcal{C}^\perp$.

$$H_c = \left\{ \mathcal{R}(\hat{h}_1), \ldots, \mathcal{R}(\hat{h}_{n-k}) \right\} = \{h_1, \ldots, h_{n-k}\} \tag{6.27}$$

Each composite generator sequence $h_x \in S^n$; $x = 1, \ldots, n - k$ has to be deinterleaved n_i symbols at a time along the ith dimension to obtain the xth row $(h_x^{(1)}, \ldots, h_x^{(n)})$ of the sequence encoder $H \in S^{(n-k)\times n}$ of the dual code $\mathcal{C}^\perp$.

$$H = \begin{pmatrix} h_1^{(1)}, & \ldots, & h_1^{(n)} \\ & \vdots & \\ h_{n-k}^{(1)}, & \ldots, & h_{n-k}^{(n)} \end{pmatrix} \tag{6.28}$$

The polynomial representation of the sequence encoder $H \in S^{(n-k)\times n}$ using the transformation $h(z)_x^{(y)} = \psi(h_x^{(y)})$ gives us the polynomial encoder $H(z) \in R^{(n-k)\times n}$ (6.12) of the dual code.

Since the composite sequences h_x in (6.27) are of size $\prod_{i=1}^{m}(w_i + n_i)$ with order $\prod_{i=1}^{m} n_i$, the length of h_x along the ith dimension is

$$w_i + n_i = n_i(M_i^{\perp} + 1). \tag{6.29}$$

Substituting values from (5.2) we get

$$M_i^{\perp} = \frac{k_i M_i}{n_i - k_i}, \tag{6.30}$$

where $M_i^{\perp}$ is the memory order of $H(z)$ along the ith dimension.

Example 6.3.1. Consider the 2-D locally invertible encoder $G(z) \in R^{2\times 6}$ (4.7) from Example 4.1.1 with memory orders $M_1 = 2$ and $M_2 = 1$. If we factor k, n as 1×2 and 2×3 to predefine a sequence ordering of 1×2 in S^2 and 2×3 in S^6, then subsequence mapping lengths along i_1 and i_2 will be $w_1 = 4$ and $w_2 = 6$. The extended subsequence map (6.13) generated by standard basis subsequences of size $\prod_{i=1}^{2}(w_i + k_i) = 5 \times 8$ results in output subsequences of size $\prod_{i=1}^{2}(w_i + n_i) = 6 \times 9$ as shown below.

$$\begin{array}{ccccccc}
b_1 & & \hat{v}_1 & & b_{40} & & \hat{v}_{40} \\
\begin{smallmatrix} 1&0&0&0&0 \\ 0&0&0&0&0 \end{smallmatrix} & & \begin{smallmatrix} 10&00&00 \\ 00&00&00 \\ 00&00&00 \end{smallmatrix} & & \begin{smallmatrix} 0&0&0&0&0 \\ 0&0&0&0&0 \end{smallmatrix} & & \begin{smallmatrix} 00&00&00 \\ 00&00&00 \\ 00&00&00 \end{smallmatrix} \\
\begin{smallmatrix} 0&0&0&0&0 \\ 0&0&0&0&0 \end{smallmatrix} & & \begin{smallmatrix} 00&00&00 \\ 00&00&00 \\ 00&00&00 \end{smallmatrix} & & \begin{smallmatrix} 0&0&0&0&0 \\ 0&0&0&0&0 \end{smallmatrix} & & \begin{smallmatrix} 00&00&00 \\ 00&00&00 \\ 00&00&00 \end{smallmatrix} \\
\begin{smallmatrix} 0&0&0&0&0 \\ 0&0&0&0&0 \end{smallmatrix} & \mapsto & \begin{smallmatrix} 00&00&00 \\ 00&00&00 \\ 00&00&00 \end{smallmatrix} & , \cdots\cdots , & \begin{smallmatrix} 0&0&0&0&0 \\ 0&0&0&0&0 \end{smallmatrix} & \mapsto & \begin{smallmatrix} 00&00&00 \\ 00&00&00 \\ 00&00&00 \end{smallmatrix} \\
\begin{smallmatrix} 0&0&0&0&0 \\ 0&0&0&0&0 \end{smallmatrix} & & \begin{smallmatrix} 00&00&00 \\ 00&00&00 \\ 00&00&00 \end{smallmatrix} & & \begin{smallmatrix} 0&0&0&0&0 \\ 0&0&0&0&0 \end{smallmatrix} & & \begin{smallmatrix} 00&00&01 \\ 00&00&00 \\ 00&00&01 \end{smallmatrix}
\end{array} \tag{6.31}$$

Each 2-D output subsequence $\hat{v}_r$ is unfolded into a 1-D sequence $\mathcal{U}_{fr}(\hat{v}_r) = \bar{v}_r$ of length 54 to form a row of the 40×54 matrix V shown in Figure 6.1. The blanks in the matrix denote zeros. We note here that the matrix V has a structure similar to that of the reduced encoding matrix $\hat{G}$ in Figure 5.4, and can therefore be formed by inspection using the sequence generator G, without actually generating the extended subsequence map shown in (6.31).

A solution to the null space of V results in $n_h = 54 - 40 = 14$, $\mathbb{F}$-linearly independent 1-D sequences $\bar{h}_j$ of length 54. Each $\bar{h}_j$ is folded into a composite 2-D sequence

Figure 6.1: 1-D representation of the extended standard basis subsequence map.

$\hat{h}_1$	$\hat{h}_2$	$\hat{h}_3$	$\hat{h}_4$	$\hat{h}_5$	$\hat{h}_6$	$\hat{h}_7$
00 00 10	00 01 00	00 00 00	00 00 00	00 00 00	00 00 01	00 00 00
10 00 00	01 00 01	00 00 00	00 00 01	00 00 00	00 00 00	00 00 00
00 00 00	10 01 00	00 00 00	10 01 10	10 00 00	01 00 00	10 00 00
00 00 00	10 00 00	00 00 10	00 00 00	00 01 00	01 00 01	00 00 00
00 00 00	00 00 00	10 00 00	00 00 10	10 00 01	00 00 00	11 00 00
00 00 00	00 00 00	00 00 00	00 00 00	10 01 00	00 00 01	00 00 00
00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	10 00 00
00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 00 00
00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 00 00
$\hat{h}_8$	$\hat{h}_9$	$\hat{h}_{10}$	$\hat{h}_{11}$	$\hat{h}_{12}$	$\hat{h}_{13}$	$\hat{h}_{14}$
00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 00 00
00 00 01	00 00 00	00 01 00	00 00 01	00 00 00	00 00 01	00 00 00
00 01 00	00 10 00	01 10 00	00 01 00	10 00 00	00 01 00	00 00 00
00 01 00	00 00 00	01 10 00	00 01 00	00 01 00	00 01 00	00 00 01
10 00 00	00 11 00	00 10 00	10 00 00	10 00 00	10 00 00	00 00 00
10 01 00	00 00 00	00 10 00	10 01 00	00 00 10	00 01 00	01 00 00
00 00 00	00 10 00	00 00 00	00 00 10	00 00 00	00 01 00	01 00 01
10 00 00	00 00 00	00 10 00	00 00 00	00 00 10	00 00 01	00 00 00
00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	10 01 00	00 00 01

(a) M_g: 2-D composite sequences obtained by solving for the null space of V.

$\mathcal{R}(\hat{h}_1)$	$\mathcal{R}(\hat{h}_2)$	$\mathcal{R}(\hat{h}_3)$	$\mathcal{R}(\hat{h}_4)$	$\mathcal{R}(\hat{h}_5)$	$\mathcal{R}(\hat{h}_6)$	$\mathcal{R}(\hat{h}_7)$
00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 00 10
00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 00 00
00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 00 00
00 00 00	00 00 10	10 00 00	00 00 00	00 01 00	01 00 01	00 00 00
00 00 00	00 00 00	00 00 10	10 00 00	01 00 10	00 00 00	00 00 11
00 00 00	00 00 00	00 00 00	00 00 00	00 01 10	01 00 00	00 00 00
10 00 00	00 01 00	00 00 00	00 00 00	00 00 00	01 00 00	00 00 00
00 00 10	01 00 01	00 00 00	01 00 00	00 00 00	00 00 00	00 00 00
00 00 00	00 01 10	00 00 00	10 01 10	00 00 10	00 00 01	00 00 10
$\mathcal{R}(\hat{h}_8)$	$\mathcal{R}(\hat{h}_9)$	$\mathcal{R}(\hat{h}_{10})$	$\mathcal{R}(\hat{h}_{11})$	$\mathcal{R}(\hat{h}_{12})$	$\mathcal{R}(\hat{h}_{13})$	$\mathcal{R}(\hat{h}_{14})$
00 00 00	00 10 00	00 00 00	10 00 00	00 00 00	00 01 00	01 00 01
00 00 10	00 00 00	00 10 00	00 00 00	10 00 00	01 00 00	00 00 00
00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 01 10	01 00 00
00 01 00	00 00 00	00 10 01	00 01 00	00 01 00	00 01 00	01 00 00
00 00 10	00 11 00	00 10 00	00 00 10	00 00 10	00 00 10	00 00 00
00 01 10	00 00 00	00 10 00	00 01 10	10 00 00	00 01 00	00 00 01
00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 00 00
01 00 00	00 00 00	00 01 00	01 00 00	00 00 00	01 00 00	00 00 00
00 01 00	00 10 00	00 10 01	00 01 00	00 00 10	00 01 00	00 00 00

(b) $\mathcal{R}(M_g)$: Spanning set of the dual code.

Figure 6.2: 2-D representation of the null space of V.

$\mathcal{U}_{fr}^{-1}(\bar{h}_r) = \hat{h}_r \in S^6$ to form the set M_g shown in Figure 6.2(a). The reverse composite form $\mathcal{R}(\hat{h}_r)$ of these sequences shown in Figure 6.2(b) is the spanning set of the dual code $\mathcal{C}^{\perp}$.

The rank of the dual code is $n - k = 4$, so only 4 out of these 14 sequences are linearly independent. Notice that the sequences $\mathcal{R}(\hat{h}_{11}), \mathcal{R}(\hat{h}_{12}), \mathcal{R}(\hat{h}_{13})$, and $\mathcal{R}(\hat{h}_{14})$ in Figure 6.2(b) have unique "ones" in the top left corner (i.e. the $(0,0)$ coordinate). This makes them R-linearly independent because they cannot be expressed in the form (6.26). Therefore, they are a basis of the dual code. These 4 sequences represent the composite sequence generator of the dual code.

$$
\begin{aligned}
H_c &= \left\{ \mathcal{R}(\hat{h}_{11}), \mathcal{R}(\hat{h}_{12}), \mathcal{R}(\hat{h}_{13}), \mathcal{R}(\hat{h}_{14}) \right\} \\
&= \{h_1, h_2, h_3, h_4\} \\
&= \left\{ \begin{array}{cccc}
\begin{matrix} 10\ 00\ 00 \\ 00\ 00\ 00 \\ 00\ 00\ 00 \end{matrix} &
\begin{matrix} 00\ 00\ 00 \\ 10\ 00\ 00 \\ 00\ 00\ 00 \end{matrix} &
\begin{matrix} 00\ 01\ 00 \\ 01\ 00\ 00 \\ 00\ 01\ 10 \end{matrix} &
\begin{matrix} 01\ 00\ 01 \\ 00\ 00\ 00 \\ 01\ 00\ 00 \end{matrix} \\
\begin{matrix} 00\ 01\ 00 \\ 00\ 00\ 10 \\ 00\ 01\ 10 \end{matrix} , &
\begin{matrix} 00\ 01\ 00 \\ 00\ 00\ 10 \\ 10\ 00\ 00 \end{matrix} , &
\begin{matrix} 00\ 01\ 00 \\ 00\ 00\ 10 \\ 00\ 01\ 00 \end{matrix} , &
\begin{matrix} 01\ 00\ 00 \\ 00\ 00\ 00 \\ 00\ 00\ 01 \end{matrix} \\
\begin{matrix} 00\ 00\ 00 \\ 01\ 00\ 00 \\ 00\ 01\ 00 \end{matrix} &
\begin{matrix} 00\ 00\ 00 \\ 00\ 00\ 00 \\ 00\ 00\ 10 \end{matrix} &
\begin{matrix} 00\ 00\ 00 \\ 01\ 00\ 00 \\ 00\ 01\ 00 \end{matrix} &
\begin{matrix} 00\ 00\ 00 \\ 00\ 00\ 00 \\ 00\ 00\ 00 \end{matrix}
\end{array} \right\}
\end{aligned}
$$

Each composite sequence $h_x \in S^6$, $x = 1, \ldots, 4$ has to be deinterleaved $n_1 = 2$ symbols at a time along i_1, and $n_2 = 3$ symbols at a time along i_2 to obtain the xth row $(h_x^{(1)}, \ldots, h_x^{(6)})$ of the sequence encoder $H \in S^{4\times 6}$ of the dual code.

$$
H = \left(\begin{array}{cccccc}
\begin{matrix} 1\,0\,0 \\ 0\,0\,0 \\ 0\,0\,0, \end{matrix} &
\begin{matrix} 0\,0\,0 \\ 0\,1\,0 \\ 0\,0\,0, \end{matrix} &
\begin{matrix} 0\,0\,0 \\ 0\,0\,1 \\ 0\,0\,0, \end{matrix} &
\begin{matrix} 0\,0\,0 \\ 0\,0\,0 \\ 1\,0\,0, \end{matrix} &
\begin{matrix} 0\,0\,0 \\ 0\,0\,1 \\ 0\,0\,0, \end{matrix} &
\begin{matrix} 0\,0\,0 \\ 0\,1\,0 \\ 0\,1\,0 \end{matrix} \\
\begin{matrix} 0\,0\,0 \\ 0\,0\,0 \\ 0\,0\,0, \end{matrix} &
\begin{matrix} 0\,0\,0 \\ 0\,1\,0 \\ 0\,0\,0, \end{matrix} &
\begin{matrix} 1\,0\,0 \\ 0\,0\,1 \\ 0\,0\,0, \end{matrix} &
\begin{matrix} 0\,0\,0 \\ 0\,0\,0 \\ 0\,0\,0, \end{matrix} &
\begin{matrix} 0\,0\,0 \\ 1\,0\,0 \\ 0\,0\,1, \end{matrix} &
\begin{matrix} 0\,0\,0 \\ 0\,0\,0 \\ 0\,0\,0 \end{matrix} \\
\begin{matrix} 0\,0\,0 \\ 0\,0\,0 \\ 0\,0\,0, \end{matrix} &
\begin{matrix} 0\,1\,0 \\ 0\,1\,0 \\ 0\,0\,0, \end{matrix} &
\begin{matrix} 0\,0\,0 \\ 0\,0\,1 \\ 0\,0\,0, \end{matrix} &
\begin{matrix} 1\,0\,0 \\ 0\,0\,0 \\ 1\,0\,0, \end{matrix} &
\begin{matrix} 0\,0\,1 \\ 0\,0\,0 \\ 0\,0\,0, \end{matrix} &
\begin{matrix} 0\,1\,0 \\ 0\,1\,0 \\ 0\,1\,0 \end{matrix} \\
\begin{matrix} 0\,0\,0 \\ 1\,0\,1 \\ 0\,0\,0, \end{matrix} &
\begin{matrix} 0\,0\,0 \\ 1\,0\,0 \\ 0\,0\,0, \end{matrix} &
\begin{matrix} 0\,0\,0 \\ 0\,0\,0 \\ 0\,0\,0, \end{matrix} &
\begin{matrix} 0\,0\,0 \\ 0\,0\,0 \\ 0\,0\,0, \end{matrix} &
\begin{matrix} 0\,0\,0 \\ 0\,0\,0 \\ 0\,0\,0, \end{matrix} &
\begin{matrix} 1\,0\,0 \\ 0\,0\,1 \\ 0\,0\,0 \end{matrix}
\end{array} \right)
$$

The polynomial representation of H using the isomorphism ψ is the gives us the polynomial encoder $H(z) = \psi(H) \in R^{4\times 6}$. Alternatively, we can obtain $H(z)$ from H_c by using the

transformation ψ^6.

$$H(z) = \begin{bmatrix} 1 & z_1z_2 & z_1^2z_2 & z_2^2 & z_1^2z_2 & z_1z_2 + z_1z_2^2 \\ 0 & z_1z_2 & 1 + z_1^2z_2 & 0 & z_2 + z_1^2z_2^2 & 0 \\ 0 & z_1 + z_1z_2 & z_1^2z_2 & 1 + z_2^2 & z_1^2 & z_1 + z_1z_2 + z_1z_2^2 \\ 0 & 1 + z_1^2 + z_2 & 0 & 0 & 0 & 1 + z_1^2z_2 \end{bmatrix}$$

Finally, observe that the memory orders of $H(z)$ along i_1 and i_2 are $M_1^\perp = 2$ and $M_2^\perp = 2$. These values can be directly obtained from the memory orders M_1 and M_2 of the locally invertible encoder $G(z)$ by using the equations $M_1^\perp = \frac{k_1 M_1}{n_1 - k_1} = \frac{1\times 2}{2-1} = 2$ and $M_2^\perp = \frac{k_2 M_2}{n_2 - k_2} = \frac{2\times 1}{3-2} = 2$ without actually solving for $H(z)$. ▲

As explained in Section 3.5, the encoder of the dual code $\mathcal{C}^\perp$ acts as a parity check matrix for the code $\mathcal{C}$. Let $v = u * G$ be a codeword that is transmitted over a noisy channel. Let $r = v + e \in S^n$ be the sequence received at the output of the channel, where $e \in S^n$ is the error sequence. If we convolve the received sequence with the syndrome former $H^T \in S^{n\times(n-k)}$, we obtain the *error syndrome*

$$s = v * H^T. \tag{6.32}$$

Discrete m-D convolution with the syndrome former is defined as

$$\begin{aligned} s^{(1)} &= v^{(1)} * h_1^{(1)} + \cdots + v^{(n)} * h_1^{(n)} \\ &\vdots \\ s^{(n-k)} &= v^{(1)} * h_{n-k}^{(1)} + \cdots + v^{(n)} * h_{n-k}^{(n)} \end{aligned}$$

The sequences $s^{(1)}, \ldots, s^{(n-k)} \in S$ are multiplexed to form the composite syndrome $s \in S^{n-k}$. Since $v = u * G$, and $G * H^T = 0$, the syndrome

$$s = v * H^T = (v + e) * H^T = e * H^T \tag{6.33}$$

will be an all-zero sequence if and only if e is a codeword. Furthermore, the syndrome depends only on the error sequence and is independent of the transmitted codeword. The decoding rule using the syndrome former can be simply a map from the unique syndromes to the apparent errors. In the 1-D case, the table-driven decoder [6, 16, 11, 15, 14] uses this approach to perform error correction. This decoding technique can easily be extended to m-D convolutional codes by using the m-D syndrome former (6.28) that we have derived in this section.

Chapter 7

Conclusion

7.1 Summary

In this dissertation we introduced a sequence space framework for describing m-D convolutional codes. Using this framework, we defined notions of encoder constraint lengths, memory orders and sequence space ordering that had not been considered in existing literature which focused mainly on algebraic aspects of m-D convolutional codes. We built on the concept of sequence space ordering and generalized 1-D locally invertible encoders introduced in [6] to higher dimensions by defining m-D locally invertible encoders.

Locally invertible encoders have the unique characteristic of an invertible subsequence map that can be used to recover input sequences from their corresponding codewords. The invertible subsequence map also endows locally invertible encoders with many useful structural properties that are desirable in convolutional encoders. In particular, m-D locally invertible encoders are basic (i.e. ZP), and for the case $m = 1$, 1-D locally invertible encoders are minimal-basic. The subsequence map of a locally invertible encoder is described by a reduced encoding matrix that has a well defined structure. For appropriate values code rate and memory orders, this structure can be exploited to explicitly construct locally invertible encoders of any dimension. The inverse of the reduced encoding matrix can be used to compute an encoder inverse and a set of pseudo-inverses. These pseudo-inverses can be used for error detection. The subsequence mapping property can also be used to compute a set of parity-check sequences which can be used to generate error syndromes. A subset of these parity-check sequences can be used to construct the encoder of the dual code. The main advantage of the techniques described in this dissertation are that all operations

involving encoder construction, computation of inverses, pseudo inverses and parity-check matrices are performed using elementary operation on the ground field without the use of any polynomial operations.

7.2 Future Directions

Compared to the well developed field of 1-D convolutional codes, the subject of m-D ($m > 1$) convolutional codes is fairly new and largely unexplored. Most importantly, work on finding distance parameters and implementation of efficient decoding algorithms remains to be undertaken. A list of research problems that need to be further investigated can be found in [56, Section 6.2] and [9, 10]. The focus of this dissertation was on the construction of locally invertible encoders, characterization of their structural properties and their use as encoders for generating m-D convolutional codes. In this section we identify two areas relating to local invertibility that we plan to address in future work.

In Section 3.6 we mentioned that the class of minimal-basic encoders are of particular importance because they require the least number delay elements over all equivalent basic encoding matrices. Unlike the 1-D case (see Theorem 3.6.2), minimal-basic m-D convolutional encoders have not been characterized [10], and a straightforward algebraic algorithm [30, pg. 58] for the constructing such encoders is not yet available. We have shown in Theorem 5.4.7 that 1-D locally invertible encoders are minimal-basic. The proof of the theorem makes use of the fact that a necessary condition for a 1-D encoder to be locally invertible is that the constraint lengths of all rows have to be equal and the resulting indicator matrix has to have full row rank. As observed in Condition 5.3.1, this phenomenon carries over to locally invertible encoders of higher dimensions in a rather interesting way. For example, a necessary condition for a 3-D encoder to be locally invertible is that the constraint lengths of all the rows along a given dimension have to be equal. That is, $\nu_{x,i} = M_i$ for all $x = 1, \ldots, k$. In addition to this, the constant 1 and the monomials $z_1^{M_1}, z_2^{M_2}, z_3^{M_3}, z_1^{M_1} z_2^{M_2}, z_1^{M_1} z_3^{M_3}, z_2^{M_2} z_3^{M_3}, z_1^{M_1} z_2^{M_2} z_3^{M_3}$ have to appear as nonzero terms in at least one component of each row of the encoder, and the indicator matrices (5.26) corresponding to each of these terms have to be of full row rank. If these conditions are not satisfied the reduced encoding matrix will have an all-zero row, thus making it singular, and the encoder will not be locally invertible. Due to this characteristic, we believe that it might not be possible to reduce either the total degree or the constraint lengths of a locally

invertible encoder any further to obtain an equivalent encoder that requires a fewer number of delay elements. This interesting feature of m-D locally invertible encoders warrants further investigation.

Traditional trellis based approaches used for decoding 1-D convolutional codes do not generalize in a straightforward manner to higher dimensions. In the 1-D case, the Viterbi and BCJR algorithms exploit the natural ordering of 1-D codewords to compute maximum likelihood (ML) and maximum a posteriori (MAP) estimates. However, for dimensions two and higher, since there is no natural notion of causality inducing a particular ordering on the paths traversed through the trellis, the size of the search space grows prohibitively large, making a direct generalization of existing 1-D ML and MAP decoding techniques computationally intractable even for the two dimensional case. Consequently, practical decoding techniques for m-D convolutional codes (except for codes generated by unit memory encoders [56, 57], where $M_1{=}M_2{=}\cdots{=}M_m{=}1$) have not yet been realized. In the absence of such decoding algorithms for m-D convolutional codes, a simpler syndrome decoding scheme can be adopted. The Table-driven decoder [6, 16] for 1-D convolutional codes generated by locally invertible encoders, is one such decoding technique that uses this approach. Table based decoding is performed as a two step process. In the first step, a syndrome former is used to generate error syndromes from the received sequence. These error syndromes are used to index a pre-computed error correction table to perform error correction on the received sequence, so as to obtain an estimate of the transmitted codeword. Inversion of the estimated codeword is then performed as a separate step, using the inverse subsequence map of the locally invertible encoder, to obtain an estimate of the input sequence that was encoded. This approach generalizes to m-D convolutional codes generated by locally invertible encoders in a straightforward manner. The parity check matrix derived in Section 6.3 can be used as a syndrome former to generate error syndromes, and the inverse subsequence map of the m-D locally invertible encoder can be used as a sequence inversion technique as described in Section 5.2. As future work we plan to build the error correction tables mentioned above and extend the table based approach for decoding m-D convolutional codes.

Bibliography

[1] A. G. Akritas. Sylvester's forgotten form of the resultant. *The Fibonacci Quarterly*, 31:325–332, 1993.

[2] G. E. Antoniou. Minimal state space realization for all-pole and all-zero lattice discrete 2D filters. *International Journal of Systems Science*, 33:799–803, 2002.

[3] L. Bahl and F. Jelinek. On the structure of rate 1/n convolutional codes. *IEEE Trans. Inform. Theory*, 18(1):192–196, January 1972.

[4] D.L. Bitzer, A. Dholakia, H. Koorapaty, and M.A. Vouk. Rate-1/n locally invertible convolutional encoders. In *Proc. IEEE International Symposium on Information Theory*, page 19, Trondheim, Norway, June 1994.

[5] D.L. Bitzer, A. Dholakia, H. Koorapaty, and M.A. Vouk. On locally invertible rate-1/n convolutional encoders. *IEEE Trans. Inform. Theory*, 44(1):420–422, January 1998.

[6] D.L. Bitzer and M.A. Vouk. A table-driven (feedback) decoder. In *Proc. Intl. Phoenix Conf. on Comp. Comm.*, pages 385–392, Phoenix, AZ, 1991.

[7] N. K. Bose, editor. *Multidimensional Systems Theory and Applications*. Kluwer Academic Pub, 2 edition, 2003.

[8] C. Charoenlarpnopparut and N.K. Bose. Multidimensional FIR filter bank design using Gröbner bases. *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, 46(12):1475–1486, Dec 1999.

[9] C. Charoenlarpnopparut and S. Tantaratana. Multidimensional convolutional code: progresses and bottlenecks. In *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS 2003)*, pages III–686 to III–689, Bangkok, Thailand, May 2003.

[10] C. Charoenlarpnopparut and S. Tantaratana. Algebraic approach to reduce the number of delay elements in the realization of multidimensional convolutional code. In *Proc. 47th IEEE Int. Midwest Symp. Circuits and Systems (MWSCAS 2004)*, pages II–529 to II–532, Hiroshima, Japan, July 2004.

[11] A. Dholakia. *Locally Invertible Convolutional Encoders, Table-Based Decoding, and their Applications to High Speed Communications*. PhD thesis, ECE, NCSU, 1993.

[12] A. Dholakia. *Introduction to Convolutional Codes*. Kulwer Academic, Dordrecht, The Netherlands, 1994.

[13] A. Dholakia, D.L. Bitzer, H. Koorapaty, and M.A. Vouk. Minimal, minimal-basic, and locally invertible convolutional encoders. In *Proc. of the IEEE International Symposium on Information Theory*, page 164, Whistler, Canada, September 1995.

[14] A. Dholakia, T.M. Lee, D.L. Bitzer, M.A. Vouk, L. Wang, and P.D. Franzon. An efficient table-driven decoder for one-half rate convolutional codes. In *Proc. 30th ACM Annual Southeast Conference*, pages 116–123, 1992.

[15] A. Dholakia, M.A. Vouk, and D.L. Bitzer. Table-driven decoding of binary one-half rate nonsystematic convolutional codes. In *proceedings of the IEEE International Symposium on Information Theory*, page 270, 1993.

[16] A. Dholakia, M.A. Vouk, and D.L. Bitzer. Table based decoding of rate one-half convolutional codes. *IEEE Trans. Commun.*, 43(2-4):681–686, 1995.

[17] P. Elias. Coding for noisy channels. *IRE Conv. Rec.*, 4:37–46, 1955.

[18] E. Fornasini and M. E. Valcher. Algebraic aspects of 2D convolutional codes. *IEEE Trans. Inform. Theory*, IT-40(4):1068–1082, 1994.

[19] E. Fornasini and M. E. Valcher. Duality analysis of 2D convolutional codes. In *Proc. IEEE Int. Symp. Information Theory*, page 23, Trondheim, Norway, June 1994.

[20] E. Fornasini and M.E. Valcher. nD polynomial matrices with applications to multidimensional signal analysis. *Multidimensional Systems and Signal Processing*, 8:387–408, 1997.

[21] E. Fornasini and M.E. Valcher. Multidimensional systems with finite support behaviors: Signal structure, generation, and detection. *SIAM J. Control Optim.*, 36(2):760–779, 1998.

[22] G. D. Forney Jr. Convolutional codes I: Algebraic structure. *IEEE Trans. Inform. Theory*, IT-16(5):720–738, 1970.

[23] P.A. Fuhrmann. *A Polynomial Approach to Linear Algebra*. Springer, 1996.

[24] K. Galkowski. Multi-dimensional systems theory – a common approach to problems in circuits, control and signal processing. In *IEE Colloquium on Multi-Dimensional Systems: Problems and Solutions*, page 5/1, 1996.

[25] F.R. Gantmacher. *Theory of Matrices*. Chelsea, New York, 1990.

[26] Jr. G.D. Forney. Structural analysis of convolutional codes via dual codes. *IEEE Trans. Inform. Theory*, IT-19(5):512–518, 1973.

[27] H. Gluesing-Luerssen, J. Rosenthal, and P. A. Weiner. Duality between multidimensional convolutional codes and systems. In F. Colonius, U. Helmke, F. Wirth, and D. Prätzel-Wolters, editors, *Advances in Mathematical Systems Theory, A Volume in Honor of Diederich Hinrichsen*, pages 135–150. Birkhauser, Boston, 2000.

[28] R.W. Hamming. Error detecting and error correcting codes. *Bell System Tech. J.*, 29(2):147–160, 1950.

[29] R. Johannesson and Z.-x. Wan. A linear algebra approach to minimal convolutional encoders. *IEEE Trans. Inform. Theory*, 39(4):1219–1233, 1993.

[30] R. Johannesson and K.S. Zigangirov. *Fundamentals of Convolutional Coding*. IEEE Press, Piscataway. N. J., 1999.

[31] T. Kailath. *Linear Systems*. Prentice-Hall, 1980.

[32] B. Kitchens. Multidimensional convolutional codes. *SIAM Journal on Discrete Mathematics*, 15(3):367–381, 2002.

[33] M.A. Laidacker. Another theorem relating sylvester's matrix and the greatest common divisor. *Math. Mag.*, 42:126–128, 1969.

[34] T.Y. Lam. *Serre's Conjecture*, volume 635 of *Lecture Notes in Mathematics*. Springer, Berlin, 1978.

[35] S. Lin and Jr. D.J. Costello. *Error Control Coding: Fundamentals and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1983.

[36] Z. Lin. Notes on n-d polynomial matrix factorizations. *Multidimensional Systems and Signal Processing*, 10(4):379–393, 1999.

[37] Z. Lin and N. K. Bose. A generalization of Serre's conjecture and some related issues. *Linear Algebra and Its Applications*, 338:125–138, November 2001.

[38] R. Lobo, D. Bitzer, and M. Vouk. Locally invertible multidimensional convolutional encoders. Submitted for review to IEEE Transactions on Information Theory.

[39] R. Lobo, D. Bitzer, and M. Vouk. Inverses of multivariate polynomial matrices using discrete convolution. In *Proceedings of the 2005 International Workshop on Coding and Cryptography (WCC 2005), Bergen, Norway*, pages 481–490, March 2005.

[40] R. Lobo, D. Bitzer, and M. Vouk. Locally invertible multivariate polynomial matrices. *Lecture Notes in Computer Science*, 3969:427–441, June 2006.

[41] J.L. Massey and M.K. Sain. Inverses of linear sequential circuits. *IEEE Trans. on Computers*, C-17(4):330–337, 1968.

[42] R.J. McEliece. The algebraic theory of convolutional codes. In V. Pless and W.C. Huffman, editors, *Handbook of Coding Theory*, volume 1, pages 1065–1138. Elsevier Science Publishers, Amsterdam, The Netherlands, 1998.

[43] W. Mingsheng and C.P. Kwong. On multivariate polynomial matrix factorization problems. *Mathematics of Control, Signals, and Systems (MCSS)*, 17:297–311, 2005.

[44] M. Morf, B.C. Levy, and S. Kung. New results in 2-D systems theory, part I: 2-D polynomial matrices, factorization, and coprimeness. *Proc. IEEE*, 65(6):861–872, 1977.

[45] Ph. Piret. *Convolutional Codes, an Algebraic Approach*. MIT Press, Cambridge, MA, 1988.

[46] J. F. Pommaret. Solving Bose conjecture on linear multidimensional systems. In *European Control Conference*, pages 1853–1855, September 2001.

[47] H.H. Rosenbrock. *State-Space and Multivariable Theory*. Wiley, New York, 1970.

[48] J. Rosenthal. Connections between linear systems and convolutional codes. In B. Marcus and J. Rosenthal, editors, *Codes, Systems and Graphical Models*, IMA Vol. 123, pages 39–66. Springer-Verlag, 2001.

[49] B. De Schutter. Minimal state-space realization in linear system theory: an overview. *J. Comput. Appl. Math.*, 121(1-2):331–354, 2000.

[50] C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, July and October 1948.

[51] N.J. A. Sloane and A.D. Wyner. *Claude Elwood Shannon: collected papers*. IEEE Press, Piscataway, NJ, USA, 1993.

[52] V. Srinivas. A generalized Serre problem. *J. Algebra*, 278(2):621–627, Aug 2004.

[53] J.J. Sylvester. On a theory of the syzygetic relations of two rational integral functions, comprising an application to the theory of sturm's functions, and that of the greatest algebraical common measure. *Philosophical Transactions of the Royal Society of London*, Vol. 143:407–548, 1853.

[54] M.E. Valcher and E. Fornasini. On 2D finite support convolutional codes: An algebraic approach. *Multidimensional Systems and Signal Processing*, 5(3):231–243, 1994.

[55] M. Wang and D. Feng. On Lin-Bose problem. *Linear Algebra and Its Applications*, 390:279–285, Oct 2004.

[56] P. A. Weiner. *Multidimensional Convolutional Codes*. PhD thesis, University of Notre Dame, April 1998.

[57] P. A. Weiner. Basic properties of multidimensional convolutional codes. In *Codes, Systems and Graphical Models*, volume 123 of *IMA Volumes in Mathematics and Its Applications*, chapter 4, pages 397–414. Springer-Verlag, New York, 2001.

[58] J. Wood, E. Rogers, and D.H. Owens. A formal theory of matrix primeness. *Mathematics of Control, Signals, and Systems*, 11:40–78, 1998.

[59] L. Xu, L. Wu, Q. Wu, Z. Lin, and Y. Xiao. On realization of 2D discrete systems by fornasini-marchesini model. *International Journal of Control, Automation, and Systems,*, 3(4):631–639, Dec 2005.

[60] D.C. Youla and G. Gnavi. Notes on n-dimensional system theory. *IEEE Trans. Circuits Syst.*, 26:105–111, 1979.

[61] D.C. Youla and P.F. Pickel. The Quillen-Suslin theorem and the structure of n-dimensional elementary polynomial matrices. *IEEE Trans. Circuits and Systems*, 31(6):513–517, 1984.

[62] E. Zerz. Primeness of multivariate polynomial matrices. *Systems and Control Letters*, 29(3):139–145, 1996.